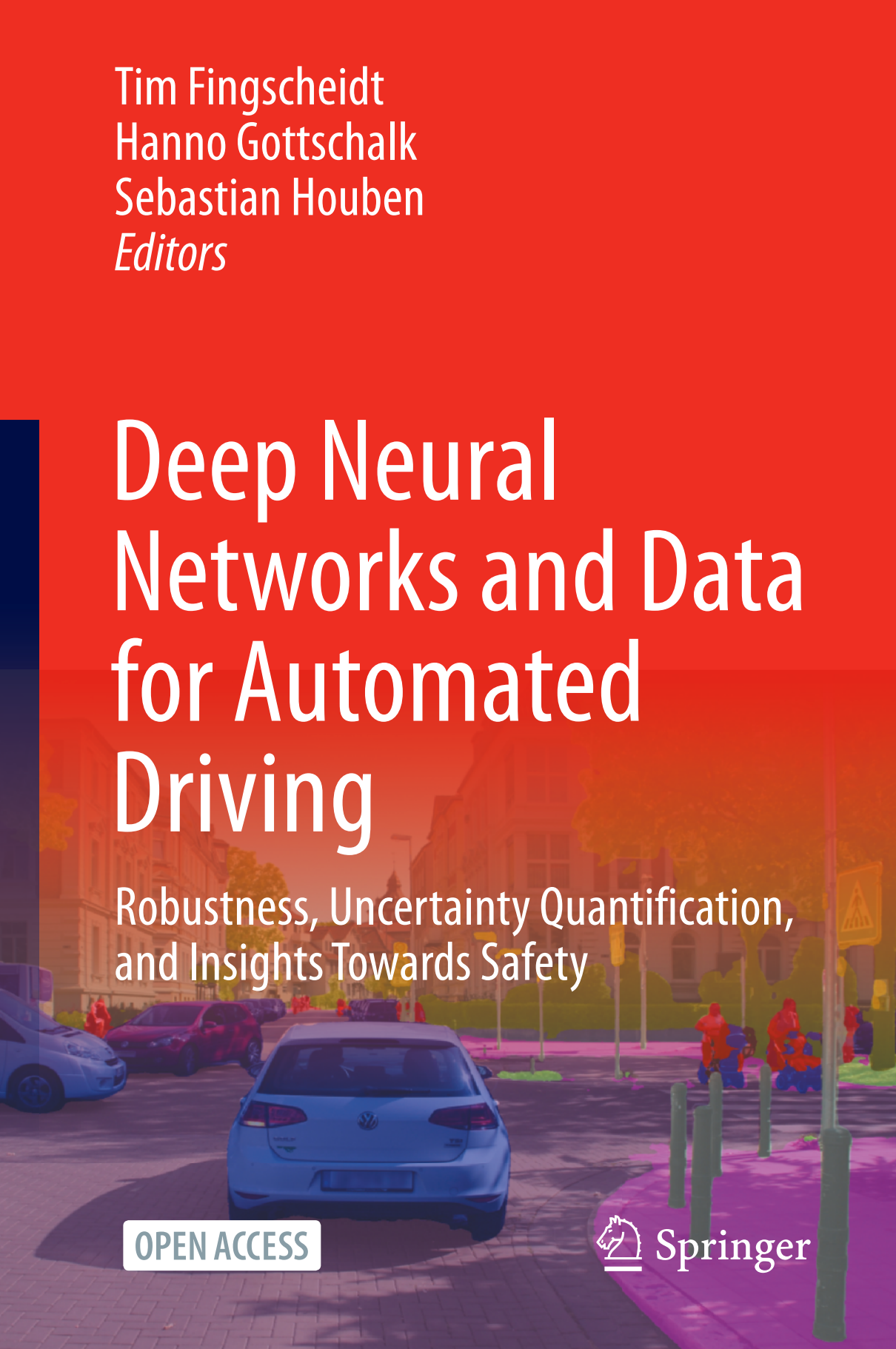
Tim Fingscheidt
Hanno Gottschalk
Sebastian Houben
Editors

Deep Neural Networks and Data for Automated Driving

Robustness, Uncertainty Quantification, and Insights Towards Safety



OPEN ACCESS

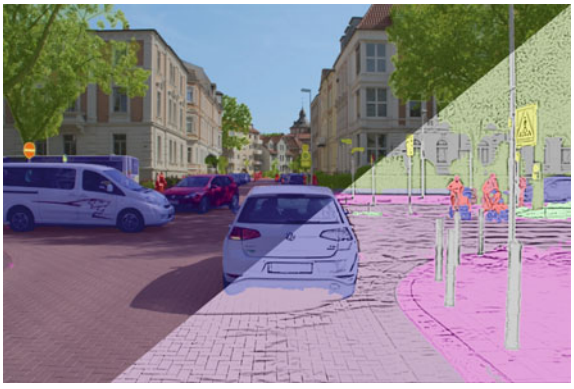
Springer

Deep Neural Networks and Data for Automated Driving

Tim Fingscheidt · Hanno Gottschalk ·
Sebastian Houben
Editors

Deep Neural Networks and Data for Automated Driving

Robustness, Uncertainty Quantification, and Insights Towards Safety



Springer

Editors
Tim Fingscheidt
Institute for Communications Technology
Technische Universität Braunschweig
Braunschweig, Germany

Hanno Gottschalk
Fachgruppe Mathematik und Informatik
Bergische Universität Wuppertal
Wuppertal, Germany

Sebastian Houben
Schloss Birlinghoven
Fraunhofer Institute for Intelligent Analysis
and Information Systems IAIS
Sankt Augustin, Germany

ISBN 978-3-031-01232-7 ISBN 978-3-031-01233-4 (eBook)
https://doi.org/10.1007/978-3-031-01233-4

Front page picture kindly provided by Jan-Aike Termöhlen and Andreas Bär, © Jan-Aike Termöhlen and Andreas Bär

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

I am thankful to the editors for sending me a prerelease copy of their book and for their kind invitation to write a foreword. I have greatly benefitted by reading the book and learning fresh ideas shared by distinguished experts from academia as well as professionals with serious engagement in the development of new generations of automobiles and driver assistance systems.

I have been fortunate to enjoy a long academic career pursuing a research agenda involving a number of topics discussed in the book. These include computer vision, intelligent vehicles and systems, machine learning, and human–robot interactions. For the past two decades, my team and collaborators have focused our explorations on developing human-centered, **safe** intelligent vehicles and transportation systems. Therefore, I am especially pleased to read this book presenting latest developments, findings, and insights in automated driving with *towards safety* as a highlighted term in the title and as a thread binding various chapters in the book. Emphasis on safety as an essential requirement was not always the case in the past. Reports of serious accidents involving automobiles with inadequate capabilities for automated driving are making safety an uncompromising requirement.

However, progress in the field does not occur at a consistent, predictable manner or rate. In the early phase, spanning around 1980–2000, autonomous vehicle research activities were mostly limited to a handful of universities and industrial labs. Important insights and milestones realized during this phase were novel perception and control systems for lane detection, cruise control, and carefully designed demonstrations of "autonomous" highway driving. However, interest and sponsorship of research in the field diminished and real-world deployment was unrealized.

In the next phase of autonomous vehicles research, spanning about mid-2000 to 2015, key enabling technologies, such as Global Positioning Systems and cost-effective mass production of sensors (radars, cameras) and embedded processors, not only invigorated serious activities in academic and technical communities, but also sparked interest and commitments by venture and commercial sectors.

It seemed that this time, our field was on an irreversible forward trajectory. Important insights and milestones realized during this phase include the introduction of active safety systems like dynamic stability control, and practical driver assistance

systems, such as lane keep assistance, collision avoidance, pedestrian protection, panoramic viewing, and other camera-based enhancements. We also started to take note of the great strides made in the machine learning and computer vision fields. Some of the major limitations encountered in the autonomous vehicles field, such as "brittle" object detectors trained using traditional, hand-crafted features, seemed resolvable with new developments in machine learning and computer vision.

Topics covered in this book are from the latest phase (from 2015 onwards) of autonomous vehicle research activities. Authors clearly acknowledge the important and essential roles of deep neural networks (DNN) and data-driven intelligent systems in making new contributions. They also recognize that there are unresolved challenges laying ahead. More focused research of both analytical and experimental nature is required. As a long-term proponent of *safe, human-centered* intelligent vehicles, I appreciate that the book places prominent emphasis on safety, integrated systems, robustness, performance evaluation, experimental metrics, benchmarks, and protocols. This makes the book very unique and an essential reading for students and scholars from multiple disciplines including engineering, computer/data/cognitive sciences, artificial intelligence, machine learning, and human/machine systems.

This is a very important book, as it deals with a topic surrounded with great excitement, not only in the technical communities, but also in the minds of people at large. With great excitement and wide popular interest in the autonomous driving field, often there is a temptation to overlook or ignore unresolved technical challenges. Given grave consequences of failures of complex systems deployed in the real world inhabited by humans, designers, engineers, and promoters of autonomous driving systems, we need to be most careful and attentive to the topics presented in the book. The editors and authors have done an outstanding job in presenting important themes and carefully selected topics in a systematic, insightful manner. The book identifies promising avenues for focused research and development, as well as acknowledges outstanding challenges yet to be resolved.

I congratulate Prof. Tim Fingscheidt, Prof. Hanno Gottschalk, and Prof. Sebastian Houben, for their vision and efforts in organizing an outstanding team of authors with deep academic and industrial backgrounds in developing this special book. I am confident that it will prove to be of great value in realizing a new generation of safer, smoother, trustworthy automobiles.

La Jolla, CA, USA Mohan M. Trivedi

Mohan M. Trivedi Distinguished Professor of Engineering University of California, San Diego, founding director of the Computer Vision and Robotics Research (CVRR) and the Laboratory for Intelligent and Safe Automobiles (LISA). Fellow IEEE, SPIE, and IAPR.

Preface

This book addresses readers from both academia and industry, since it is written by authors from both academia and industry. Accordingly, it takes on diverse viewing angles, but keeps a clear focus on machine-learned environment perception in autonomous vehicles. Special interest is on deep neural networks themselves, their robustness and uncertainty awareness, the data they are trained on, and, last but not least, on safety aspects.

The book is also special in its structure. While a first part is an extensive survey of literature in the field, the second part consists of 14 chapters, each detailing a particular aspect in the area of interest. This book wouldn't exist without the large-scale national flagship project KI Absicherung (safe AI for automated driving), with academia, car manufacturers, and suppliers being involved to work towards robustness and safety arguments in machine-learned environment perception for autonomous vehicles. I came up with the idea of a book project when I saw a first draft of a wonderful survey that was prepared by Sebastian Houben in collaboration with many partners from the project. My immediate thought was that this should not only be some project deliverable, it should become the core of a book (namely, Part I), appended with a number of novel contributions on specific topics in the field (namely, Part II). Beyond Sebastian Houben, I am happy that Hanno Gottschalk agreed to join the editorial team with his expertise in uncertainty and safety aspects. We thank KI Absicherung project leader (PL) Stephan Scholz (Volkswagen), Michael Mock (Scientific PL, Fraunhofer IAIS), and Fabian Hüger (Part PL, Volkswagen) very much, since without their immediate and ongoing support this book project wouldn't have come alive. We also thank the many reviewers for the time and energy they put into providing valuable feedback on the initial versions of the book chapters.

But the book reaches beyond the project. There are several contributions from outside the project enriching the diversity of views. Aiming at increasing impact, we editors decided to make the book an open-access publication. The funds for the open-access publication have been covered by the German Federal Ministry for Economic Affairs and Energy (BMWi) in the framework of the "Safe AI for Automated Driving" research consortium. Beyond that we are very grateful for the financial support both from Bergische Universität Wuppertal and Technische Universität Braunschweig.

A very special thanks goes to Andreas Bär, TU Braunschweig, for his endless efforts in running the editorial office, for any extra proofreading, formatting, and for keeping close contact to all the authors.

Tim Fingscheidt, in the name of the editorial team, jointly with Hanno Gottschalk and Sebastian Houben

Braunschweig, Germany — Tim Fingscheidt
Wuppertal, Germany — Hanno Gottschalk
Sankt Augustin, Germany — Sebastian Houben
November 2021

About This Book

An autonomous vehicle must be able to perceive its environment and react adequately to it. In highly automated vehicles, camera-based perception is increasingly performed by artificial intelligence (AI). One of the greatest challenges in integrating these technologies in highly automated vehicles is to ensure the functional safety of the system. Existing and established safety processes cannot simply be transferred to machine learning methods.

In the nationally funded project KI Absicherung, which is funded by the German Federal Ministry for Economic Affairs and Energy (BMWi), 24 partners from industry, academia, and research institutions collaborate to set up a stringent safety argumentation, with which AI-based function modules (AI modules) can be secured and validated for highly automated driving. The project brings together experts from the fields of artificial intelligence and machine learning, functional safety, and synthetic sensor data generation.

The topics and focus of this book have a large overlap with the field of research being tackled in subproject three of the KI Absicherung project, in which methods and measures are developed to identify and reduce inherent insufficiencies of the AI modules. We consider such methods and measures to be key elements in supporting the general safeguarding AI modules in a car.

We gratefully thank the editors of this book as well as Springer Verlag for making this research accessible to a broad audience and wish the reader interesting insights and new research ideas.

Stephan Scholz, Michael Mock, and Fabian Hüger

Introduction

Environment perception for driver assistance systems, highly automated driving, and autonomous driving has seen the same disruptive technology paradigm shift as many other disciplines: machine learning approaches and along with them deep neural networks (DNNs) have taken over and oftentimes replaced classical algorithms from signal processing, tracking, and control theory.

Along with this, many questions arise. Of fundamental interest is the question, how safe an artificial intelligence (AI) system actually is. Part I of this book provides an extensive literature survey introducing various relevant aspects to the reader. It not only provides guidance on how to inspect and understand deep neural networks, but also on how to overcome robustness issues or safety problems. Part I will also provide motivations and links to Part II of the book, which features novel approaches in various aspects of deep neural networks in the context of environment perception, such as their robustness and uncertainty awareness, the data they are trained on, and, last but not least, again, it will be about safety.

It all starts from choosing the right *data* from the right set of sensors, and, very important, the right amount of training data. Chapter "Does Redundancy in AI Perception Systems Help to Test for Super-Human Automated Driving Performance?" (p. 81) will provide insights into the problem of obtaining statistically sufficient test data and will discuss this under the assumption of redundant and independent (sensor) systems. The reader will learn about the curse of correlation in error occurrences of redundant sub-systems. A second aspect of dataset optimization is covered in Chapter "Analysis and Comparison of Datasets by Leveraging Data Distributions in Latent Spaces" (p. 107). Investigating variational autoencoder latent space data distributions, it will present a new technique to assess the dissimilarity of the domains of different datasets. Knowledge about this will help in mixing diverse datasets in DNN training, an important aspect to keep both training efficiency and generalization at a high level. When training or inferring on real data, outlier detection is important and potentially also to learn so far unknown objects. Chapter "Detecting and Learning the Unknown in Semantic Segmentation" (p. 277) presents a method to detect anomalous objects by statistically analyzing entropy responses in semantic segmentation to suggest new semantic categories. Such method can also be

used to pre-select diverse datasets for training. Chapter "Optimized Data Synthesis for DNN Training and Validation by Sensor Artifact Simulation" (p. 127) follows the path of training with synthetic data, which has high potential, simply because rich ground truth is available for supervised learning even on sequence data. The authors are concerned with intelligent strategies of augmentation and use an appropriate metric to better match synthetic and real data, thereby increasing performance of synthetically trained networks.

A second focus area of this book is *robustness*. Robustness comprises the ability of a DNN to keep high performance also under adverse conditions such as varying weather and light conditions, dirt on the sensors, any kind of domain shift, or even under adversarial attacks. In Chapter "Improved DNN Robustness by Multi-task Training with an Auxiliary Self-Supervised Task" (p. 149), it is shown how intelligent multi-task training can be employed to strengthen a semantic segmentation DNN against various noise types and against adversarial attacks. An important aspect is that the supporting task of depth estimation does not require extra ground truth labels. Among the adversarial attacks, the universal ones are particularly dangerous as they can be applied to any current sensor input. Chapter "Improving Transferability of Generated Universal Adversarial Perturbations for Image Classification and Segmentation" (p. 171) addresses a way to generate these dangerous attacks both for semantic segmentation and for classification tasks, and should not be missed in any test suite for environment perception functions. Obtaining compact and efficient networks by compression techniques typically comes along with a degradation in performance. Not so in Chapter "Joint Optimization for DNN Model Compression and Corruption Robustness" (p. 405): The authors present a joint pruning and quantization framework which yields more robust networks against various corruptions. Mixture-of expert architectures for DNN aggregation can be a suitable means for improved robustness but also interpretability. Chapter "Evaluating Mixture-of-Experts Architectures for Network Aggregation" (p. 315) also shows that the models deal favorably with out-of-distribution (OoD) objects.

Uncertainty and interpretability are third focus areas of the book. Both concepts are fundamental towards self-awareness of environment perception for autonomous driving. In Chapter "Invertible Neural Networks for Understanding Semantics of Invariances of CNN Representations" (p. 197), invertible neural networks are deployed to recover invariant representations present in a trained network. They allow for a mapping to semantic concepts and thereby interpretation (and manipulation) of inner model representations. Confidence calibration, on the other hand, is an important means to obtain reliable confidences for further processing. Chapter "Confidence Calibration for Object Detection and Segmentation" (p. 225) proposes a multivariate calibration approach not only suitable for classification but also for semantic segmentation and object detection. Also aiming at improved object detection calibration, the authors of Chapter "Uncertainty Quantification for Object Detection: Output- and Gradient-Based Approaches" (p. 251) show that output-based and learning-gradient-based uncertainty metrics are uncorrelated. This allows advantageous combination of both paradigms leading to better overall detection accuracy and to better detection uncertainty estimates.

Part II started with safety aspects related to datasets (Chapter "Does Redundancy in AI Perception Systems Help to Test for Super-Human Automated Driving Performance?", p. 81). In this fourth focus area, further *safety-related aspects* are taken up, including validation of networks. To validate environment perception functions, Chapter "A Variational Deep Synthesis Approach for Perception Validation" (p. 359) introduces a novel data generation framework for DNN validation purposes. The framework allows for effectively defining and testing common traffic scenes. The authors of Chapter "The Good and the Bad: Using Neuron Coverage as a DNN Validation Technique" (p. 383) present a useful metric to assess the training regimen and final quality of a DNN. Different forms of neuron coverage are discussed along with their roots in traditional verification and validation (V&V) methods. Finally, in Chapter "Safety Assurance of Machine Learning for Perception Functions" (p. 335), an assurance case is constructed and acceptance criteria for DNN models are proposed. Various use cases are covered.

Tim Fingscheidt
Hanno Gottschalk
Sebastian Houben

Contents

About the Editors

Tim Fingscheidt Chair of Signal Processing and Machine Learning, Institute for Communications Technology, Technische Universität Braunschweig, Braunschweig, Germany

Tim Fingscheidt received the Dipl.-Ing. degree in Electrical Engineering in 1993 and the Ph.D. degree in 1998 from RWTH Aachen University, Germany, both with distinction. He joined AT&T Labs, Florham Park, NJ, USA, for a Postdoc in 1998, and Siemens AG, Information and Communication, Munich, Germany, in 1999, heading a team in digital signal processing. After a stay with Siemens Corporate Technology, Munich, Germany, from 2005 to 2006, he became Full Professor with the Institute for Communications Technology, Technische Universität (TU) Braunschweig, Germany, holding the Chair of "Signal Processing and Machine Learning". His research interests are machine learning methods for robust environment perception, information fusion, signal enhancement, and coding, with applications both in speech technology and computer vision. He is founder of the TU Braunschweig Deep Learning Lab (TUBS.dll), a graduate student research think tank. Many of his projects have been dealing with automotive applications, such as autonomous driving. In recent years, he has been actively involved in the large-scale national research projects AI Validation, AI Delta Learning, and AI Data Tooling, contributing research in robust semantic segmentation, monocular depth estimation, domain adaptation, corner case detection, and learned image coding. He received numerous national and international awards for his publications, among these three CVPR Workshop Best Paper Awards in 2019, 2020, and 2021.

Hanno Gottschalk Professor for Stochastics at the University of Wuppertal, Germany

Hanno Gottschalk received diploma degrees of Physics and Mathematics from the Ruhr University of Bochum in 1995 and 1997, respectively. There he also received his Ph.D. in Mathematical Physics under the supervision of Sergio Albeverio in 1999. As a fellow of the German Academic Exchange Service (DAAD) he thereafter spent a Postdoc year at Università "La Sapienza" in Rome. Returning to the University of Bonn, he led a research project funded by the research council (DFG)

in 2000–2003, before completing his habilitation in 2003. Acquiring a C2 lecturer position at the University of Bonn in 2007, he left academia for a 3-year experience with Siemens Energy, where he held the position of a core competency owner in probabilistic design in the field of gas turbine engineering. Returning to academia in 2011, since then he holds a position as a Professor of Stochastics at the University of Wuppertal. In 2016, he became spokesperson of the Institute of Mathematical Modeling, Analysis and Computational Mathematics (IMACM). In 2018, he founded the Interdisciplinary Center of Machine Learning and Data Analytics (IZMD) at the University of Wuppertal as the spokesperson jointly with Anton Kummert and in 2020 he became the elected spokesperson of the institute. In the field of computer vision, his research is centered around the online detection and reduction of the errors of artificial intelligence.

Sebastian Houben Senior Data Scientist, Project Lead KI Absicherung, Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), St. Augustin, Germany

Sebastian Houben studied Mathematics and Computer Science in Aachen and Hagen working as a scientific staff member at the Institute for Bio and Nanosystems at the Helmholtz Center in Jülich, Germany. He received his Ph.D. degree in 2015 from the Ruhr University of Bochum, Germany, after which he joined the Autonomous Intelligent Systems Group at the University of Bonn as a Postdoctoral Researcher and was appointed Junior Professor at the University of Bochum in 2017. In early 2020, he took over the project lead KI Absicherung within the Fraunhofer Institute for Intelligent Analysis and Information Systems. He has a strong research interest in environment perception for autonomous robots, in particular, in automated vehicles, and the ramifications from today's state-of-the-art data-driven approaches. During his Ph.D. studies he has been involved in multiple pre-development projects for omnidirectional cameras deployed in advanced driver assistance systems. He participated in several flight robotics challenges, such as the European Robotics Challenge and the Mohamed Bin Zayed International Robotics Challenge, which he, as part of the University Bonn team NimbRo, won in 2017. As of October 2021, he has been appointed Full Professor with specialization in Autonomous Systems and Machine Learning at the University of Applied Sciences Bonn-Rhein-Sieg.

Safe AI—An Overview

Inspect, Understand, Overcome: A Survey of Practical Methods for AI Safety

Sebastian Houben, Stephanie Abrecht, Maram Akila, Andreas Bär, Felix Brockherde, Patrick Feifel, Tim Fingscheidt, Sujan Sai Gannamaneni, Seyed Eghbal Ghobadi, Ahmed Hammam, Anselm Haselhoff, Felix Hauser, Christian Heinzemann, Marco Hoffmann, Nikhil Kapoor, Falk Kappel, Marvin Klingner, Jan Kronenberger, Fabian Küppers, Jonas Löhdefink, Michael Mlynarski, Michael Mock, Firas Mualla, Svetlana Pavlitskaya, Maximilian Poretschkin, Alexander Pohl, Varun Ravi-Kumar, Julia Rosenzweig, Matthias Rottmann, Stefan Rüping, Timo Sämann, Jan David Schneider, Elena Schulz, Gesina Schwalbe, Joachim Sicking, Toshika Srivastava, Serin Varghese, Michael Weber, Sebastian Wirkert, Tim Wirtz, and Matthias Woehrle

S. Houben (✉) · M. Akila · S. S. Gannamaneni · M. Mock · M. Poretschkin · J. Rosenzweig · S. Rüping · E. Schulz · J. Sicking · T. Wirtz
Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, Sankt Augustin, Germany
e-mail: sebastian.houben@h-brs.de

M. Akila
e-mail: maram.akila@iais.fraunhofer.de

S. S. Gannamaneni
e-mail: sujan.sai.gannamaneni@iais.fraunhofer.de

M. Mock
e-mail: michael.mock@iais.fraunhofer.de

M. Poretschkin
e-mail: maximilian.poretschkin@iais.fraunhofer.de

J. Rosenzweig
e-mail: julia.rosenzweig@iais.fraunhofer.de

S. Rüping
e-mail: stefan.rueping@iais.fraunhofer.de

E. Schulz
e-mail: elena.schulz@iais.fraunhofer.de

J. Sicking
e-mail: joachim.sicking@iais.fraunhofer.de

T. Wirtz
e-mail: tim.wirtz@iais.fraunhofer.de

S. Abrecht · C. Heinzemann · M. Woehrle
Robert Bosch GmbH, Gerlingen, Germany
e-mail: stephanie.abrecht@de.bosch.com

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_1

Abstract Deployment of modern data-driven machine learning methods, most often realized by deep neural networks (DNNs), in safety-critical applications such as health care, industrial plant control, or autonomous driving is highly challenging due to numerous model-inherent shortcomings. These shortcomings are diverse and range from a lack of generalization over insufficient interpretability and implausible predictions to directed attacks by means of malicious inputs. Cyber-physical systems employing DNNs are therefore likely to suffer from so-called safety concerns, properties that preclude their deployment as no argument or experimental setup can help to assess the remaining risk. In recent years, an abundance of state-of-the-art techniques aiming to address these safety concerns has emerged. This chapter provides a structured and broad overview of them. We first identify categories of insufficiencies to then describe research activities aiming at their detection, quantification, or mitigation. Our work addresses machine learning experts and safety engineers alike: The former ones might profit from the broad range of machine learning topics covered and discussions on limitations of recent methods. The latter ones might gain insights into the specifics of modern machine learning methods. We hope that this contribution fuels discussions on desiderata for machine learning systems and strategies on how to help to advance existing approaches accordingly.

C. Heinzemann
e-mail: christian.heinzemann@de.bosch.com

M. Woehrle
e-mail: matthias.woehrle@de.bosch.com

G. Schwalbe
Continental AG, Hanover, Germany
e-mail: gesina.schwalbe@continental.com

V. Ravi-Kumar · T. Sämann
Valeo S.A., Paris, France
e-mail: varun-ravi.kumar@valeo.com

T. Sämann
e-mail: timo.saemann@valeo.com

M. Rottmann
Interdisciplinary Center for Machine Learning and Data Analytics, University of Wuppertal, Wuppertal, Germany
e-mail: rottmann@uni-wuppertal.de

S. Wirkert
Bayerische Motorenwerke AG, Munich, Germany
e-mail: sebastian.wa.wirkert@bmwgroup.com

N. Kapoor · J. D. Schneider · S. Varghese
Volkswagen AG, Wolfsburg, Germany
e-mail: nikhil.kapoor@volkswagen.de

J. D. Schneider
e-mail: jan.david.schneider@volkswagen.de

S. Varghese
e-mail: john.serin.varghese@volkswagen.de

1 Introduction

In barely a decade, deep neural networks (DNNs) have revolutionized the field of machine learning by reaching unprecedented, sometimes superhuman, performances on a growing variety of tasks. Many of these neural models have found their way into consumer applications like smart speakers, machine translation engines, or content feeds. However, in safety-critical systems, where human life might be at risk, the use of recent DNNs is challenging as various model-immanent insufficiencies are yet difficult to address.

This work summarizes the promising lines of research in how to identify, address, and at least partly mitigate these DNN insufficiencies. While some of the reviewed works are theoretically grounded and foster the overall understanding of training and predictive power of DNNs, others provide practical tools to adapt their development, training, or predictions. We refer to any such method as a *safety mechanism* if it addresses one or several safety concerns in a feasible manner. Their effectiveness in mitigating safety concerns is assessed by *safety metrics* [CNH+18, OOAG19, BGS+19, SS20a]. As most safety mechanisms target only a particular

P. Feifel · S. E. Ghobadi · A. Hammam
Opel Automobile GmbH, Rüsselsheim, Germany
e-mail: patrick.feifel@opel-vauxhall.com

S. E. Ghobadi
e-mail: seyed.eghbal.ghobadi@opel-vauxhall.com

A. Hammam
e-mail: ahmedmostafa.hammam@opel-vauxhall.com

A. Haselhoff · J. Kronenberger · F. Küppers
Institute of Computer Science, Hochschule Ruhr West, Mülheim, Germany
e-mail: anselm.haselhoff@hs-ruhrwest.de

J. Kronenberger
e-mail: jan.kronenberger@hs-ruhrwest.de

F. Küppers
e-mail: fabian.kueppers@hs-ruhrwest.de

F. Brockherde
umlaut AG, Aachen, Germany
e-mail: felix.brockherde@umlaut.com

F. Hauser
Karlsruhe Institute of Technology, Karlsruhe, Germany
e-mail: felix.hauser@kit.edu

T. Srivastava
Audi AG, Ingolstadt, Germany
e-mail: toshika.srivastava@audi.de

F. Kappel · F. Mualla
ZF Friedrichshafen AG, Friedrichshafen, Germany
e-mail: falk.kappel@zf.com

insufficiency, we conclude that a *holistic safety argumentation* [BGS+19, SSH20, SS20a, WSRA20] for complex DNN-based systems will in many cases rely on a variety of safety mechanisms.

We structure our review of these mechanisms as follows: Sect. 2 focuses on *dataset optimization* for network training and evaluation. It is motivated by the well-known fact that, in comparison to humans, DNNs perform poorly on data that is structurally different from training data. Apart from insufficient generalization capabilities of these models, the data acquisition process and distributional data shifts over time play vital roles. We survey potential counter-measures, e.g., augmentation strategies and outlier detection techniques.

Mechanisms that improve on *robustness* are described in Sects. 3 and 4, respectively. They deserve attention as DNNs are generally not resilient to common perturbations and adversarial attacks.

Section 5 addresses incomprehensible network behavior and reviews mechanisms that aim at *explainability*, e.g., a more transparent functioning of DNNs. This is particularly important from a safety perspective as interpretability might allow for tracing back model failure cases thus facilitating purposeful improvements.

Moreover, DNNs tend to overestimate their prediction confidence, especially on unseen data. Straightforward ways to estimate prediction confidence yield mostly

F. Mualla
e-mail: firas.mualla@zf.com

S. Pavlitskaya · M. Weber
FZI Research Center for Information Technology, Karlsruhe, Germany
e-mail: pavlitskaya@fzi.de

M. Weber
e-mail: weber@fzi.de

A. Bär · T. Fingscheidt · M. Klingner · J. Löhdefink
Institute for Communcations Technology (IfN), Technische Universität Braunschweig, Braunschweig, Germany
e-mail: andreas.baer@tu-bs.de

T. Fingscheidt
e-mail: t.fingscheidt@tu-bs.de

M. Klingner
e-mail: m.klingner@tu-bs.de

J. Löhdefink
e-mail: j.loehdefink@tu-bs.de

M. Hoffmann · M. Mlynarski · A. Pohl
QualityMinds GmbH, München, Germany
e-mail: marco.hoffmann@qualityminds.de

M. Mlynarski
e-mail: michael.mlynarski@qualityminds.de

A. Pohl
e-mail: alexander.pohl@qualityminds.de

unsatisfying results. Among others, this observation fueled research on more sophisticated *uncertainty estimations* (see Sect. 6), *redundancy mechanisms* (see Sect. 7), and attempts to reach *formal verification* as addressed in Sect. 8.

At last, many safety-critical applications require not only accurate but also near real-time decisions. This is covered by mechanisms on the DNN *architectural level* (see Sect. 9) and furthermore by *compression* and *quantization* methods (see Sect. 10).

We conclude this review of mechanism categories with an outlook on the steps to transfer a carefully arranged combination of safety mechanisms into an actual holistic safety argumentation.

2 Dataset Optimization

The performance of a trained model inherently relies on the nature of the underlying dataset. For instance, a dataset with poor variability will hardly result in a model ready for real-world applications. In order to approach the latter, data selection processes, such as corner case selection and active learning, are of utmost importance. These approaches can help to design datasets that contain the most important information, while preventing the so-much-desired information from getting lost in an ocean of data. For a given dataset and active learning setup, data augmentation techniques are very common aiming at extracting as much model performance out of the dataset as possible.

On the other hand, safety arguments also require the analysis of how a model behaves on out-of-distribution data, data that contains concepts the model has not encountered during training. This is quite likely to happen as our world is under constant change, in other words, exposed to a constantly growing domain shift. Therefore, these fields are lately gaining interest, also with respect to perception in automated driving.

In this section, we provide an overview over anomaly and outlier detection, active learning, domain shift, augmentation, and corner case detection. The highly relevant problem of obtaining statistically sufficient test data, even under the assumption of redundant and independent systems, will then be discussed in Chapter "Does Redundancy in AI Perception Systems Help to Test for Super-Human Automated Driving Performance?" [GRS22]. There it will be shown that neural networks trained on the same computer vision task show high correlation in error cases, even if training data and other design choices are kept independent. Using different sensor modalities, however, diminishes the problem to some extent.

2.1 Outlier/Anomaly Detection

The terms anomaly, outlier, and *out-of-distribution* (OoD) data detection are often used interchangeably in literature and refer to task of identifying data samples that are not representative of training data distribution. Uncertainty evaluation (cf. Sect. 6) is closely tied to this field as self-evaluation of models is one of the active areas of research for OoD detection. In particular, for image classification problems it has been reported that neural networks often produce high confidence predictions on OoD data [NYC15, HG17]. The detection of such OoD inputs can either be tackled by post-processing techniques that adjust the estimated confidence [LLS18, DT18] or by enforcing low confidence on OoD samples during training [HAB19, HMD19]. Even guarantees that neural networks produce low confidence predictions for OoD samples can be provided under specific assumptions (cf. [MH20b]). More precisely, this work utilizes Gaussian mixture models that, however, may suffer from high-dimensional data and require strong assumptions on the distribution parameters. Some approaches use generative models like *GANs* [SSW+17, AAAB18] and *autoencoders* [ZP17] for outlier detection. The models are trained to learn in-distribution data manifolds and will produce higher reconstruction loss for outliers.

For OoD detection in semantic segmentation, only a few works have been presented so far. Angus et al. [ACS19] present a comparative study of common OoD detection methods, which mostly deal with image-level classification. In addition, they provide a novel setup of relevant OoD datasets for this task. Another work trains a fully convolutional binary classifier that distinguishes image patches from a known set of classes from image patches stemming from an unknown class [BKOŠ18]. The classifier output applied at every pixel will give the per-pixel confidence value for an OoD object. Both of these works perform at pixel level and without any sophisticated feature generation methods specifically tailored for the detection of entire OoD instances. Up to now, outlier detection has not been studied extensively for object detection tasks. In [GBA+19], two CNNs are used to perform object detection and binary classification (benign or anomaly) in a sequential fashion, where the second CNN takes the localized object within the image as input. From a safety standpoint, detecting outliers or OoD samples is extremely important and beneficial as training data cannot realistically be large enough to capture all situations. Research in this area is heavily entwined with progress in uncertainty estimation (cf. Sect. 6) and domain adaptation (cf. Sect. 2.3). Extending research works to segmentation and object detection tasks would be particularly significant for leveraging automated driving research. In addition to safety, OoD detection can be beneficial in other aspects, e.g., when using local expert models. Here, an expert model for segmentation of urban driving scenes and another expert model for segmentation of highway driving scenes can be deployed in parallel, where an additional OoD detector could act as a trigger on which models can be switched. We extend this discussion in Chapter "Detecting and Learning the Unknown in Semantic Segmentation" [CURG22], where we investigate the handling of unknown objects in semantic segmentation. In the scope of semantic

segmentation, we detect anomalous objects via high-entropy responses and perform a statistical analysis over these detections to suggest new semantic categories.

With respect to the approaches presented above, uncertainty-based and generative-model-based OoD detection methods are currently promising directions of research. However, it remains an open question whether they can unfold their potential well on segmentation and object detection tasks.

2.2 Active Learning

It is widely known that, as a rule of thumb, for the training of any kind of artificial neural network, an increase of training data leads to increased performance. Obtaining labeled training data, however, is often very costly and time-consuming. *Active learning* provides one possible remedy to this problem: Instead of labeling every data point, active learning utilizes a *query strategy* to request labels from a teacher/an oracle which leverage the model performance most. The survey paper by Settles [Set10] provides a broad overview regarding query strategies for active learning methods. However, except for *uncertainty sampling* and *query by committee*, most of them seem to be infeasible in deep learning applications up to now. Hence, most of the research activities in active deep learning focus on these two query strategies, as we outline in the following.

It has been shown for image classification [GIG17, RKG18] that labels corresponding to uncertain samples can leverage the network's performance significantly and that a combination with semi-supervised learning is promising. In both works, uncertainty of unlabeled samples is estimated via Monte Carlo (MC) dropout inference. MC dropout inference and a chosen number of training epochs are executed alternatingly, after performing MC dropout inference, the unlabeled samples' uncertainties are assessed by means of sample-wise dispersion measures. Samples for which the DNN model is very uncertain about its prediction are presented to an oracle and labeled.

With respect to object detection, a moderate number of active learning methods has been introduced [KLSL18, RUN18, DCG+19, BKD19]. These approaches include uncertainty sampling [KLSL18, BKD19] and query-by-committee methods [RUN18]. In [KLSL18, DCG+19], additional algorithmic features specifically tailored for object detection networks are presented, i.e., separate treatment of the localization and classification loss [KLSL18], as well as weak and strong supervision schemes [DCG+19]. For semantic segmentation, an uncertainty-sampling-based approach has been presented [MLG+18], which queries polygon masks for image sections of a fixed size (128×128). Queries are performed by means of accumulated entropy in combination with a cost estimation for each candidate image section. Recently, new methods for estimating the quality of a prediction [DT18, RCH+20] as well as new uncertainty quantification approaches, e.g., gradient-based ones [ORG18], have been proposed. It remains an open question whether they are suitable for active learning. Since most of the conducted studies are rather of academic

nature, also their applicability to real-life data acquisition is not yet demonstrated sufficiently. In particular, it is not clear whether the proposed active learning schemes, including the label acquisition, for instance, in semantic segmentation, is suitable to be performed by human labelers. Therefore, labeling acquisition with a common understanding of the labelers' convenience and suitability for active learning are a promising direction for research and development.

2.3 Domains

The classical assumption in machine learning is that the training and testing datasets are drawn from the same distribution, implying that the model is deployed under the same conditions as it was trained under. However, as it is mentioned in [MTRA+12, JDCR12], for real-world applications this assumption is often violated in the sense that the training and the testing set stems from different domains having different distributions. This poses difficulties for statistical models and the performance will mostly degrade when they are deployed on a domain $\mathcal{D}^{\text{test}}$, having a different distribution than the training dataset (i.e., generalizing from the training to the testing domain is not possible). This makes the study of domains not only relevant from the machine learning perspective, but also from a safety point of view.

More formally, there are differing notions of a "domain" in literature. For [Csu17, MD18], a domain $\mathcal{D} = \{\mathcal{X}, P(\mathbf{x})\}$ consists of a feature space $\mathcal{X} \subset \mathbb{R}^d$ together with a marginal probability distribution $P(\mathbf{x})$ with $\mathbf{x} \in \mathcal{X}$. In [BCK+07, BDBC+10], a domain is a pair consisting of a distribution over the inputs together with a labeling function. However, instead of a sharp labeling function, it is also widely accepted to define a (training) domain $\mathcal{D}^{\text{train}} = \{(\mathbf{x}_i, \overline{y}_i)\}_{i=1}^n$ to consist of n (labeled) samples that are sampled from a joint distribution $P(\mathbf{x}, \overline{y})$ (cf. [LCWJ18]).

The reasons for *distributional shift* are diverse—as are the names to indicate a shift. For example, if the rate of (class) images of interest is different between training and testing set this can lead to a domain gap and, result in differing overall error rates. Moreover, as Chen et al. [CLS+18] mention, changing weather conditions and camera setups in cars lead to a domain mismatch in applications of autonomous driving. In biomedical image analysis, different imaging protocols and diverse anatomical structures can hinder generalization of trained models (cf. [KBL+17, DCO+19]). Common terms to indicate distributional shift are *domain shift, dataset shift, covariate shift, concept drift, domain divergence, data fracture, changing environments*, or *dataset bias*. References [Sto08, MTRA+12] provide an overview. Methods and measures to overcome the problem of domain mismatch between one or more (cf. [ZZW+18]) source domains and target domain(s) and the resulting poor model performance are studied in the field of transfer learning and, in particular, its subtopic domain adaptation (cf. [MD18]). For instance, adapting a model that is trained on synthetically generated data to work on real data is one of the core challenges, as can be seen [CLS+18, LZG+19, VJB+19]. Furthermore, detecting when samples are out-of-domain or out-of-distribution is an active field of research (cf. [LLLS18]

and Sect. 2.1 as well as Sect. 8.2 for further reference). This is particularly relevant for machine learning models that operate in the real world: if an automated vehicle encounters some situation that deviates strongly from what was seen during training (e.g., due to some special event like a biking competition, carnival, etc.) this can lead to wrong predictions and thereby potential safety issues if not detected in time.

In Chapter "Analysis and Comparison of Datasets by Leveraging Data Distributions in Latent Spaces" [SRL+22], a new technique to automatically assess the discrepancy of the domains of different datasets is proposed, including domain shift within one target application. The aptitude of encodings generated by different machine-learned models on a variety of automotive datasets is considered. In particular, loss variants of the variational autoencoder that enforce disentangled latent space representations yield promising results in this respect.

2.4 Augmentation

Given the need for big amounts of data to train neural networks, one often runs into a situation where data is lacking. This can lead to insufficient generalization and an overfitting to the training data. An overview over different techniques to tackle this challenge can be found in [KGC17]. One approach to try and overcome this issue is the augmentation of data. It aims at optimizing available data and increasing its amount, curating a dataset that represents a wide variety of possible inputs during deployment. Augmentation can as well be of help when having to work with a heavily unbalanced dataset by creating more samples of underrepresented classes. A broad survey on data augmentation is provided by [SK19]. They distinguish between two general approaches to data augmentation with the first one being data warping augmentations that focus on taking existing data and transforming it in a way that does not affect labels. The other options are oversampling augmentations, which create synthetic data that can be used to increase the size of the dataset.

Examples of some of the most basic augmentations are flipping, cropping, rotating, translating, shearing, and zooming. These are affecting the geometric properties of the image and are easily implemented [SK19]. The machine learning toolkit `Keras`, for example, provides an easy way of applying them to data using their `ImageDataGenerator` class [C+15]. Other simple methods include adaptations in color space that affect properties such as lighting, contrast, and tints, which are common variations within image data. Filters can be used to control increased blur or sharpness [SK19]. In [ZZK+20], random erasing is introduced as a method with similar effect as cropping, aiming at gaining robustness against occlusions. An example for mixing images together as an augmentation technique can be found in [Ino18].

The abovementioned methods have in common that they work on the input data but there are different approaches that make use of deep learning for augmentation. An example for making augmentations in feature space using autoencoders can be found in [DT17]. They use the representation generated by the encoder and generate new samples by interpolation and extrapolation between existing samples of a

class. The lack of interpretability of augmentations in feature space in combination with the tendency to perform worse than augmentations in image space presents open challenges for those types of augmentations [WGSM17, SK19]. Adversarial training is another method that can be used for augmentation. The goal of adversarial training is to discover cases that would lead to wrong predictions. That means the augmented images won't necessarily represent samples that could occur during deployment but that can help in achieving more robust decision boundaries [SK19]. An example of such an approach can be found in [LCPB18]. Generative modeling can be used to generate synthetic samples that enlarge the dataset in a useful way with GANs, variational autoencoders and the combination of both are important tools in this area [SK19]. Examples for data augmentation in medical context using a CycleGAN [ZPIE17] can be found in [SYPS19] and using a progressively growing GAN [KALL18] in [BCG+18]. Next to neural style transfer [GEB15] that can be used to change the style of an image to a target style, AutoAugment [CZM+19] and population-based augmentation [HLS+19] are two more interesting publications. In both, the idea is to search a predefined search space of augmentations to gather the best selection.

The field of augmenting datasets with purely synthetic images, including related work, is addressed in Chapter "Optimized Data Synthesis for DNN Training and Validation by Sensor Artifact Simulation" [HG22], where a novel approach to apply realistic sensor artifacts to given synthetic data is proposed. The better overall quality is demonstrated via established per-image metrics and a domain distance measure comparing entire datasets. Exploiting this metric as optimization criterion leads to an increase in performance for the `DeeplabV3+` model as demonstrated on the Cityscapes dataset.

2.5 Corner Case Detection

Ensuring that AI-based applications behave correctly and predictably even in unexpected or rare situations is a major concern that gains importance especially in safety-critical applications such as autonomous driving. In the pursuit of more robust AI corner cases play an important role. The meaning of the term corner case varies in the literature. Some consider mere erroneous or incorrect behavior as corner cases [PCYJ19, TPJR18, ZHML20]. For example, in [BBLFs19] corner cases are referred to as situations in which an object detector fails to detect relevant objects at relevant locations. Others characterize corner cases mainly as rare combinations of input parameter values [KKB19, HDHH20]. This project adopts the first definition: inputs that result in unexpected or incorrect behavior of the AI function are defined as corner cases. Contingent on the hardware, the AI architecture and the training data, the search space of corner cases quickly becomes incomprehensibly large. While manual creation of corner cases (e.g., constructing or re-enacting scenarios) might be more controllable, approaches that scale better and allow for a broader and more systematic search for corner cases require extensive automation.

One approach to automatic corner case detection is based on transforming the input data. The *DeepTest* framework [TPJR18] uses three types of image transformations: linear, affine, and convolutional transformations. In addition to these transformations, metamorphic relations help detect undesirable behaviors of deep learning systems. They allow changing the input while asserting some characteristics of the result [XHM+11]. For example, changing the contrast of input frames should not affect the steering angle of a car [TPJR18]. Input–output pairs that violate those metamorphic relations can be considered as corner cases.

Among other things, the white-box testing framework *DeepXplore* [PCYJ19] applies a method called *gradient ascent* to find corner cases (cf. Sect. 8.1). In the experimental evaluation of the framework, three variants of deep learning architectures were used to classify the same input image. The input image was then changed according to the gradient ascent of an objective function that reflected the difference in the resulting class probabilities of the three model variants. When the changed (now artificial) input resulted in different class label predictions by the model variants, the input was considered as a corner case.

In [BBLFs19], corner cases are detected on video sequences by comparing predicted frames with actual frames. The detector has three components: the first component, semantic segmentation, is used to detect and locate objects in the input frame. As the second component, an image predictor trained on frame sequences predicts the actual frame based on the sequence preceding that frame. An error is determined by comparing the actual with the predicted (i.e., expected) frame, following the idea that only situations that are unexpected for AI-based perception functions may be potentially dangerous and therefore a corner case. Both the segmentation and the prediction error are then fed into the third component of the detector, which determines a corner case score that reflects the extent to which unexpected relevant objects are at relevant locations.

In [HDHH20], a corner case detector based on simulations in a CARLA environment [DRC+17] is presented. In the simulated world, AI agents control the vehicles. During simulations, state information of both the environment and the AI agents are fed into the corner case detector. While the environment provides the real vehicle states, the AI agents provide estimated and perceived state information. Both sources are then compared to detect conflicts (e.g., collisions). These conflicts are recorded for analysis. Several ways of automatically generating and detecting corner cases exist. However, corner case detection is a task with challenges of its own: depending on the operational design domain including its boundaries, the space of possible inputs can be very large. Also, some types of corner cases are specific to the AI architecture, e.g., the network type or the network layout used. Thus, corner case detection has to assume a holistic point of view on both model and input, adding further complexity and reducing transferability of previous insights.

Although it can be argued that rarity does not necessarily characterize corner cases, rare input data might have the potential of challenging the AI functionality (cf. Sect. 2.1). Another research direction could investigate whether structuring the input space in a way suitable for the AI functionality supports the detection of corner cases. Provided that the operational design domain is conceptualized as an ontol-

ogy, ontology-based testing [BMM18] may support automatic detection. A properly adapted generator may specifically select promising combinations of extreme parameter values and, thus, provide valuable input for synthetic test data generation.

3 Robust Training

Recent works [FF15, RSFD16, BRW18, AW19, HD19, ETTS19, BHSFs19] have shown that state-of-the-art deep neural networks (DNNs) performing a wide variety of computer vision tasks, such as image classification [KSH12, HZRS15, MGR+18], object detection [Gir15, RDGF15, HGDG17], and semantic segmentation [CPSA17, ZSR+19, WSC+20, LBS+19], are not robust to small changes in the input.

Robustness of neural networks is an active and open research field that can be considered highly relevant for achieving safety in automated driving. Currently, most of the research is directed toward either improving adversarial robustness [SZS+14] (robustness against carefully designed perturbations that aim at causing misclassifications with high confidence) or improving corruption robustness [HD19] (robustness against commonly occurring augmentations such as weather changes, addition of Gaussian noise, photometric changes, etc.). While adversarial robustness might be more of a security issue than a safety issue, corruption robustness, on the other hand, can be considered highly safety-relevant.

Equipped with these definitions, we broadly term *robust training* here as methods or mechanisms that aim at improving either adversarial or corruption robustness of a DNN, by incorporating modifications into the architecture or into the training mechanism itself.

In this section, we cover three widespread techniques for fostering robustness during model training: hyperparameter optimization, modification of loss, and domain generalization. Additionally, in Chapter "Improved DNN Robustness by Multi-Task Training With an Auxiliary Self-Supervised Task" [KFs22], an approach for robustification via multi-task training is presented. Semantic segmentation is combined with the additional target of depth estimation and is proven to show increased robustness on the Cityscapes and KITTI datasets.

A useful metric to assess the training regimen and final quality of a neural network is presented in Chapter "The Good and the Bad: Using Neuron Coverage as a DNN Validation Technique" [GAHW22]. The use of different forms of neuron coverage is discussed and juxtaposed with pair-wise coverage on a tractable example that is being developed.

3.1 Hyperparameter Optimization

The final performance of a neural network highly depends on the learning process. The process includes the actual optimization and may additionally introduce training methods such as dropout, regularization, or parametrization of a multi-task loss.

These methods adapt their behavior for predefined parameters. Hence, their optimal configuration is a priori unknown. We refer to them as *hyperparameters*. Important hyperparameters comprise, for instance, the initial learning rate, steps for learning-rate reduction, learning-rate decay, momentum, batch size, dropout rate, and number of iterations. Their configuration has to be determined according to the architecture and task of the CNN [FH19]. The search of an optimal hyperparameter configuration is called hyperparameter optimization (HO).

HO is usually described as an optimization problem [FH19]. Thereby, the combined configuration space is defined as $\Lambda = \lambda_1 \times \lambda_2 \times \cdots \times \lambda_N$, according to each domain λ_n. Their individual spaces can be continuous, discrete, categorical, or binary.

Hence, one aims to find an optimal hyperparameter configuration $\boldsymbol{\lambda}^\star$ by minimizing an objective function $\mathcal{O}(\cdot)$, which evaluates a trained model $\mathbf{F}$ having parameters $\boldsymbol{\theta}$ on the validation dataset $\mathcal{D}^{\text{val}}$ with the loss J:

$$\boldsymbol{\lambda}^\star = \underset{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}}{\arg\min}\ \mathcal{O}\left(J, \mathbf{F}, \boldsymbol{\theta}, \mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}\right). \tag{1}$$

This problem statement is widely regarded in traditional machine learning and primarily based on Bayesian optimization (BO) in combination with Gaussian processes. However, a straightforward application to deep neural networks encounters problems due to a *lack of scalability, flexibility*, and *robustness* [FKH18, ZCY+19]. To exploit the benefits of BO, many authors proposed different combinations with other approaches. Hyperband [LJD+18] in combination with BO (BOHB) [FKH18] frames the optimization as "[...] a pure exploration non-stochastic infinite-armed bandit problem [...]". The method of BO for iterative learning (BOIL) [NSO20] iteratively internalizes collected information about the learning curve and the learning algorithm itself. The authors of [WTPFW19] introduce the trace-aware knowledge gradient (taKG) as an acquisition function for BO (BO-taKG) which "leverages both trace information and multiple fidelity controls". Thereby BOIL and BO-taKG achieve state-of-the-art performance regarding CNNs outperforming hyperband.

Other approaches such as the orthogonal array tuning method (OATM) [ZCY+19] or HO by reinforcement learning (Hyp-RL) [JGST19] turn away from the Bayesian approaches and offer new research directions.

Finally, the insight that many authors include kernel sizes and number of kernels and layers in their hyperparameter configuration should be emphasized. More work should be spent on the distinct integration of HO in the performance estimation strategy of neural architecture search (cf. Sect. 9.3).

3.2 Modification of Loss

There exist many approaches that aim at directly modifying the loss function with an objective of improving either adversarial or corruption robustness [PDZ18, KS18, LYZZ19, XCKW19, SLC19, WSC+20, SR20]. One of the earliest approaches for improving corruption robustness was introduced by Zheng et al. [ZSLG16] and is called *stability training*, where they introduce a regularization term that penalizes the network prediction to a clean and an augmented image. However, their approach does not scale to many augmentations at the same time. Janocha et al. [JC17] then introduced a detailed analysis on the influence of multiple loss functions to model performance as well as robustness and suggested that expectation-based losses tend to work better with noisy data and squared-hinge losses tend to work better for clean data. Other well-known approaches are mainly based on variations of data augmentation [CZM+19, CZSL19, ZCG+19, LYP+19], which can be computationally quite expensive.

In contrast to corruption robustness, there exist many more approaches based on adversarial examples. We highlight some of the most interesting and relevant ones here. Mustafa et al. [CWL+19] propose to add a loss term that maximally separates class-wise feature map representations, hence increasing the distance from data points to the corresponding decision boundaries. Similarly, Pang et al. [PXD+20] proposed the Max-Mahalanobis center (MMC) loss to learn more structured representations and induce high-density regions in the feature space. Chen et al. [CBLR18] proposed a variation of the well-known cross-entropy (CE) loss that not only maximizes the model probabilities of the correct class, but, in addition, also minimizes model probabilities of incorrect classes. Cisse et al. [CBG+17] constraint the Lipschitz constant of different layers to be less than one which restricts the error propagation introduced by adversarial perturbations to a DNN. Dezfooli et al. [MDFUF19] proposed to minimize the curvature of the loss surface locally around data points. They emphasize that there exists a strong correlation between locally small curvature and correspondingly high adversarial robustness.

All of these methods highlighted above are evaluated mostly for image classification tasks on smaller datasets, namely, CIFAR-10 [Kri09], CIFAR-100 [Kri09], SVHN [NWC+11], and only sometimes on ImageNet [KSH12]. Very few approaches have been tested rigorously on complex safety-relevant tasks, such as *object detection*, *semantic segmentation*, etc. Moreover, methods that improve adversarial robustness are only tested on a small subset of attack types under differing attack specifications. This makes comparing multiple methods difficult.

In addition, methods that improve corruption robustness are evaluated over a standard dataset of various corruption types which may or may not be relevant to its application domain. In order to assess multiple methods for their effect on safety-related aspects, a thorough robustness evaluation methodology is needed, which is largely missing in the current literature. This evaluation would need to take into account relevant disturbances/corruption types present in the real world (application domain) and have to assess robustness toward such changes in a rigorous manner.

Without such an evaluation, we run into the risk of being overconfident in our network, thereby harming safety.

3.3 Domain Generalization

Domain generalization (DG) can be seen as an extreme case of *domain adaptation* (DA). The latter is a type of transfer learning, where the source and target tasks are the same (e.g., shared class labels) but the source and target domains are different (e.g., another image acquisition protocol or a different background) [Csu17, WYKN20]. The DA can be either supervised (SDA), where there is little available labeled data in the target domain, or unsupervised (UDA), where data in the target domain is not labeled. The DG goes one step further by assuming that the target domain is entirely unknown. Thus, it seeks to solve the train-test domain shift in general. While DA is already an established line of research in the machine learning community, DG is relatively new [MBS13], though with an extensive list of papers in the last few years.

Probably, the first intuitive solution that one may think of to implement DG is neutralizing the domain-specific features. It was shown in [WHLX19] that the gray-level co-occurrence matrices (GLCM) tend to perform poorly in semantic classification (e.g., digit recognition) but yield good accuracy in textural classification compared to other feature sets, such as speeded up robust features (SURF) and local binary patterns (LBP). DG was thus implemented by decorrelating the model's decision from the GLCM features of the input image even without the need of domain labels.

Besides the aforementioned intensity-based statistics of an input image, it is known that characterizing image style can be done based on the correlations between the filter responses of a DNN layer [GEB16] (neural style transfer). In [SMK20], the training images are enriched with stylized versions, where a style is defined either by an external style (e.g., cartoon or art) or by an image from another domain. Here, DG is addressed as a *data augmentation* problem.

Some approaches [LTG+18, MH20a] try to learn generalizable latent representations by a kind of adversarial training. This is done by a generator or an encoder, which is trained to generate a hidden feature space that maximizes the error of a domain discriminator but at the same time minimizes the classification error of the task of concern. Another variant of adversarial training can be seen in [LPWK18], where an adversarial autoencoder [MSJ+16] is trained to generate features, which a discriminator cannot distinguish from random samples drawn from a prior Laplace distribution. This regularization prevents the hidden space from overfitting to the source domains, in a similar spirit to how variational autoencoders do not leave gaps in the latent space. In [MH20a], it is argued that the domain labels needed in such approaches are not always well-defined or easily available. Therefore, they assume unknown latent domains which are learned by clustering in a space similar to the style-transfer features mentioned above. The pseudo-labels resulting from clustering are then used in the adversarial training.

Autoencoders have been employed for DG not only in an adversarial setup, but also in the sense of *multi-task learning* nets [Car97], where the classification task in such nets is replaced by a reconstruction one. In [GKZB15], an autoencoder is trained to reconstruct not only the input image but also the corresponding images in the other domains.

In the core of both DA and DG, we are confronted with a distribution matching problem. However, estimating the probability density in high-dimensional spaces is intractable. The density-based metrics, such as Kullback–Leibler divergence, are thus not directly applicable. In statistics, the so-called *two-sample tests* are usually employed to measure the distance between two distributions in a point-wise manner, i.e., without density estimation. For deep learning applications, these metrics need not only to be point-wise but also differentiable. The two-sample tests were approached in the machine learning literature using (differentiable) K-NNs [DK17], classifier two-sample tests (C2ST) [LO17], or based on the theory of kernel methods [SGSS07]. More specifically, the *maximum mean discrepancy* (MMD) [GBR+06, GBR+12], which belongs to the kernel methods, is widely used for DA [GKZ14, LZWJ17, YDL+17, YLW+20] but also for DG [LPWK18]. Using the MMD, the distance between two samples is estimated based on pair-wise kernel evaluations, e.g., the radial basis function (RBF) kernel.

While the DG approaches generalize to domains from which zero shots are available, the so-called *zero-shot learning* (ZSL) approaches generalize to tasks (e.g., new classes in the same source domains) for which zero shots are available. Typically, the input in ZSL is mapped to a semantic vector per class instead of a simple class label. This can be, for instance, a vector of visual attributes [LNH14] or a word embedding of the class name [KXG17]. A task (with zero shots at training time) can be then described by a vector in this space. In [MARC20], there is an attempt to combine ZSL and DG in the same framework in order to generalize to new domains as well as new tasks, which is also referred to as *heterogeneous domain generalization*.

Note that most discussed approaches for DG require non-standard handling, i.e., modifications to models, data, and/or the optimization procedure. This issue poses a serious challenge as it limits the practical applicability of these approaches. There is a line of research which tries to address this point by linking DG to other machine learning paradigms, especially the model-agnostic meta-learning (MAML) [FAL17] algorithm, in an attempt to apply DG in a model-agnostic way. Loosely speaking, a model can be exposed to simulated train-test domain shift by training on a small *support set* to minimize the classification error on a small *validation set*. This can be seen as an instance of a *few-shot learning* (FSL) problem [WYKN20]. Moreover, the procedure can be repeated on other (but related) FSL tasks (e.g., different classes) in what is known as *episodic training*. The model transfers its knowledge from one task to another task and learns how to learn fast for new tasks. Thus, this can be seen as a *meta-learning* objective [HAMS20] (in a FSL setup). Since the goal of DG is to adapt to new domains rather than new tasks, several model-agnostic approaches [LYSH18, BSC18, LZY+19, DdCKG19] try to recast this procedure in a DG setup.

4 Adversarial Attacks

Over the last few years, deep neural networks (DNNs) consistently showed state-of-the-art performance across several vision-related tasks. While their superior performance on clean data is indisputable, they show a lack of robustness to certain input patterns, denoted as *adversarial examples* [SZS+14]. In general, an algorithm for creating adversarial examples is referred to as an *adversarial attack* and aims at fooling an underlying DNN, such that the output changes in a desired and malicious way. This can be carried out without any knowledge about the DNN to be attacked (black-box attack) [MDFF16, PMG+17], or with full knowledge about the parameters, architecture, or even training data of the respective DNN (white-box attack) [GSS15, CW17a, MMS+18]. While initially being applied on simple classification tasks, some approaches aim at finding more realistic attacks [TVRG19, JLS+20], which particularly pose a threat to safety-critical applications, such as DNN-based environment perception systems in autonomous vehicles. Altogether, this motivated the research in finding ways of defending against such adversarial attacks [GSS15, GRCvdM18, MDFUF19, XZZ+19]. In this section, we introduce the current state of research regarding adversarial attacks in general, more realistic adversarial attacks closely related to the task of environment perception for autonomous driving, and strategies for detecting or defending adversarial attacks. We conclude each section by clarifying current challenges and research directions.

4.1 Adversarial Attacks and Defenses

The term *adversarial example* was first introduced by Szegedy et al. [SZS+14]. From there on, many researchers tried to find new ways of crafting adversarial examples more effectively. Here, the fast gradient sign method (FGSM) [GSS15], DeepFool [MDFF16], least-likely class method (LLCM) [KGB17a, KGB17b], C&W [CW17b], momentum iterative fast gradient sign method (MI-FGSM) [DLP+18], and projected gradient descent (PGD) [MMS+18] are a few of the most famous attacks so far. In general, these attacks can be executed in an iterative fashion, where the underlying adversarial perturbation is usually bounded by some norm and is following additional optimization criteria, e.g., minimizing the number of changed pixels.

The mentioned attacks can be further categorized as image-specific attacks, where for each image a new perturbation needs to be computed. On the other hand, image-agnostic attacks aim at finding a perturbation, which is able to fool an underlying DNN on a set of images. Such a perturbation is also referred to as a *universal adversarial perturbation* (UAP). Here, the respective algorithm UAP [MDFFF17], fast feature fool (FFF) [MGB17], and prior driven uncertainty approximation (PD-UA) [LJL+19] are a few honorable mentions. Although the creation process of a universal adversarial perturbation typically relies on a white-box setting, they show a high

transferability across models [HAMFs22]. This allows for black-box attacks, where one model is used to create a universal adversarial perturbation, and another model is being attacked with the beforehand-created perturbation. Universal adversarial perturbations are investigated in more detail in Chapter "Improving Transferability of Generated Universal Adversarial Perturbations for Image Classification and Segmentation" [HAMFs22], where a way to construct these perturbations on the task of image classification and segmentation is presented. In detail, the low-level feature extraction layer is attacked via an additional loss term to generate universal adversarial perturbations more effectively. Another way of designing black-box attacks is to create a surrogate DNN, which mimics the respective DNN to be attacked and thus can be used in the process of adversarial example creation [PMG+17]. On the contrary, some research has been done to create completely incoherent images (based on evolutionary algorithms or gradient ascent) to fool an underlying DNN [NYC15]. Different from that, another line of work has been proposed to alter only some pixels in images to attack a respective model. Here [NK17] has used optimization approaches to perturb some pixels in images to produce targeted attacks, aiming at a specific class output, or non-targeted attacks, aiming at outputting a class different from the network output or the ground truth. This can be extended up to finding one pixel in the image to be exclusively perturbed to generate adversarial images [NK17, SVS19]. The authors of [BF17, SBMC17, PKGB18] proposed to train generative models to generate adversarial examples. Given an input image and the target label, a generative model is trained to produce adversarial examples for DNNs. However, while the produced adversarial examples look rather unrealistic to a human, they are able to completely deceive a DNN.

The existence of adversarial examples not only motivated research in finding new attacks, but also in finding *defense strategies* to effectively defend these attacks. Especially for safety-critical applications, such as DNN-based environment perception for autonomous driving, the existence of adversarial examples needs to be handled accordingly. Similar to adversarial attacks, one can categorize defense strategies into two types: *model-specific* defense strategies and *model-agnostic* defense strategies. The former refers to defense strategies, where the model of interest is modified in certain ways. The modification can be done on the architecture, training procedure, training data, or model weights. On the other hand, model-agnostic defense strategies consider the model to be a black box. Here, only the input or the output is accessible. Some well-known model-specific defense strategies include adversarial training [GSS15, MMS+18], the inclusion of robustness-oriented loss functions during training [CLC+19, MDFUF19, KYL+20], removing adversarial patterns in features by denoising layers [HRF19, MKH+19, XWvdM+19], and redundant teacher-student frameworks [BHSFs19, BKV+20]. Majority of model-agnostic defense strategies primarily focus on various kinds of (gradient masking) pre-processing strategies [GRCvdM18, BFW+19, GR19, JWCF19, LLL+19, RSFM19, TCBZ19]. The idea is to remove the adversary from the respective image, such that the image is transformed from the adversarial space back into the clean space.

Nonetheless, Athalye et al. [ACW18] showed that gradient masking alone is not a sufficient criterion for a reliable defense strategy. In addition, detection and out-of-

distribution techniques have also been proposed as model-agnostic defense strategies against adversarial attacks. Here, the Mahalanobis distance [LLLS18] or area under the receiver operating characteristic curve (AUROC) and area under the precision-recall curve (AUPR) [HG17] are used to detect adversarial examples. The authors of [HG17, LLLS17, MCKBF17], on the other hand, proposed to train networks to detect, whether the input image is out-of-distribution or not.

Moreover, Feinman et al. [FCSG17] proved that adversarial attacks usually produce high uncertainty on the output of the DNN. As a consequence, they proposed to use the dropout technique to estimate uncertainty on the output to identify a possible adversarial attack. Regarding adversarial attacks, majority of the listed attacks are designed for image classification. Only a few adversarial attacks consider tasks that are closely related to autonomous driving, such as bounding box detection, semantic segmentation, instance segmentation, or even panoptic segmentation. Also, the majority of the adversarial attacks rely on a white-box setting, which is usually not present for a potential attacker. Especially universal adversarial perturbations have to be considered as a real threat due to their high model transferability. Generally speaking, the existence of adversarial examples has not been thoroughly studied yet. An analytical interpretation is still missing, but could help in designing more mature defense strategies.

Regarding defense strategies, adversarial training is still considered as one of the most effective ways of increasing the robustness of a DNN. Nonetheless, while adversarial training is indeed effective, it is rather inefficient in terms of training time. In addition, model-agnostic defenses should be favored as once being designed, they can be easily transferred to different models. Moreover, as most model-agnostic defense strategies rely on gradient-masking and it has been shown that gradient-masking is not a sufficient property for a defense strategy, new ways of designing model-agnostic defenses should be taken into account. Furthermore, out-of-distribution and adversarial attacks detection or even correction methods have been a new trend for identifying attacks. However, as the environment perception system of an autonomous driving vehicle could rely on various information sources, including LiDAR, optical flow, or depth from a stereo camera, techniques of information fusion should be further investigated to mitigate or even eliminate the effect of adversarial examples.

4.2 More Realistic Attacks

We consider the following two categories of realistic adversarial attacks: (1) image-level attacks, which not only fool a neural network but also pose a provable threat to autonomous vehicles and (2) attacks which have been applied in a real world or in a simulation environment, such as car learning to act (CARLA) [DRC+17].

Some notable examples in the first category of attacks include attacks on semantic segmentation [MCKBF17] or person detection [TVRG19].

In the second group of approaches, the attacks are specifically designed to survive real-world distortions, including different distances, weather and lighting conditions,

as well as camera angles. For this, adversarial perturbations are usually concentrated in a specific image area, called *adversarial patch*. Crafting an adversarial patch involves specifying a patch region in each training image, applying transformations to the patch, and iteratively changing the pixel values within this region to maximize the network prediction error. The latter step typically relies on an algorithm, proposed for standard adversarial attacks, which aim at crafting invisible perturbations while misleading neural networks, e.g., C&W [CW17b], Jacobian-based saliency map attack (JSMA) [PMG+17], and PGD [MMS+18].

The first printable adversarial patch for image classification was described by Brown et al. [BMR+17]. Expectation over transformations (EOT) [AEIK18] is one of the influential updates to the original algorithm—it permits to robustify patch-based attacks to distortions and affine transformations. Localized and visible adversarial noise (LaVAN) [KZG18] is a further method to generate much smaller patches (up to 2% of the pixels in the image). In general, fooling image classification with a patch is a comparatively simple task, because adversarial noise can mimic an instance of another class and thus lower the prediction probability for a true class.

Recently, patch-based attacks for a more tricky task of object detection have been described [LYL+19, TVRG19]. Also, Lee and Kolter [LK19] generate a patch using PGD [MMS+18], followed by EOT applied to the patch. With this approach, all detections in an image can be successfully suppressed, even without any overlap of a patch with bounding boxes. Furthermore, several approaches for generating an adversarial T-shirt have been proposed, including [XZL+20, WLDG20].

DeepBillboard [ZLZ+20] is the first attempt to attack end-to-end driving models with adversarial patches. The authors propose to generate a single patch for a sequence of input images to mislead four steering models, including DAVE-2 in a drive-by scenario.

Apart from physical feasibility, inconspicuousness is crucial for a realistic attack. Whereas adversarial patches usually look like regions of noise, several works have explored attacks with an inconspicuous patch. In particular, Eykholt et al. [EEF+18] demonstrate the vulnerability of road sign classification to the adversarial perturbations in the form of only black and white stickers. In [BHG+19], an end-to-end driving model is attacked in CARLA by painting of black lines on the road. Also, Kong and Liu [KGLL20] use a generative adversarial network to get a realistic billboard to attack an end-to-end driving model in a drive-by scenario. In [DMW+20], a method to hide visible adversarial perturbations with customized styles is proposed, which leads to adversarial traffic signs that look unsuspicious to a human. Current research mostly focuses on attacking image-based perception of an autonomous vehicle. Adversarial vulnerability of further components of an autonomous vehicle, e.g., LiDAR-based perception, optical flow, and depth estimation, has only recently gained attention. Furthermore, most attacks consider only a single component of an autonomous driving pipeline, the question whether the existing attacks are able to propagate to further pipeline stages has not been studied yet. The first work in this direction [JLS+20] describes an attack on object detection and tracking. The evaluation is, however, limited to a few clips, where no experiments in the real world

have been performed. Overall, the research on realistic adversarial attacks, especially combined with physical tests, is currently in the starting phase.

5 Interpretability

Neural networks are, by their nature, black boxes and therefore intrinsically hard to interpret [Tay06]. Due to their unrivaled performance, they still remain first choice for advanced systems even in many safety-critical areas, such as level 4 automated driving. This is why the research community has invested considerable effort to unhinge the black-box character and make deep neural networks more transparent.

We can observe three strategies that provide different viewpoints toward this goal in the state of the art. First is the most direct approach of opening up the black box and looking at intermediate representations. Being able to interpret individual layers of the system facilitates interpretation of the whole. The second approach tries to provide interpretability by explaining the network's decisions with pixel attributions. Aggregated explanations of decisions can then lead to interpretability of the system itself. Third is the idea of approximating the network with interpretable proxies to benefit from the deep neural networks performance while allowing interpretation via surrogate models. Underlying all aspects here is the area of visual analytics.

There exists earlier research in the medical domain to help human experts understand and convince them of machine learning decisions [CLG+15]. Legal requirements in the finance industry gave rise to interpretable systems that can justify their decisions. An additional driver for interpretability research was the concern for Clever Hans predictors [LWB+19].

5.1 Visual Analytics

Traditional data science has developed a huge tool set of automated analysis processes conducted by computers, which are applied to problems that are well defined in the sense that the dimensionality of input and output as well as the size of the dataset they rely on is manageable. For those problems that in comparison are more complex, the automation of the analysis process is limited and/or might not lead to the desired outcome. This is especially the case with unstructured data like image or video data in which the underlying information cannot directly be expressed by numbers. Rather, it needs to be transformed to some structured form to enable computers to perform some task of analysis. Additionally, with an ever-increasing amount of various types of data being collected, this "information overload" cannot solely be analyzed by automatic methods [KAF+08, KMT09].

Visual analytics addresses this challenge as "the science of analytical reasoning facilitated by interactive visual interfaces" [TC05]. Visual analytics therefore does not only focus on either computationally processing data or visualizing results but

coupling both tightly with interactive techniques. Thus, it enables an integration of the human expert into the iterative visual analytics process: through visual understanding and human reasoning, the knowledge of the human expert can be incorporated to effectively refine the analysis. This is of particular importance, where a stringent safety argumentation for complex models is required. With the help of visual analytics, the line of argumentation can be built upon arguments that are understandable for humans. To include the human analyst efficiently into this process, a possible guideline is the *visual analytics mantra* by Keim: "Analyze first, show the important, zoom, filter and analyze further, details on demand" [KAF+08].[1]

The core concepts of visual analytics therefore rely on well-designed interactive visualizations, which support the analyst in the tasks of, e.g., reviewing, understanding, comparing, and inferring not only the initial phenomenon or data but also the computational model and its results itself with the goal of enhancing the analytical process. Driven by various fields of application, visual analytics is a multidisciplinary field with a wide variety of task-oriented development and research. Recent work has been done in several areas: depending on the task, there exist different pipeline approaches to create whole *visual analytics systems* [WZM+16]; the injection of human expert knowledge into the process of determining trends and patterns from data is the focus of *predictive visual analytics* [LCM+17, LGH+17]; enabling the human to explore *high-dimensional data* [LMW+17] interactively and visually (e.g., via dimensionality reduction [SZS+17]) is a major technique to enhance the understandability of complex models (e.g., neural networks); the iterative improvement and the understanding of machine learning models is addressed by using interactive visualizations in the field of *general machine learning* [LWLZ17], or the other way round: using machine learning to improve visualizations and guidance based on user interactions [ERT+17]. Even more focused on the loop of simultaneously developing and refining machine learning models is the area of *interactive machine learning*, where the topics of *interface design* [DK18] and the *importance of users* [ACKK14, SSZ+17] are discussed. One of the current research directions is using visual analytics in the area of *deep learning* [GTC+18, HKPC18, CL18]. However, due to the interdisciplinarity of visual analytics, there are still open directions and ongoing research opportunities.

Especially in the domain of neural networks and deep learning, visual analytics is a relatively new approach in tackling the challenge of *explainability* and *interpretability* of those often called *black boxes*. To enable the human to better interact with the models, research is done in enhancing the *understandability* of complex deep learning models and their outputs with the use of proper visualizations. Other research directions attempt to achieve improving the *trustability* of the models, giving the opportunity to inspect, diagnose, and refine the model. Further, possible areas for research are *online training processes* and the development of *interactive systems* covering the whole process of training, enhancing, and monitoring machine learning models. Here, the approach of *mixed guidance*, where system-initiated guidance is

[1] Extending the original *visualization mantra* by Shneiderman "Overview first, filter and zoom, details on demand" [Shn96].

combined with user-initiated guidance, is discussed among the visual analytics community as well. Another challenge and open question is creating ways of *comparing models* to examine which model yields a better performance, given specific situations and selecting or combining the best models with the goal of increasing performance and overall safety.

5.2 Intermediate Representations

In general, representation learning [BCV13] aims to extract lower dimensional features in latent space from higher dimensional inputs. These features are then used as an effective representation for regression, classification, object detection, and other machine learning tasks. Preferably, latent features should be disentangled, meaning that they represent separate factors found in the data that are statistically independent. Due to their importance in machine learning, finding meaningful intermediate representations has long been a primary research goal. Disentangled representations can be interpreted more easily by humans and can, for example, be used to explain the reasoning of neural networks [HDR18].

Among the longer known methods for extracting disentangled representations are principal component analysis (PCA) [FP78, JC16], independent component analysis [HO00], and nonnegative matrix factorization [BBL+07]. PCA is highly sensitive to outliers and noise in the data. Therefore, more robust algorithms were proposed. In [SBS12] already a small neural network was used as an encoder and the algorithm proposed in [FXY12] can deal with high-dimensional data. Some robust PCA algorithms are provided with analytical performance guarantees [XCS10, RA17, RL19].

A popular method for representation learning with deep networks is the variational autoencoder (VAE) [KW14]. An important generalization of the method is the β-VAE variant [HMP+17], which improved the disentanglement capability [FAA18]. Later analysis added to the theoretical understanding of β-VAE [BHP+18, SZYP19, KP20]. Compared to standard autoencoders, VAEs map inputs to a distribution, instead of mapping them to a fixed vector. This allows for additional regularization of the training to avoid overfitting and ensure good representations. In β-VAEs, the trade-off between reconstruction quality and disentanglement can be fine-tuned by the hyperparameter β.

Different regularization schemes have been suggested to improve the VAE method. Among them are Wasserstein autoencoders [TBGS19, XW19], attribute regularization [PL20], and relational regularization [XLH+20]. Recently, a connection between VAEs and nonlinear independent component analysis was established [KKMH20] and then expanded [SRK20].

Besides VAEs, deep generative adversarial networks can be used to construct latent features [SLY15, CDH+16, MSJ+16]. Other works suggest centroid encoders [GK20] or conditional learning of Gaussian distributions [SYZ+21] as alternatives to VAEs. In [KWG+18], concept activation vectors are defined as being orthogonal

to the decision boundary of a classifier. Apart from deep learning, entirely new architectures, such as capsule networks [SFH17], might be used to disassemble inputs.

While many different approaches for disentangling exist, the feasibility of the task is not clear yet and a better theoretical understanding is needed. The disentangling performance is hard to quantify, which is only feasible with information about the latent ground truth [EW18]. Models that overly rely on single directions, single neurons in fully connected networks, or single feature maps in CNNs have the tendency to overfit [MBRB18]. According to [LBL+19], unsupervised learning does not produce good disentangling and even small latent spaces do not reduce the sample complexity for simple tasks. This is in direct contrast to newer findings that show a decreased sample complexity for more complex visual downstream tasks [vLSB20]. So far, it is unclear if disentangling improves the performance of machine learning tasks.

In order to be interpretable, latent disentangled representations need to be aligned with human understandable concepts. In [EIS+19], training with adversarial examples was used and the learned representations were shown to be more aligned with human perception. For explainable AI, disentangling alone might not be enough to generate interpretable output, and additional regularization could be needed.

In Chapter "Invertible Neural Networks for Understanding Semantics of Invariances of CNN Representations" [REBO22], a new approach is presented, which aims at extracting invariant representations that are present in a trained network. To this end, invertible neural networks are deployed to recover these invariances and to map them to accessible semantic concepts. These concepts allow for both interpretation of an inner model representation and means to carefully manipulate them and examine the effect on the prediction.

5.3 Pixel Attribution

The non-linearity and complexity of DNNs allow them to solve perception problems, such as detecting a pedestrian, that cannot be specified in detail. At the same time, the automatic extraction of features given in an input image and the mapping to the respective prediction is counterintuitive and incomprehensible for humans, which makes it hard to argue safety for a neural-network-based perception task. Feature importance techniques are currently predominantly used to diagnose the causes of incorrect model behaviors [BXS+20]. So-called *attribution maps* are a visual technique to express the relationship between relevant pixels in the input image and the network's prediction. Regions in an image that contain relevant features are highlighted accordingly. Attribution approaches mostly map to one of three categories.

Gradient-based and activation-based approaches (such as [SVZ14, SDBR14, BBM+15, MLB+17, SCD+20, STK+17, SGK19] among others) rely on the gradient of the prediction with respect to the input. Regions that were most relevant for the prediction are highlighted. Activation-based approaches relate the feature maps of the last convolutional layer to output classes.

Perturbation-based approaches [ZF14, FV17, ZCAW17, HMK+19] suggest manipulating the input. If the prediction changes significantly, the input may hold a possible explanation at least.

While gradient-based approaches are oftentimes faster in computation, perturbation-based approaches are much easier to interpret. As many studies have shown [STY17, AGM+20], there is still a lot of research to be done before attribution methods are able to robustly provide explanations for model predictions, in particular, erroneous behavior. One key difficulty is the lack of an agreed-upon definition of a good attribution map including important properties. Even between humans, it is hard to agree on what a good explanation is due to its subjective nature. This lack of ground truth makes it hard or even impossible to quantitatively evaluate an explanation method. Instead, this evaluation is done only implicitly. One typical way to do this is the axiomatic approach. Here a set of desiderata of an attribution method are defined, on which different attribution methods are then evaluated. Alternatively, different attribution methods may be compared by perturbing the input features starting with the ones deemed most important and measuring the drop in accuracy of the perturbed models. The best method will result into the greatest overall loss in accuracy as the number of inputs are omitted [ACÖG17]. Moreover, for gradient-based methods it is hard to assess if an unexpected attribution is caused by a poorly performing network or a poorly performing attribution method [FV17]. How to cope with negative evidence, i.e., the object was predicted because a contrary clue in the input image was missing, is an open research question. Additionally, most methods were shown on classification tasks until now. It remains to be seen how they can be transferred to object detection and semantic segmentation tasks. In the case of perturbation-based methods, the high computation time and single-image analysis inhibit widespread application.

5.4 Interpretable Proxies

Neural networks are capable of capturing complicated logical (cor)relations. However, this knowledge is encoded on a *sub-symbolic* level in the form of learned weights and biases, meaning that the reasoning behind the processing chain cannot be directly read out or interpreted by humans [CCO98]. To explain the sub-symbolic processing, one can either use attribution methods (cf. Sect. 5.3), or lift this sub-symbolic representation to a *symbolic* one [GBY+18], meaning a more interpretable one. Interpretable proxies or surrogate models try to achieve the latter: the DNN behavior is approximated by a model that uses symbolic knowledge representations. Symbolic representations can be linear models like local interpretable model-agnostic explanations (LIME) [RSG16] (proportionality), decision trees (if-then chains) [GBY+18], or loose sets of logical rules. Logical connectors can simply be AND and OR but also more general ones like at-least-M-of-N [CCO98]. The *expressiveness* of an approach refers to the logic that is used: Boolean-only versus first-order logic, and binary versus fuzzy logic truth values [TAGD98]. Other than attribution methods (cf.

Sect. 5.3), these representations can capture combinations of features and (spatial) relations of objects and attributes. As an example, consider "eyes are closed" as explanation for "person asleep": attribution methods only could mark the location of the eyes, dismissing the relations of the attributes [RSS18]. All mentioned surrogate model types (linear, set of rules) require interpretable input features in order to be interpretable themselves. These features must either be directly obtained from the DNN input or (intermediate) output, or automatically be extracted from the DNN representation. Examples for extraction are the super-pixeling used in LIME for input feature detection, or concept activation vectors [KWG+18] for DNN representation decoding.

Quality criteria and goals for interpretable proxies are [TAGD98]: *accuracy* of the standalone surrogate model on unseen examples, *fidelity* of the approximation by the proxy, *consistency* with respect to different training sessions, and *comprehensibility* measured by the complexity of the rule set (number of rules, number of hierarchical dependencies). The criteria are usually in conflict and need to be balanced: Better accuracy may require a more complex, thus less expressive set of rules.

Approaches for interpretable proxies differ in the validity range of the representations: Some aim for surrogates that are only valid *locally* around specific samples, like in LIME [RSG16] or in [RSS18] via inductive logic programming. Other approaches try to be more *globally* approximate aspects of the model behavior. Another categorization is defined by whether full access (*white box*), some access (*gray box*), or no access (*black box*) to the DNN internals is needed. One can further differentiate between *post hoc* approaches that are applied to a trained model, and approaches that try to integrate or *enforce symbolic representations* during training. Post hoc methods cover the wide field of rule extraction techniques for DNNs. The interested reader may refer to [AK12, Hai16]. Most white- and gray-box methods try to turn the DNN connections into if-then rules that are then simplified, like done in DeepRED [ZLMJ16]. A black-box example is validity interval analysis [Thr95], which refines or generalizes rules on input intervals, either starting from one sample or a general set of rules. Enforcement of symbolic representations can be achieved by enforcing an output structure that provides insights to the decision logic, such as textual explanations or a rich output structure allowing investigation of correlations [XLZ+18]. An older discipline for enforcing symbolic representations is the field of neural-symbolic learning [SSZ19]. The idea is based on a hybrid learning cycle in which a symbolic learner and a DNN iteratively update each other via rule insertion and extraction. The comprehensibility of global surrogate models suffers from the complexity and size of concurrent DNNs. Thus, stronger rule simplification methods are required [Hai16]. The alternative direction of local approximations mostly concentrates on linear models instead of more expressive rules [Thr95, RSS18]. Furthermore, balancing of the quality objectives is hard since available indicators for interpretability may not be ideal. And lastly, applicability is heavily infringed by the requirement of interpretable input features. These are usually not readily available from input (often pixel level) or DNN output. Supervised extraction approaches vary in their fidelity, and unsupervised ones do not guarantee to yield meaningful or interpretable results, respectively, such as the super-pixel clusters of LIME.

6 Uncertainty

Uncertainty refers to the view that a neural network is not conceived as a deterministic function but as a probabilistic function or estimator, delivering a random distribution for each input point. Ideally, the mean value of the distribution should be as close as possible to the ground-truth value of the function being approximated by the neural network and the uncertainty of the neural network refers to its variance when being considered as a random variable, thus allowing to derive a confidence with respect to the mean value. Regarding safety, the variance may lead to estimations about the confidence associated with a specific network output and opens the option for discarding network outputs with insufficient confidence.

There are roughly two broad approaches for training neural networks as probabilistic functions: parametric approaches [KG17] and Bayesian neural networks on the one hand, such as [BCKW15], where the transitions along the network edges are modeled as probability distributions, and ensemble-based approaches on the other hand [LPB17, SOF+19], where multiple networks are trained and considered as samples of a common output distribution. Apart from training as probabilistic function, uncertainty measures have been derived from single, standard neural networks by post-processing on the trained network logits, leading, for example, to calibration measures (cf. e.g., [SOF+19]).

6.1 Generative Models

Generative models belong to the class of unsupervised machine learning models. From a theoretical perspective, these are particularly interesting, because they offer a way to analyze and model the density of data. Given a finite dataset $\mathcal{D}$ independently distributed according to some distribution $p(\mathbf{x})$, $\mathbf{x} \in \mathcal{D}$, generative models aim to estimate or enable sampling from the underlying density $p(\mathbf{x})$ in a model $\mathbf{F}(\mathbf{x}, \boldsymbol{\theta})$. The resulting model can be used for data indexing [Wes04], data retrieval [ML11], for visual recognition [KSH12], speech recognition and generation [HDY+12], language processing [KM02, CE17], and robotics [Thr02]. Following [OE18], we can group generative models into two main classes:

- Cost function-based models, such as autoencoder [KW14, Doe16], deep belief networks [Hin09], and generative adversarial networks [GPAM+14, RMC15, Goo17].
- Energy-based models [LCH+07, SH09], where the joint probability density is modeled by an energy function.

Besides these *deep learning* approaches, generative models have been studied in machine learning in general for quite some time (cf. [Fry77, Wer78, Sil86, JMS96, She04, Sco15, Gra18]). A very prominent example of generative networks is Gaussian process [WR96, Mac98, WB98, Ras03] and their deep learning extensions [DL13, BHLHL+16] as generative models.

An example of a generative model being employed for image segmentation uncertainty estimation is the probabilistic `U-Net` [KRPM+18]. Here a variational autoencoder (VAE) conditioned on the image is trained to model uncertainties. Samples from the VAE are fed into a segmentation `U-Net` which can thus give different results for the same image. This was tested in context of medical images, where inter-rater disagreements lead to uncertain segmentation results and Cityscapes segmentation. For the Cityscapes segmentation, the investigated use case was label ambiguity (e.g., is a BMW X7 a car or a van) using artificially created, controlled ambiguities. Results showed that the probabilistic `U-Net` could reproduce the segmentation ambiguity modes more reliably than competing methods, such as a dropout `U-Net` which is based on techniques elaborated in the next section.

6.2 Monte Carlo Dropout

A widely used technique to estimate model uncertainty is Monte Carlo (MC) dropout [GG16] that offers a Bayesian motivation, conceptual simplicity, and scalability to application-size networks. This combination distinguishes MC dropout from competing Bayesian neural network (BNN) approximations (e.g., [RBB18, BCKW15], see Sect. 6.3). However, these approaches and MC dropout share the same goal: to equip neural networks with a *self-assessment* mechanism that detects unknown input concepts and thus potential model insufficiencies.

On a technical level, MC dropout assumes prior distributions on network activations, usually independent and identically distributed (i.i.d.) Bernoulli distributions. Model training with iteratively drawn Bernoulli samples, the so-called *dropout masks*, then yields a data-conditioned posterior distribution within the chosen parametric family. It is interesting to note that this training scheme was used earlier—independent from an uncertainty context—for better model generalization [SHK+14]. At inference, sampling provides estimates of the input-dependent output distributions. The spread of these distributions is then interpreted as the prediction uncertainty that originates from limited knowledge of model parameters. Borrowing "frequentist" terms, MC dropout can be considered as an implicit network ensemble, i.e., as a set of networks that share (most of) their parameters.

In practice, MC dropout requires only a minor modification of the optimization objective during training and multiple, trivially parallelizable forward passes during inference. The loss modification is largely agnostic to network architecture and does not cause substantial overhead. This is in contrast to the sampling-based inference that increases the computational effort massively—by estimated factors of 20–100 compared to networks without MC dropout. A common practice is therefore the use of *last-layer dropout* [SOF+19] that reduces computational overhead to estimated factors of 2–10. Alternatively, analytical moment propagation allows sampling-free MC dropout inference at the price of additional approximations (e.g., [PFC+19]). Further extensions of MC dropout target the integration of data-inherent (aleatoric)

uncertainty [KG17] and tuned performance by learning layer-specific dropout rate using concrete relaxations [GHK17].

The quality of MC dropout uncertainties is typically evaluated using negative log-likelihood (NLL), expected calibration error (ECE) and its variants (cf. Sect. 6.6) and by considering correlations between uncertainty estimates and model errors (e.g., AUSE [ICG+18]). Moreover, it is common to study how useful uncertainty estimates are for solving auxiliary tasks like out-of-distribution classification [LPB17] or robustness w.r.t. adversarial attacks.

MC dropout is a working horse of safe ML, being used with various networks and for a multitude of applications (e.g., [BFS18]). However, several authors pointed out shortcomings and limitations of the method: MC dropout bears the risk of overconfident false predictions ([Osb16]), offers less diverse uncertainty estimates compared to (the equally simple and scalable) deep ensembles ([LPB17], see Sect. 7.1), and provides only rudimentary approximations of true posteriors.

Relaxing these modeling assumptions and strengthening the Bayesian motivation of MC dropout is therefore an important research avenue. Further directions for future work are the development of *semantic uncertainty mechanisms* (e.g., [KRPM+18]), improved local uncertainty calibrations, and a better understanding of the outlined sampling-free schemes to uncertainty estimation.

6.3 Bayesian Neural Networks

As the name suggests, Bayesian neural networks (BNNs) are inspired by a Bayesian interpretation of probability (for an introduction cf. [Mac03]). In essence, it rests on Bayes' theorem,

$$p(a|b)\,p(b) = p(a,b) = p(b|a)\,p(a) \quad \Rightarrow \quad p(a|b) = \frac{p(b|a)\,p(a)}{p(b)}\,, \tag{2}$$

stating that the conditional probability density function (PDF) $p(a|b)$ for a given b may be expressed in terms of the inverted conditional PDF $p(b|a)$. For machine learning, where one intends to make predictions y for unknown $\mathbf{x}$ given some training data $\mathcal{D}$, this can be reformulated into

$$y = \mathrm{NN}(\mathbf{x}|\boldsymbol{\theta}) \quad \text{with} \quad p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})\,p(\boldsymbol{\theta})}{p(\mathcal{D})}\,. \tag{3}$$

Therein NN denotes a conventional (deep) neural network (DNN) with model parameters $\boldsymbol{\theta}$, e.g., the set of weights and biases. In contrast to a regular DNN, the weights are given in terms of a probability distribution $p(\boldsymbol{\theta}|\mathcal{D})$ turning also the output y of a BNN into a distribution. This allows to study the mean $\mu = \langle y^1 \rangle$ of the DNN for a given $\mathbf{x}$ as well as higher moments of the distribution, typically the resulting variance $\sigma^2 = \langle (y-\mu)^2 \rangle$ is of interest, where

$$\left\langle y^k \right\rangle = \int \mathrm{NN}(\mathbf{x}|\boldsymbol{\theta})^k \, p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} \, . \tag{4}$$

While μ yields the output of the network, the variance σ^2 is a measure for the uncertainty of the model for the prediction at the given point. Central to this approach is the probability of the data given the model, here denoted by $p(\mathcal{D}|\boldsymbol{\theta})$, as it is the key component connecting model and training data. Typically, the prior distribution $p(\mathcal{D})$ is "ignored" as it only appears as a normalization constant within the averages, see (4). In the cases where the data $\mathcal{D}$ is itself a distribution due to inherent uncertainty, i.e., presence of aleatoric risk [KG17], such a concept seems natural. However, Bayesian approaches are also applicable for all other cases. In those, loosely speaking, the likelihood of $\boldsymbol{\theta}$ is determined via the chosen loss function (for the connection between the two concepts cf. [Bis06]).

On this general level, Bayesian approaches are broadly accepted and also find use for many other model classes besides neural networks. However, the loss surfaces of DNNs are known for their high dimensionality and strong non-convexity. Typically, there are abundant parameter combinations $\boldsymbol{\theta}$ that lead to (almost) equally good approximations to the training data $\mathcal{D}$ with respect to a chosen loss. This makes an evaluation of $p(\boldsymbol{\theta}|\mathcal{D})$ for DNNs close to impossible in full generality. At least no (exact) solutions for this case exist at the moment. Finding suitable approximations to the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ is an ongoing challenge for the construction of BNNs. At this point we only summarize two major research directions in the field. One approach is to assume that the distribution factorizes. While the full solution would be a joint distribution implying correlations between different weights (etc.), possibly even across layers, this approximation takes each element of $\boldsymbol{\theta}$ to be independent form the others. Although this is a strong assumption, it is often made, in this case, parameters for the respective distributions of each element can be learned via training (cf. [BCKW15]). The second class of approaches focuses on the region of the loss surface around the minimum chosen for the DNN. As discussed, the loss relates to the likelihood and quantities such as the curvature at the minimum, therefore directly connected to the distribution of $\boldsymbol{\theta}$. Unfortunately, already using this type of quantities requires further approximations [RBB18]. Alternatively, the convergence of the training process may be altered to sample networks close to the minimum [WT11]. While this approach contains information about correlations among the weights, it is usually restricted to a specific minimum. For a non-Bayesian approach taking into account several minima, see deep ensembles in Sect. 7.1. BNNs also touch other concepts such as MC dropout (cf. [GG16] and Sect. 7.1), or prior networks, which are based on a Bayesian interpretation but use conventional DNNs with an additional (learned) σ output [MG18].

6.4 Uncertainty Metrics for DNNs in Frequentist Inference

Classical uncertainty quantification methods in frequentist inference are mostly based on the outputs of statistical models. Their uncertainty is quantified and assessed, for instance, via dispersion measures in classification (such as entropy, probability margin, or variation ratio), or confidence intervals in regression. However, the nature of DNN architectures [RDGF15, CPK+18] and the cutting-edge applications tackled by those (e.g., semantic segmentation, cf. [COR+16]) open the way toward more elaborate uncertainty quantification methods. Besides the mentioned classical approaches, intermediate feature representations within a DNN (cf. [CZYS19, OSM19]) or gradients according to self-affirmation that represent re-learning stress (see [ORG18]) reveal additional information. In addition, in case of semantic segmentation, the geometry of a prediction may give access to further information, cf. [MRG20, RS19, RCH+20]. By the computation of statistics of those quantities as well as low-dimensional representations thereof, we obtain more elaborate uncertainty quantification methods specifically designed for DNNs that can help us to detect misclassifications and out-of-distribution objects (cf. [HG17]). Features gripped during a forward pass of a data point $\mathbf{x}$ through a DNN $\mathbf{F}$ can be considered layer-wise, i.e., $\mathbf{F}_\ell(\mathbf{x})$ after the ℓ-th layer. These can be translated into a handful of quantities per layer [OSM19] or further processed by another DNN that aims at detecting errors [CZYS19]. While, in particular, [OSM19] presents a proof of concept on small-scale classification problems, their applicability to large-scale datasets and problems, such as semantic segmentation and object detection, remains open.

The development for gradient-based uncertainty quantification methods [ORG18] is guided by one central question: if the present prediction was true, how much re-learning would this require. The corresponding hypothesis is that wrong predictions would be more in conflict with the knowledge encoded in the deep neural network than correct ones, therefore causing increased re-learning stress. Given a predicted class

$$\hat{y} = \arg\max_{s \in \mathcal{S}} \mathbf{F}(\mathbf{x}) \tag{5}$$

we compute the gradient of layer ℓ corresponding to the predicted label. That is, given a loss function J, we compute

$$\nabla_\ell J(\hat{y}, \mathbf{x}, \boldsymbol{\theta}) \tag{6}$$

via backpropagation. The obtained quantities can be treated similarly to the case of forward pass features. While this concept seems to be prohibitively expensive for semantic segmentation (at least when calculating gradients for each pixel of the multidimensional output $\hat{\mathbf{y}}$), its applicability to object detection might be feasible, in particular, with respect to offline applications. Gradients are also of special interest in active learning with query by *expected model change* (cf. [Set10]).

In the context of semantic segmentation, geometrical information on segments' shapes as well as neighborhood relations of predicted segments can be taken into

account alongside with dispersion measures. It has been demonstrated [RS19, MRG20, RCH+20] that the detection of errors in an in-distribution setting strongly benefits from geometrical information. Recently, this has also been considered in scenarios under moderate domain shift [ORF20]. However, its applicability to out-of-distribution examples and to other sensors than the camera remains subject to further research.

The problem of realistically quantifying uncertainty measures will be taken up in Chapter "Uncertainty Quantification for Object Detection: Output- and Gradient-based Approaches" [RSKR22]. Here output-based and learning-gradient-based uncertainty metrics for object detection will be examined, showing that they are non-correlated. Thus, a combination of both paradigms leads, as is demonstrated, to a better object detection uncertainty estimate and by extension to a better overall detection accuracy.

6.5 Markov Random Fields

Although deep neural networks are currently the state of the art for almost all computer vision tasks, Markov random fields (MRF) remain one of the fundamental techniques used for many computer vision tasks, specifically image segmentation [LWZ09, KK11]. MRFs hold its power in the essence of being able to model dependencies between pixels in an image. With the use of energy functions, MRFs integrate pixels into models relating between unary and pair-wise pixels together [WKP13]. Given the model, MRFs are used to infer the optimal configuration yielding the lowest energy using mainly maximum a posteriori (MAP) techniques. Several MAP inference approaches are used to yield the optimal configuration such as graph cuts [KRBT08] and belief propagation algorithms [FZ10]. However, as with neural networks, MAP inference techniques result in deterministic point estimates of the optimal configuration without any sense of uncertainty in the output. To obtain uncertainties on results from MRFs, most of the work is directed toward modeling MRFs with Gaussian distributions. Getting uncertainties from MRFs with Gaussian distributions is possible by two typical methods: either approximate models are inferred to the posterior, from which sampling is easy or the variances can be estimated analytically, or approximate sampling from the posterior is used. Approximate models include those inferred using variational Bayesian (VB) methods, like mean-field approximations, and using Gaussian process (GP) models enforcing a simplified prior model [Bis06, LUAD16]. Examples of approximate sampling methods include traditional Markov chain Monte Carlo (MCMC) methods like Gibbs sampling [GG84]. Some recent theoretical advances propose the perturb-and-MAP framework and a Gumbel perturbation model (GPM) [PY11, HMJ13] to exactly sample from MRF distributions. Another line of work has also been proposed, where MAP inference techniques are used to estimate the probability of the network output. With the use of graph cuts, [KT08] try to estimate uncertainty using the min-marginals associated with the label assignments of a random field. Here, the work by Kohli and Torr [KT08]

was extended to show how this approach can be extended to techniques other than graph cuts [TA12] or compute uncertainties on multi-label marginal distributions [STP17]. A current research direction is the incorporation of MRFs with deep neural networks, along with providing uncertainties on the output [SU15, CPK+18]. This can also be extended to other forms of neural networks such as recurrent neural networks to provide uncertainties on segmentation of streams of videos with extending dependencies of pixels to previous frames [ZJRP+15, LLL+17].

6.6 Confidence Calibration

Neural network classifiers output a label $\hat{Y} \in \mathcal{Y}$ on a given input $\mathbf{x} \in \mathcal{D}$ with an associated confidence $\hat{P}$. This confidence can be interpreted as a probability of correctness that the predicted label matches the ground-truth label $\overline{Y} \in \mathcal{Y}$. Therefore, these probabilities should reflect the "self-confidence" of the system. If the empirical accuracy for any confidence level matches the predicted confidence, a model is called *well calibrated*. Therefore, a *classification* model is perfectly calibrated if

$$\underbrace{P(\hat{Y} = Y | \hat{P} = p)}_{\text{accuracy given } p} = \underbrace{p}_{\text{confidence}} \qquad \forall p \in [0, 1] \tag{7}$$

is fulfilled [GPSW17]. For example, assume 100 predictions with confidence values of 0.9. We call the model well calibrated if 90 out of these 100 predictions are actually correct. However, recent work has shown that modern neural networks tend to be too overconfident in their predictions [GPSW17]. The deviation of a model to the perfect calibration can be measured by the expected calibration error (ECE) [NCH15]. It is possible to recalibrate models as a post-processing step after classification. One way to get a calibration mapping is to group all predictions into several bins by their confidence. Using such a binning scheme, it is possible to compute the empirical accuracy for certain confidence levels, as it is known for a long time already in reconstructing confidence outputs for Viterbi decoding [HR90]. Common methods are histogram binning [ZE01], isotonic regression [ZE02], or more advanced methods like Bayesian binning into quantiles (BBQ) [NCH15] and ensembles of near-isotonic regression (ENIR) [NC16]. Another way to get a calibration mapping is to use scaling methods based on logistic regression like Platt scaling [Pla99], temperature scaling [GPSW17], and beta calibration [KSFF17].

In the setting of probabilistic regression, a model is calibrated if 95% of the true target values are below or equal to a credible level of 95% (so-called *quantile-calibrated regression*) [GBR07, KFE18, SDKF19]. A regression model is usually calibrated by fine-tuning its predicted CDF in a post-processing step to match the empirical frequency. Common approaches utilize isotonic regression [KFE18], logistic and beta calibration [SKF18], as well as Gaussian process models [SKF18, SDKF19] to build a calibration mapping. In contrast to quantile-calibrated regression, [SDKF19]

have recently introduced the concept of distribution calibration, where calibration is applied on a distribution level and naturally leads to calibrated quantiles. Recent work has shown that miscalibration in the scope of *object detection* also depends on the position and scale of a detected object [KKSH20]. The additional box regression output is denoted by $\hat{\mathbf{R}}$ with A as the size of the used box encoding. Furthermore, if we have no knowledge about all anchors of a model (which is a common case in many applications), it is not possible to determine the accuracy. Therefore, Küppers et al. [KKSH20] use the precision as a surrogate for accuracy and propose that an *object detection model* is perfectly calibrated if

$$\underbrace{P(M = 1|\hat{P} = p, \hat{Y} = y, \hat{\mathbf{R}} = \mathbf{r})}_{\text{precision given } p,y,\mathbf{r}} = \underbrace{p}_{\text{confidence}} \quad \forall p \in [0, 1], y \in \mathcal{Y}, \mathbf{r} \in \mathbb{R}^A \tag{8}$$

is fulfilled, where $M = 1$ denotes a correct prediction that matches a ground-truth object with a chosen IoU threshold and $M = 0$ denotes a mismatch, respectively. The authors propose the detection-expected calibration error (D-ECE) as the extension of the ECE to object detection tasks in order to measure miscalibration also by means of the position and scale of detected objects. Other approaches try to fine-tune the regression output in order to obtain more reliable object proposals [JLM+18, RTG+19] or to add a regularization term to the training objective such that training yields models that are both well performing and well calibrated [PTC+17, SSH19].

In Chapter "Confidence Calibration for Object Detection and Segmentation" [KHKS22], an extension of the ECE for general multivariate learning problems, such as object detection or image segmentation, will be discussed. In fact, the proposed multivariate confidence calibration is designed to take additional information into account, such as bounding boxes or shape descriptors. It is shown that this extended calibration reduces the calibration error as expected but also has a positive bearing on the quality of segmentation masks.

7 Aggregation

From a high-level perspective, a *neural network* is based on processing inputs and coming to some output conclusion, e.g., mapping incoming image data onto class labels. Aggregation or collection of non-independent information on either the input or output side of this network function can be used as a tool to leverage its performance and reliability. Starting with the input, any additional "dimension" to add data can be of use. For example, in the context of automated vehicles, this might be input from any further sensor measuring the same scene as the original one, e.g., stereo cameras or LiDAR. Combining those sensor sets for prediction is commonly referred to as *sensor fusion* [CBSW19]. Staying with the example, the scene will be monitored consecutively providing a whole (temporally ordered) stream of input information. This may be used either by adjusting the network for this kind of input [KLX+17] or

in terms of a post-processing step, in which the predictions are aggregated by some measure of temporal consistency.

Another more implicit form of aggregation is training the neural network on several "independent" tasks, e.g., segmentation and depth regression. Although the individual task is executed on the same input, the overall performance can still benefit from the correlation among all given tasks. We refer to the discussion on *multi-task networks* in Sect. 9.2. By extension, solving the same task in multiple different ways can be beneficial for performance and provide a measure of redundancy. In this survey, we focus on single-task systems and discuss *ensemble methods* in the next section and the use of *temporal consistency* in the one thereafter.

7.1 Ensemble Methods

Training a neural network is optimizing its parameters to fit a given training dataset. The commonly used gradient-based optimization schemes cause convergence in a "nearby" local minimum. As the loss landscapes of neural networks are notoriously non-convex [CHM+15], various locally optimal model parameter sets exist. These local optima differ in the degree of optimality ("deepness"), qualitative characteristics ("optimal for different parts of the training data"), and their generalizability to unseen data (commonly referred to by the geometrical terms of "sharpness" and "flatness" of minima [KMN+17]).

A single trained network corresponds to one local minimum of such a loss landscape and thus captures only a small part of a potentially diverse set of solutions. *Network ensembles* are collections of models and therefore better suited to reflect this multi-modality. Various modeling choices shape a loss landscape: the selected model class and its meta-parameters (like architecture and layer width), the training data and the optimization objective. Accordingly, approaches to diversify ensemble components range from combinations of different model classes over varying training data (bagging) to methods that train and weight ensemble components to make up for the flaws of other ensemble members (boosting) [Bis06].

Given the millions of parameters of application-size networks, ensembles of NNs are resource-demanding w.r.t. computational load, storage, and runtime during training and inference. This complexity increases linearly with ensemble size for naïve ensembling. Several approaches were put forward to reduce some dimensions of this complexity: *snapshot ensembles* [HLP+17] require only one model optimization with a cyclical learning-rate schedule leading to an optimized training runtime. The resulting training trajectory passes through several local minima. The corresponding models compose the ensemble. On the contrary, *model distillation* [HVD14] tackles runtime at inference. They "squeeze" an NN ensemble into a single model that is optimized to capture the gist of the model set. However, such a compression goes along with reduced performance compared to the original ensemble.

Several hybrids of single model and model ensemble exist: multi-head networks [AJD18] share a backbone network that provides inputs to multiple prediction net-

works. Another variant is mixture-of-expert model that utilizes a gating network to assign inputs to specialized expert networks [SMM+17]. Multi-task networks (cf. Sect. 9.2) and Bayesian approximations of NNs (cf. Sects. 6.2 and 6.3) can be seen as implicit ensembles.

NN ensembles (or deep ensembles) are not only used to boost model quality. They pose the frequentist's approach to estimating NN uncertainties and are state of the art in this regard [LPB17, SOF+19]. The emergent field of federated learning is concerned with the integration of decentrally trained ensemble components [MMR+17] and safety-relevant applications of ensembling range from automated driving [Zha12] to medical diagnostics [RRMH17]. Taking this safe-ML perspective, promising research directions comprise a more principled and efficient composition of model ensembles, by application-driven diversification as well as improved techniques to miniaturize ensembles and by gaining a better understanding of methods like model distillation. In the long run, better designed, more powerful learning systems might partially reduce the need for combining weaker models in a network ensemble.

In Chapter "Evaluating Mixture-of-Expert Architectures for Network Aggregation" [PHW22], the advantages and drawbacks of mixture-of-experts architectures with respect to robustness, interpretability, and overall test performance will be discussed. Two state-of-the-art architectures are investigated, obtaining, sometimes surpassing, baseline performance. What is more, the models exhibit improved awareness for OoD data.

7.2 Temporal Consistency

The focus of previous DNN development for semantic segmentation has been on single-image prediction. This means that the final and intermediate results of the DNN are discarded after each image. However, the application of a computer vision model often involves the processing of images in a sequence, i.e., there is a temporal consistency in the image content between consecutive frames (for a metric, cf., e.g., [VBB+20]). This consistency has been exploited in previous work to increase quality and reduce computing effort. Furthermore, this approach offers the potential to improve the robustness of DNN prediction by incorporating this consistency as a priori knowledge into DNN development. The relevant work in the field of video prediction can be divided into two major approaches:

1. DNNs are specially designed for video prediction. This usually requires training from scratch and the availability of training data in a sequence.
2. A transformation from single prediction DNNs to video prediction DNNs takes place. Usually no training is required, i.e., the existing weights of the model can be used unaltered.

The first set of approaches often involves *conditional random fields* (CRF) and its variants. CRFs are known for their use as post-processing step in the prediction of

semantic segmentation, in which their parameters are learned separately or jointly with the DNN [ZJRP+15]. Another way to use spatiotemporal features is to include 3D convolutions, which add an additional dimension to the conventional 2D convolutional layer. Tran et al. [TBF+15] use 3D convolution layers for video recognition tasks, such as action and object recognition. One further approach to use spatial and temporal characteristics of the input data is to integrate *long short-term memory* (LSTM) [HS97], a variant of the *recurrent neural network* (RNN). Fayyaz et al. [FSS+16] integrate LSTM layers between the encoder and decoder of their convolutional neural network for semantic segmentation. The significantly higher GPU memory requirements and computational effort are disadvantages of this method. More recently, Nilsson and Sminchisescu [NS18] deployed *gated recurrent units*, which generally require significantly less memory. An approach to improve temporal consistency of automatic speech recognition outputs is known as a posterior-in-posterior-out (PIPO) LSTM "sequence enhancer", a postfilter which could be applicable to video processing as well [LSFs19]. A disadvantage of the described methods is that sequential data for training must be available, which may be limited or show a lack of diversity.

The second class of approaches has the advantage that it is model independent most of the time. Shelhamer et al. [SRHD16] found that the deep feature maps within the network change only slightly with temporal changes in video content. Accordingly, [GJG17] calculate the optical flow of the input images from time steps $t = 0$ and $t = -1$ and convert it into the so-called *transform flow* which is used to transform the feature maps of the time step $t = -1$ so that an aligned representation to the feature map $t = 0$ is achieved. Sämann et al. [SAMG19] use a confidence-based combination of feature maps from previous time steps based on the calculated optical flow.

8 Verification and Validation

Verification and validation is an integral part of the safety assurance for any safety-critical systems. As of the functional safety standard for automotive systems [ISO18], *verification* means to determine whether given requirements are met [ISO18, 3.180], such as performance goals. *Validation*, on the other side, tries to assess whether the given requirements are sufficient and adequate to guarantee safety [ISO18, 3.148], e.g., whether certain types of failures or interactions simply were overlooked. The latter is usually achieved via extensive testing in real operation conditions of the integrated product. This differs from the notion of validation used in the machine learning community in which it usually refers to simple performance tests on a selected dataset. In this section, we want to concentrate on general verification aspects for deep neural networks.

Verification as in the safety domain encompasses (manual) inspection and analysis activities, and testing. However, the contribution of single processing steps within a neural network to the final behavior can hardly be assessed manually (compared

to the problem of interpretability in Sect. 5). Therefore, we here will concentrate on different approaches to verification testing, i.e., formal testing and black-box methods.

The discussion on how to construct an assurance case and how to derive suitable acceptance criteria for ML models in the scope of autonomous driving, as well as how to determine and acquire sufficient evidences for such an assurance case is provided in Chapter "Safety Assurance of Machine Learning for Perception Functions" [BHH+22]. The authors look into several use cases on how to systematically obtain evidences and incorporate them into a structured safety argumentation.

In Chapter "A Variational Deep Synthesis Approach for Perception Validation" [GHS22], a novel data generation framework is introduced, aiming at validating perception functions, in particular, in the context of autonomous driving. The framework allows for effectively defining and testing common traffic scenes. Experiments are performed for pedestrian detection and segmentation in versatile urban scenarios.

8.1 Formal Testing

Formal testing refers to testing methods that include formalized and formally verifiable steps, e.g., for test data acquisition, or for verification in the local vicinity of test samples. For image data, local testing around valid samples is usually more practical than fully formal verification: (safety) properties are not expected to hold on the complete input space but only on the much smaller unknown lower dimensional manifold of real images [WGH19]. Sources of such samples can be real ones or generated ones using an input space formalization or a trained generative model.

Coverage criteria for the data samples are commonly used for two purposes: (a) deciding when to stop testing or (b) identifying missing tests. For CNNs, there are at least three different approaches toward coverage: (1) approaches that establish coverage based on a model with semantic features of the input space [GHHW20], (2) approaches trying to semantically cover the latent feature space of neural network or a proxy network (e.g., an autoencoder) [SS20a], and (3) approaches trying to cover neurons and their interactions, inspired by classical software white-box analysis [PCYJ19, SHK+19].

Typical types of properties to verify are simple test performance, local stability (robustness), a specific structure of the latent spaces like embedding of semantic concepts [SS20b], and more complex logical constraints on inputs and outputs, which can be used for testing when fuzzified [RDG18]. Most of these properties require in-depth semantic information about the DNN inner workings, which is often only available via interpreting intermediate representations [KWG+18], or interpretable proxies/surrogates (cf. Sect. 5.4), which do not guarantee fidelity.

There exist different testing and formal verification methods from classical software engineering that have already been applied to CNNs. *Differential testing* as used by DeepXPlore [PCYJ19] trains n different CNNs for the same task using independent datasets and compares the individual prediction results on a test set. This

allows to identify inconsistencies between the CNNs but no common weak spots. *Data augmentation* techniques start from a given dataset and generate additional transformed data. *Generic data augmentation* for images like rotations and translation is state of the art for training but may also be used for testing. *Concolic testing* approaches incrementally grow test suites with respect to a coverage model to finally achieve completeness. Sun et al. [SWR+18] use an adversarial input model based on some norm (cf. Sect. 4.1), e.g., an L_p-norm, for generating additional images around a given image using concolic testing. *Fuzzing* generates new test data constrained by an input model and tries to identify interesting test cases, e.g., by optimizing white-box coverage mentioned above [OOAG19]. Fuzzing techniques may also be combined with the differential testing approach discussed above [GJZ+18]. In all these cases, it needs to be ensured that the image as well as its meta-data remains valid for testing after transformation. Finally, *proving methods* surveyed by Liu et al. [LAL+20] try to formally prove properties on a trained neural network, e.g., based on satisfiability modulo theories (SMT). These approaches require a formal characterization of an input space and the property to be checked, which is hard for non-trivial properties like contents of an image.

Existing formal testing approaches can be quite costly to integrate into testing workflows: differential testing and data augmentation require several inferences per initial test sample; concolic and fuzzy testing apply an optimization to each given test sample, while convergence toward the coverage goals is not guaranteed; also, the iterative approaches need tight integration into the testing workflow; and lastly, proving methods usually have to balance computational efficiency against the precision or completeness of the result [LAL+20]. Another challenge of formal testing is that machine learning applications usually solve problems for which no (formal) specification is possible. This makes it hard to find useful requirements for testing [ZHML20] and properties that can be formally verified. Even partial requirements, such as specification of useful input perturbations, specified corner cases, and valuable coverage goals, are typically difficult to identify [SS20a, BTLFs20].

8.2 Black-Box Methods

In machine learning literature, neural networks are often referred to as black boxes due to the fact that their internal operations and their decision-making are not completely understood [SZT17], hinting at a lack of interpretability and transparency. However, in this survey, we consider a black box to be a machine learning model to which we only have Oracle (query) access [TZJ+16, PMG+17]. That means we can query the model to get input–output pairs, but we do not have access to the specific architecture (or weights, in case of neural networks). As [OAFS18] describes, black boxes are increasingly widespread, e.g., health care, autonomous driving, or ML as a service in general, due to proprietary, privacy, or security reasons. As deploying black boxes gains popularity, so do methods that aim to extract internal information, such as architecture and parameters, or to find out, whether a sample belongs to the

training dataset. These include *model extraction attacks* [TZJ+16, KTP+20], *membership inference attacks* [SSSS17], general attempts to reverse-engineer the model [OAFS18], or to attack it adversarially [PMG+17]. Protection and counter-measures are also actively researched: Kesarwani et al. [KMAM18] propose a warning system that estimates how much information an attacker could have gained from queries. Adi et al. [ABC+18] use watermarks for models to prevent illegal re-distribution and to identify intellectual property.

Many papers in these fields make use of so-called *surrogate*, *avatar*, or *proxy models* that are trained on input–output pairs of the black box. In case the black-box output is available in soft form (e.g., logits), distillation as first proposed by [HVD14] can be applied to train the surrogate (student) model. Then, any white-box analysis can be performed on the surrogates (cf. e.g., [PMG+17]) to craft adversarial attacks targeted at the black box. More generally, (local) surrogates, as, for example, in [RSG16], can be used to (locally) explain its decision-making. Moreover, these techniques are also of interest if one wants to compare or test black-box models (cf. Sect. 8.1). This is the case, among others, in ML marketplaces, where you wish to buy a pre-trained model [ABC+18], or if you want to verify or audit that a third-party black-box model obeys regulatory rules (cf. [CH19]).

Another topic of active research is so-called *observers*. The concept of observers is to evaluate the interface of a black-box module to determine if it behaves as expected within a given set of parameters. The approaches can be divided into *model-explaining* and *anomaly-detecting* observers. First, model explanation methods answer the question of which input characteristic is responsible for changes at the output. The observer is able to alter the inputs for this purpose. If the input of the model under test evolves only slightly but the output changes drastically, this can be a signal that the neural network is misled, which is also strongly related to adversarial examples (cf. Sect. 4). Hence, the reason for changes in the classification result via the input can be very important. In order to figure out in which region of an input image the main reason for the classification is located, [FV17] "delete" information from the image by replacing regions with generated patches until the output changes. This replaced region is likely responsible for the decision of the neural network. Building upon this, [UEKH19] adapt the approach to medical images and generate "deleted" regions by a variational autoencoder (VAE). Second, anomaly-detecting observers register input and output anomalies, either examining input and output independently or as an input–output pair, and predict the black-box performance in the current situation. In contrast to model-explaining approaches, this set of approaches has high potential to be used in an online scenario since it does not need to modify the model input. The maximum mean discrepancy (MMD) [BGR+06] measures the domain gap between two data distributions independently from the application and can be used to raise a warning if input or output distributions during inference deviate too strongly from their respective training distributions. Another approach is using a GAN-based autoencoder [LFK+20] to perform a domain shift estimation where the Wasserstein distance is used as domain mismatch metric. This metric can also be evaluated by use of a casual time-variant aggregation of distributions during inference time.

9 Architecture

In order to solve a specific task, the architecture of a CNN and its building blocks play a significant role. Since the early days of using CNNs in image processing, when they were applied to handwriting recognition [LBD+89] and the later breakthrough in general image classification [KSH12], the architecture of the networks has changed radically. Did the term of *deep learning* for these first convolutional neural networks imply a depth of approximately four layers, their depth increased significantly during the last years and new techniques had to be developed to successfully train and utilize these networks [HZRS16]. In this context, new activation functions [RZL18] as well as new loss functions [LGG+17] have been designed and new optimization algorithms [KB15] were investigated.

With regard to the layer architecture, the initially alternating repetition of convolution and pooling layers as well as their characteristics have changed significantly. The convolution layers made the transition from a few layers with often large filters to many layers with small filters. A further trend was then the definition of entire modules, which were used repeatedly within the overall architecture as so-called *network in network* [LCY14].

In areas such as automated driving, there is also a strong interest in the simultaneous execution of different tasks within one single convolutional neural network architecture. This kind of architecture is called *multi-task learning* (MTL) [Car97] and can be utilized in order to save computational resources and at the same time to increase performance of each task [KTMFs20]. Within such multi-task networks, usually one shared feature extraction part is followed by one separate so-called head per task [TWZ+18].

In each of these architectures, manual design using expert knowledge plays a major role. The role of the expert is the crucial point here. In recent years, however, there have also been great efforts to automate the process of finding architectures for networks or, in the best case, to learn them. This is known under the name *neural architecture search* (NAS).

9.1 Building Blocks

Designing a convolutional neural network typically includes a number of design choices. The general architecture usually contains a number of convolutional and pooling layers which are arranged in a certain pattern. Convolutional layers are commonly followed by a nonlinear activation function. The learning process is based on a loss function which determines the current error and an optimization function that propagates the error back to the single convolution layers and its learnable parameters.

When CNNs became state of the art in computer vision [KSH12], they were usually built using a few alternating convolutional and pooling layers having a few

fully connected layers in the end. It turned out that better results are achieved with deeper networks and so the number of layers increased [SZ15] over the years. To deal with these deeper networks, new architectures had to be developed. In a first step, to reduce the number of parameters, the convolutional layers with partly large filter kernels were replaced by several layers with small 3×3 kernels. Today, most architectures are based on the *network-in-network* principle [LCY14], where more complex modules are used repeatedly. Examples of such modules are the *inception module* from `GoogleNet` [SLJ+15] or the *residual block* from `ResNet` [HZRS16]. While the inception module consists of multiple parallel strings of layers, the residual blocks are based on the *highway network* [SGS15], which means that they can bypass the original information and the layers in between are just learning residuals. With `ResNeXt` [XGD+17] and `Inception-ResNet` [SIVA17] there already exist two networks that combine both approaches. For most tasks, it turned out that replacing the fully connected layers by convolutional layers is much more convenient making the networks fully convolutional [LSD15]. These so-called *fully convolutional networks* (FCN) are no longer bound to fixed input dimensions. Note that with the availability of convolutional long short-term memory (ConvLSTM) structures also fully convolutional recurrent neural networks (FCRNs) became available for fully scalable sequence-based tasks [SCW+15, SDF+20].

Inside the CNNs, the *rectified linear unit* (ReLU) has been the most frequently used activation function for a long time. However, since this function suffers from problems related to the mapping of all negative values to zero like the vanishing gradient problem, new functions have been introduced in recent years. Examples are the *exponential linear unit* (ELU), *swish* [RZL18], and the *non-parametric linearly scaled hyperbolic tangent* (LiSHT) [RMDC19]. In order to be able to train a network consisting of these different building blocks, the loss function is the most crucial part. This function is responsible for how and what the network ultimately learns and how exactly the training data is applied during the training process to make the network train faster or perform better. So the different classes can be weighted in a classification network with fixed values or so-called α-balancing according to their probability of occurrence. Another interesting approach is weighting training examples according to their easiness for the current network [LGG+17, WFZ19]. For multi-task learning also weighting tasks based on their uncertainty [KGC18] or gradients [CBLR18] can be done as further explained in Sect. 9.2. A closer look on how a modification of the loss function might affect safety-related aspects is given in Sect. 3, Modification of Loss.

9.2 Multi-task Networks

Multi-task learning (MTL) in the context of neural networks describes the process of optimizing several tasks simultaneously by learning a unified feature representation [Car97, RBV18, GHL+20, KBFs20] and coupling the task-specific loss contributions, thereby enforcing cross-task consistency [CPMA19, LYW+19, KTMFs20].

Unified feature representation is usually implemented by sharing the parameters of the initial layers inside the encoder (also called feature extractor). It not only improves the single tasks by more generalized learned features but also reduces the demand for computational resources at inference. Not an entirely new network has to be added for each task but only a task-specific decoder head. It is essential to consider the growing amount of visual perception tasks in automated driving, e.g., depth estimation, semantic segmentation, motion segmentation, and object detection. While the parameter sharing can be soft, as in *cross stitch* [MSGH16] and *sluice networks* [RBAS18], or hard [Kok17, TWZ+18], meaning ultimately sharing the parameters, the latter is usually preferred due to its straightforward implementation and lower computational complexity during training and inference.

Compared to implicitly coupling tasks via a shared feature representation, there are often more direct ways to optimize the tasks inside cross-task losses jointly. It is only made possible as, during MTL, there are network predictions for several tasks, which can be enforced to be consistent. As an example, sharp depth edges should only be at class boundaries of semantic segmentation predictions. Often both approaches to MTL are applied simultaneously [YZS+18, CLLW19] to improve a neural network's performance as well as to reduce its computational complexity at inference.

While the theoretical expectations for MTL are quite clear, it is often challenging to find a good weighting strategy for all the different loss contributions as there is no theoretical basis on which one could choose such a weighting with early approaches either involving heuristics or extensive hyperparameter tuning. The easiest way to balance the tasks is to use uniform weight across all tasks. However, the losses from different tasks usually have different scales, and uniformly averaging them suppresses the gradient from tasks with smaller losses. Addressing these problems, Kendall et al. [KGC18] propose to weigh the loss functions by the *homoscedastic uncertainty* of each task. One does not need to tune the weighting parameters of the loss functions by hand, but they are adapted automatically during the training process. Concurrently Chen et al. [CBLR18] propose *GradNorm*, which does not explicitly weigh the loss functions of different tasks but automatically adapts the gradient magnitudes coming from the task-specific network parts on the backward pass. Liu et al. [LJD19] proposed dynamic weight average (DWA), which uses an average of task losses over time to weigh the task losses.

9.3 Neural Architecture Search

In the previous sections, we saw manually engineered modifications of existing CNN architectures proposed by `ResNet` [HZRS16] or `Inception` [SLJ+15]. They are results of human design and showed their ability to improve performance. `ResNet` introduces a *skip connection* in building blocks and `Inception` makes use of its specific *inception module*. Hereby, the intervention by an expert is crucial. The

approach of *neural architecture search* (NAS) aims to automate this time-consuming and manual design of neural network architectures.

NAS is closely related to hyperparameter optimization (HO), which is described in Sect. 3, hyperparameter optimization. Originally, both tasks were solved simultaneously. Consequently, the kernel size or number of filters were seen as additional hyperparameters. Nowadays, the distinction between HO and NAS should be stressed. The concatenation of complex building blocks or modules cannot be accurately described with single parameters. This simplification is no longer suitable.

To describe the NAS process, the authors of [EMH19b] define three steps: (1) definition of search space, (2) search strategy, and (3) performance estimation strategy.

Majority of search strategies take advantage of the *NASNet search space* [ZVSL18] which arranges various operations, e.g., convolution, pooling within a single cell. However, other spaces based on a chain or multi-branch structure are possible [EMH19b]. The search strategy comprises advanced methods from sequential model-based optimization (SMBO) [LZN+18], Bayesian optimization [KNS+18], evolutionary algorithms [RAHL19, EMH19a], reinforcement learning [ZVSL18, PGZ+18], and gradient descent [LSY19, SDW+19]. Finally, the performance estimation describes approximation techniques due to the impracticability of multiple evaluation runs. For a comprehensive survey regarding the NAS process, we refer to [EMH19b]. Recent research has shown that reinforcement learning approaches such as NASNet-A [ZVSL18] and ENAS [PGZ+18] are partly outperformed by evolutionary algorithms, e.g., AmoebaNet [RAHL19] and gradient-based approaches, e.g., DARTS [LSY19].

Each of these approaches focuses on different optimization aspects. Gradient-based methods are applied to a continuous search space and offer faster optimization. On the contrary, the evolutionary approach LEMONADE [EMH19a] enables multi-object optimization by considering the conjunction of resource consumption and performance as the two main objectives. Furthermore, single-path NAS [SDW+19] extends the multi-path approach of former gradient-based methods and proposes the integration of "over-parameterized superkernels", which significantly reduces memory consumption.

The focus of NAS is on the optimized combination of humanly predefined CNN elements with respect to objectives such as resource consumption and performance. NAS offers automation, however, the realization of the objectives is strongly limited by the potential of the CNN elements.

10 Model Compression

Recent developments in CNNs have resulted in neural networks being the state of the art in computer vision tasks like image classification [KSH12, HZRS15, MGR+18], object detection [Gir15, RDGF15, HGDG17], and semantic segmentation [CPSA17, ZSR+19, LBS+19, WSC+20]. This is largely due to the increasing availability of

hardware computational power and an increasing amount of training data. We also observe a general upward trend of the complexities of the neural networks along with their improvement in state-of-the-art performance. These CNNs are largely trained on back-end servers with significantly higher computing capabilities. The use of these CNNs in real-time applications is inhibited due to the restrictions on hardware, model size, inference time, and energy consumption. This led to an emergence of a new field in machine learning, commonly termed as model compression. Model compression basically implies reducing the memory requirements, inference times, and model size of DNNs to eventually enable the use of neural networks on edge devices. This is tackled by different approaches, such as *network pruning* (identifying weights or filters that are not critical for network performance), *weight quantizations* (reducing the precision of the weights used in the network), *knowledge distillation* (a smaller network is trained with the knowledge gained by a bigger network), and *low-rank factorization* (decomposing a tensor into multiple smaller tensors). In this section, we introduce some of these methods for model compression and discuss in brief the current open challenges and possible research directions with respect to its use in automated driving applications.

In Chapter "Joint Optimization for DNN Model Compression and Corruption Robustness" [VHB+22], the concept of both pruning and quantization on the application of semantic segmentation in autonomous driving is elucidated, while at the same time robustifying the model against corruptions in the input data. An improved performance and robustness for a state-of-the-art model for semantic segmentation is demonstrated.

10.1 Pruning

Pruning has been used as a systematic tool to reduce the complexity of deep neural networks. The redundancy in DNNs may exist on various levels, such as the individual weights, filters, and even layers. All the different methods for pruning try to take advantage of these available redundancies on various levels. Two of the initial approaches for neural networks proposed weight pruning in the 1990s as a way of systematically damaging neural networks [CDS90, Ree93]. As these weight pruning approaches do not aim at changing the structure of the neural network, these approaches are called *unstructured pruning*. Although there is reduction in the size of the network when it is saved in sparse format, the acceleration depends on the availability of hardware that facilitates sparse multiplications. As pruning filters and complete layers aim at exploiting the available redundancy in the architecture or structure of neural networks, these pruning approaches are called *structured pruning*. Pruning approaches can also be broadly classified into: data-dependent and data-independent methods. Data-dependent methods [LLS+17, LWL17, HZS17] make use of the training data to identify filters to prune. Theis et al. [TKTH18] and Molchanov et al. [MTK+17] propose a greedy pruning strategy that identifies the importance of feature maps one at a time from the network and measures the effect

of removal of the filters on the training loss. This means that filters corresponding to those feature maps that have least effect on training loss are removed from the network. Within data-independent methods [LKD+17, HKD+18, YLLW18, ZQF+18], the selection of CNN filters to be pruned are based on the statistics of the filter values. Li et al. [LKD+17] proposed a straightforward method to calculate the rank of filters in a CNN. The selection of filters are based on the L_1-norm, where the filter with the lowest norm is pruned away. He et al. [HZS17] employ a least absolute shrinkage and selection operator (LASSO) regression-based selection of filters to minimize the least squares reconstruction.

Although the abovementioned approaches demonstrated that a neural network can be compressed without affecting the accuracy, the effect on robustness is largely unstudied. Dhillon et al. [DAL+18] proposed pruning a subset of activations and scaling up the survivors to show improved adversarial robustness of a network. Lin et al. [LGH19] quantize the precision of the weights after controlling the Lipschitz constant of layers. This restricts the error propagation property of adversarial perturbations within the neural network. Ye et al. [YXL+19] evaluated the relationship between adversarial robustness and model compression in detail and show that naive compression has a negative effect on robustness. Gui et al. [GWY+19] co-optimize robustness and compression constraints during the training phase and demonstrate improvement in the robustness along with reduction in the model size. However, these approaches have mostly been tested on image classification tasks and on smaller datasets only. Their effectiveness on safety-relevant automated driving tasks, such as object detection and semantic segmentation, is not studied and remains an open research challenge.

10.2 Quantization

Quantization of a random variable x having a probability density function $p(x)$ is the process of dividing the range of x into intervals, each is represented using a single value (also called reconstruction value), such that the following reconstruction error is minimized:

$$\sum_{i=1}^{I} \int_{b_i}^{b_{i+1}} (q_i - x)^2 p(x) dx, \tag{9}$$

where b_i is the left-side border of the i-th interval, q_i is its reconstruction value, and I is the number of intervals, e.g., $I = 8$ for a 3-bit quantization. This definition can be extended to multiple dimensions as well.

Quantization of neural networks has been around since the 1990s [Guo18], however, with a focus in the early days on improving the hardware implementations of these networks. In the deep learning literature, a remarkable application of quantization combined with unstructured pruning can be found in the approach of *deep compression* [HMD16], where one-dimensional k-means is utilized to cluster the

weights per layer and thus finding the I cluster centers (q_i values in (9)) iteratively. This procedure conforms to an implicit assumption that $p(x)$ has the same spread inside all clusters. Deep compression can reduce the network size needed for image classification by a factor of 35 for `AlexNet` and a factor of 49 for `VGG-16` without any loss in accuracy. However, as pointed out in [JKC+18], these networks from the early deep learning days are over-parameterized and a less impressing compression factor is thus expected when the same technique is applied to lightweight architectures, such as `MobileNet` and `SequeezeNet`. For instance, considering `SqueezeNet` (50 times smaller than `AlexNet`), the compression factor of deep compression without accuracy loss drops to about 10.

Compared to scalar quantization used in deep compression, there were attempts to exploit the structural information by applying variants of vector quantization of the weights [GLYB14, CEL20, SJG+20]. Remarkably, in the latter (i.e., [SJG+20]), the reconstruction error of the activations (instead of the weights) is minimized in order to find an optimal codebook for the weights, as the ultimate goal of quantization is to approximate the network's output not the network itself. This is performed in a layer-by-layer fashion (as to prevent error accumulation) using activations generated from unlabeled data.

Other techniques [MDSN17, JKC+18] apply variants of so-called "linear" *quantization*, i.e., the quantization staircase has a fixed interval size. This paradigm conforms to an implicit assumption that $p(x)$ in (9) is uniform and is thus also called *uniform quantization*. The uniform quantization is widely applied both in specialized software packages, such as `Texas Instruments Deep Learning Library` (automotive boards) [MDS+18], and in general-purpose libraries, such as `Tensorflow Lite`. The linearity assumption enables practical implementations, as the quantization and dequantization can be implemented using a scaling factor and an intercept, whereas no codebook needs to be stored. In many situations, the intercept can be omitted by employing a symmetric quantization mapping. Moreover, for power of 2 ranges, the scaling ends up being a bit-wise shift operator, where quantization and dequantization differ only in the shift direction. It is also straightforward to apply this scheme *dynamically*, i.e., for each tensor separately using a tensor-specific multiplicative factor. This can be easily applied not only to filters (weight tensors) but also to activation tensors (see, for instance, [MDSN17]).

Unless the scale factor in the linear quantization is assumed constant by construction, it is computed based on the statistics of the relevant tensor and can be thus sensitive to outliers. This is known to result in a low-precision quantization. In order to mitigate this issue, the original range can be *clipped* and thus reduced to the most relevant part of the signal. Several approaches are proposed in the literature for finding an optimal clipping threshold: simple percentile analysis of the original range (e.g., clipping 2% of the largest magnitude values), minimizing the mean squared error between the quantized and original range in the spirit of (9) [BNS19], or minimizing the Kullback–Leibler divergence between the original and the quantized distributions [Mig17]. While the clipping methods trade off large quantization errors of outliers against small errors of inliers [WJZ+20], other methods tackle the

outliers problem using a different trade-off, see, for instance, the outlier channel splitting approach in [ZHD+19].

An essential point to consider when deciding for a quantization approach for a given problem is the allowed or intended interaction with the training procedure. The so-called *post-training quantization*, i.e., quantization of a pre-trained network, seems to be attractive from a practical point of view: no access to training data is required and the quantization and training toolsets can be independent from each other. On the other hand, the training-aware quantization methods often yield higher inference accuracy and shorter training times. The latter is a serious issue for large complicated models which may need weeks to train on modern GPU clusters. The training-aware quantization can be implemented by inserting fake quantization operators in the computational graph of the forward pass during training (simulated quantization), whereas the backward pass is done as usual in floating-point resolution [JKC+18]. Other approaches [ZWN+18, ZGY+19] go a step further by quantizing the gradients as well. This leads to much lower training time, as the time of the often computationally expensive backward pass is reduced. The gradient's quantization, however, is not directly applicable as it requires the derivative of the quantization function (staircase-like), which is zero almost everywhere. Luckily, this issue can be handled by employing a *straight-through estimator* [BLC13] (approximating the quantization function by an identity mapping). There are also other techniques proposed recently to mitigate this problem [UMY+20, LM19].

11 Discussion

We have presented an extensive overview of approaches to effectively handle safety concerns accompanying deep learning: lack of generalization, robustness, explainability, plausibility, and efficiency. It has been described which lines of research we deem prevalent, important, and promising for each of the individual topics and categories into which the presented methods fall.

The reviewed methods alone will not provide safe-ML systems as such and neither will their future extensions. This is due to the limitations of quantifying complex real-world contexts. A complete and plausible safety argumentation will, thus, require more than advances in methodology and theoretical understanding of neural network properties and training processes. Apart from methodological progress, it will be necessary to gain practical experience in using the presented methods to gather evidence for overall secure behavior, using this evidence to construct a tight safety argument, and testing its validity in various situations.

In particular, each autonomously acting robotic system with state-of-the-art deep-learning-based perception and non-negligible actuation may serve as an object of study and is, in fact, in need of this kind of systematic reasoning before being transferred to widespread use or even market entry. We strongly believe that novel scientific insights, the potential market volume, and public interest will drive the arrival of reliant and trustworthy AI technology.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)". The authors would like to thank the consortium for the successful cooperation.

References

[AAAB18] S. Akcay, A. Atapour-Abarghouei, T.P. Breckon, Ganomaly: semi-supervised anomaly detection via adversarial training, in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, Perth, WA, Australia (2018), pp. 622–637

[ABC+18] Y. Adi, C. Baum, M. Cisse, B. Pinkas, J. Keshet, Turning your weakness into a strength: watermarking deep neural networks by backdooring, in *Proceedings of the USENIX Security Symposium*, Baltimore, MD, USA (2018), pp. 1615–1631

[ACKK14] S. Amershi, M. Cakmak, W.B. Knox, T. Kulesza, Power to the people: the role of humans in interactive machine learning. AI Mag. **35**(4), 105–120 (2014)

[ACÖG17] M. Ancona, E. Ceolini, C. Öztireli, M.H. Gross, A unified view of gradient-based attribution methods for deep neural networks (2017), pp. 1–11. arXiv:1711.06104

[ACS19] M. Angus, K. Czarnecki, R. Salay, Efficacy of pixel-level OOD detection for semantic segmentation (2019), pp. 1–13. arXiv:1911.02897

[ACW18] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden (2018), pp. 274–283

[AEIK18] A. Athalye, L. Engstrom, A. Ilyas, K. Kwok, Synthesizing robust adversarial examples, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden (2018), pp. 284–293

[AGM+20] J. Adebayo, J. Gilmer, M. Muelly, I.J. Goodfellow, M. Hardt, B. Kim, Sanity checks for saliency maps (2020), pp. 1–30. arXiv:1810.03292

[AJD18] S. Arık, H. Jun, G. Diamos, Fast spectrogram inversion using multi-head convolutional neural networks. IEEE Signal Proc. Lett. **26**(1), 94–98 (2018)

[AK12] M.G. Augasta, T. Kathirvalavakumar, Rule extraction from neural networks – a comparative study, in *Proceedings of the International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME)*, Salem, India (2012), pp. 404–408

[AW19] A. Azulay, Y. Weiss, Why do deep convolutional networks generalize so poorly to small image transformations? (2019), pp. 1–25. arXiv:1805.12177

[BBL+07] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, R.J. Plemmons, Algorithms and applications for approximate nonnegative matrix factorization. Comput. Stat. & Data Anal. **52**(1), 155–173 (2007)

[BBLFs19] J.-A. Bolte, A. Bär, D. Lipinski, T. Fingscheidt, Towards corner case detection for autonomous driving, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, Paris, France (2019), pp. 438–445

[BBM+15] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On Pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLOS ONE **10**(7), 1–46 (2015)

[BCG+18] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R.N. Gunn, A. Hammers, D.A. Dickie, M. del C. Valdés Hernández, J.M. Wardlaw, D. Rueckert, GAN augmentation: augmenting training data using generative adversarial networks (2018), pp. 1–12. arXiv:1810.10863

[BCK+07] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, J. Wortman, Learning bounds for domain adaptation, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2007), pp. 129–136

[BCKW15] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, Lille, France (2015), pp. 1613–1622

[BCV13] Y. Bengio, A. Courville, P. Vincent, R. Learning, A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **35**(8), 1798–1828 (2013)

[BDBC+10] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, J. Vaughan, A theory of learning from different domains. Mach. Learn. **79**(1), 151–175 (2010)

[BF17] S. Baluja, I. Fischer, Adversarial transformation networks: learning to generate adversarial examples (2017), pp. 1–13. arXiv:1703.09387

[BFS18] A. Bhattacharyya, M. Fritz, B. Schiele, Long-term on-board prediction of people in traffic scenes under uncertainty, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 4194–4202

[BFW+19] Y. Bai, Y. Feng, Y. Wang, T. Dai, S.-T. Xia, Y. Jiang, Hilbert-based generative defense for adversarial examples, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea (2019), pp. 4784–4793

[BGR+06] K.M. Borgwardt, A. Gretton, M.J. Rasch, H.-P. Kriegel, B. Schölkopf, A.J. Smola, Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics **22**(14), 49–57 (2006)

[BGS+19] S. Burton, L. Gauerhof, B.B. Sethy, I. Habli, R. Hawkins, Confidence arguments for evidence of performance in machine learning for highly automated driving functions, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, Toulouse, France (2019), pp. 365–377

[BHG+19] A. Boloor, X. He, C. Gill, Y. Vorobeychik, X. Zhang, Simple physical adversarial examples against end-to-end autonomous driving models, in *Proceedings of the IEEE International Conference on Embedded Software and Systems (ICESS)*, Las Vegas, NV, USA (2019), pp. 1–7

[BHH+22] S. Burton, C. Hellert, F. Hüger, M. Mock, A. Rohatschek, Safety assurance of machine learning for perception functions, in *Deep Neural Networks and Data for Automated Driving – Robustness, Uncertainty Quantification, and Insights Towards Safety*, eds. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 365–387

[BHLHL+16] T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato, Y. Li, R. Turner, Deep gaussian processes for regression using approximate expectation propagation, in *Proceedings of the International Conference on Machine Learning (ICML)*, New York, NY, USA (2016), pp. 1472–1481

[BHP+18] C.P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, A. Lerchner, Understanding Disentangling in β-VAE (2018), pp. 1–11. arXiv:1804.03599

[BHSFs19] A. Bär, F. Hüger, P. Schlicht, T. Fingscheidt, On the robustness of redundant teacher-student frameworks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Long Beach, CA, USA (2019), pp. 1380–1388

[Bis06] C.M. Bishop, *Pattern Recognition and Machine Learning* (Springer, Berlin, 2006)

[BKD19] C.A. Brust, C. Käding, J. Denzler, Active learning for deep object detection, in *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, Prague, Czech Republic (2019), pp. 181–190

[BKOŠ18] P. Bevandić, I. Krešo, M. Oršić, S. Šegvić, Discriminative out-of-distribution detection for semantic segmentation (2018), pp. 1–18. arXiv:1808.07703

[BKV+20] A. Bär, M. Klingner, S. Varghese, F. Hüger, P. Schlicht, T. Fingscheidt, Robust semantic segmentation by redundant networks with a layer-specific loss contribution and majority vote, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2020), pp. 1348–1358, Virtual Conference

[BLC13] Y. Bengio, N. Léonard, A.C. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation (2013), pp. 1–12. arXiv:1308.3432

[BMM18] G. Bagschik, T. Menzel, M. Maurer, Ontology based scene creation for the development of automated vehicles, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China (2018), pp. 1813–1820

[BMR+17] T.B. Brown, D. Mané, A. Roy, M. Abadi, J. Gilmer, Adversarial patch, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops*, Long Beach, CA, USA (2017), pp. 1–6

[BNS19] R. Banner, Y. Nahshan, D. Soudry, Post training 4-bit quantization of convolutional networks for rapid-deployment, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2019), pp. 7948–7956

[BRW18] C. Bunne, L. Rahmann, T. Wolf, Studying invariances of trained convolutional neural networks (2018), pp. 1–7. arXiv:1803.05963

[BSC18] Y. Balaji, S. Sankaranarayanan, R. Chellappa, MetaReg: towards domain generalization using meta-regularization, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2018), pp. 1006–1016

[BTLFs20] J. Breitenstein, J.-A. Termöhlen, D. Lipinski, T. Fingscheidt, Systematization of corner cases for visual perception in automated driving, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)* (2020), pp. 1257–1264. Virtual Conference

[BXS+20] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J.M.F. Moura, P. Eckersley, Explainable machine learning in deployment, in *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT/FAT*)*, Barcelona, Spain (2020), pp. 648–657

[C+15] F. Chollet et al., Keras (2015). Accessed 18 Nov 2021

[Car97] R. Caruana, Multitask learning. Mach. Learn. **28**(1), 41–75 (1997)

[CBG+17] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier, Parseval networks: improving robustness to adversarial examples, in *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, NSW, Australia (2017), pp. 854–863

[CBLR18] Z. Chen, V. Badrinarayanan, C.-Y. Lee, A. Rabinovich, GradNorm: gradient normalization for adaptive loss balancing in deep multitask networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden (2018), pp. 794–803

[CBSW19] L. Caltagirone, M. Bellone, L. Svensson, M. Wahde, LiDAR-camera fusion for road detection using fully convolutional neural networks. Robot. Auton. Syst. **111**, 125–131 (2019)

[CCO98] A.I. Cristea, P.D. Cristea, T. Okamoto, N.N.K. Extraction, Revue Roumaine des Sciences Techniques. Série Électrotechnique et Énergétique **43**(1), 1–14 (1998)

[CDH+16] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, InfoGAN: interpretable representation learning by information maximizing generative adversarial nets, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Barcelona, Spain (2016), pp. 2172–2180

[CDS90] Y.L. Cun, J.S. Denker, S.A. Solla, Optimal brain damage, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Denver, CO, USA (1990), pp. 598–605

[CE17] R. Cotterell, J. Eisner, Probabilistic typology: deep generative models of vowel inventories (2017), pp. 1–11. arXiv:1705.01684

[CEL20] Y. Choi, M. El-Khamy, J. Lee, Universal deep neural network compression. IEEE J. Select. Topics Signal Proc. **14**(4), 715–726 (2020)

[CH19] J. Clark, G.K. Hadfield, Regulatory markets for AI safety (2019), pp. 1–23. arXiv:2001.00078

[CHM+15] A. Choromanska, M. Henaff, M. Mathieu, G.B. Arous, Y. LeCun, The loss surfaces of multilayer networks, in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, San Diego, CA, USA (2015), pp. 192–204

[CL18] J. Choo, S. Liu, Visual analytics for explainable deep learning (2018), pp. 1–10. arXiv:1804.02527

[CLC+19] H.-Y. Chen, J.-H. Liang, S.-C. Chang, J.-Y. Pan, Y.-T. Chen, W. Wei, D.-C. Juan, Improving adversarial robustness via guided complement entropy, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea (2019), pp. 4881–4889

[CLG+15] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, N. Elhadad, Intelligible models for healthcare: predicting pneumonia risk and hospital 30-day readmission, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Sydney, NSW, Australia (2015), pp. 1721–1730

[CLLW19] P.-Y. Chen, A.H. Liu, Y.-C. Liu, Y.-C.F. Wang, Towards scene understanding: unsupervised monocular depth estimation with semantic-aware representation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 2624–2632

[CLS+18] Y. Chen, W. Li, C. Sakaridis, D. Dai, L. Van Gool, Domain adaptive faster R-CNN for object detection in the wild, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 3339–3348

[CNH+18] C.-H. Cheng, G. Nührenberg, C.-H. Huang, H. Ruess, H. Yasuoka, Towards dependability metrics for neural networks, in *Proceedings of the ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, Beijing, China (2018), pp. 43–46

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The Cityscapes dataset for semantic urban scene understanding, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA (2016), pp. 3213–3223

[CPK+18] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **40**(4), 834–848 (2018)

[CPMA19] V. Casser, S. Pirk, R. Mahjourian, A. Angelova, Depth prediction without the sensors: leveraging structure for unsupervised learning from monocular videos, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA (2019), pp. 8001–8008

[CPSA17] L.-C. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking Atrous convolution for semantic image segmentation (2017), pp. 1–14. arXiv:1706.05587

[Csu17] G. Csurka, Domain adaptation for visual applications: a comprehensive survey, in *Domain Adaptation in Computer Vision Applications*, ed. by G. Csurka (Springer, Berlin, 2017), pp. 1–35

[CURG22] R. Chan, S. Uhlemeyer, M. Rottmann, H. Gottschalk, Detecting and learning the unknown in semantic segmentation, in *Deep Neural Networks and Data for Automated Driving – Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 307–344

[CW17a] N. Carlini, D. Wagner, Adversarial examples are not easily detected: bypassing ten detection methods, in *Proceedings of the ACM Workshop on Artificial Intelligence and Security (AISec)*, New York, NY, USA (2017), pp. 3–14

[CW17b] N. Carlini, D.A. Wagner, Towards evaluating the robustness of neural networks, in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA (2017), pp. 39–57

[CWL+19] H.-Y. Chen, P.-H. Wang, C.-H. Liu, S.-C. Chang, J.-Y. Pan, Y.-T. Chen, W. Wei, D.-C. Juan, Complement objective training, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA (2019), pp. 1–10

[CZM+19] E.D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q.V. Le, AutoAugment: learning augmentation strategies from data, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 113–123

[CZSL19] E.D. Cubuk, B. Zoph, J. Shlens, Q.V. Le, RandAugment: practical data augmentation with no separate search (2019), pp. 1–13. arXiv:1909.13719

[CZYS19] F. Cheng, H. Zhang, D. Yuan, M. Sun, Leveraging semantic segmentation with learning-based confidence measure. Neurocomputing **329**, 21–31 (2019)

[DAL+18] G. Dhillon, K. Azizzadenesheli, Z. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, A. Anandkumar, Stochastic activation pruning for robust adversarial defense, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada (2018), pp. 1–13

[DCG+19] S.V. Desai, A.L. Chandra, W. Guo, S. Ninomiya, V.N. Balasubramanian, An adaptive supervision framework for active learning in object detection, in *Proceedings of the British Machine Vision Conference (BMVC)*, Cardiff, UK (2019), pp. 1–13

[DCO+19] Q. Dou, C. Chen, C. Ouyang, H. Chen, P.A. Heng, Unsupervised domain adaptation of ConvNets for medical image segmentation via adversarial learning, in *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*, ed. by L. Le, X. Wang, G. Carneiro, L. Yang (Springer, Berlin, 2019), pp. 93–115

[DdCKG19] Q. Dou, D.C. de Castro, K. Kamnitsas, B. Glocker, Domain generalization via model-agnostic learning of semantic features, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2019), pp. 6447–6458

[DK17] J. Djolonga, A. Krause, Learning implicit generative models using differentiable graph tests (2017), pp. 1–16. arXiv:1709.01006

[DK18] J.J. Dudley, P.O. Kristensson, A review of user interface design for interactive machine learning. ACM Trans. Inter. Intell. Syst. (TIIS) **8**(2), 1–37 (2018)

[DL13] A. Damianou, N. Lawrence, Deep Gaussian processes, in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Scottsdale, AZ, USA (2013), pp. 207–215

[DLP+18] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, Boosting adversarial attacks with momentum, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 9185–9193

[DMW+20] R. Duan, X. Ma, Y. Wang, J. Bailey, A.K. Qin, Y. Yang, Adversarial camouflage: hiding physical-world attacks with natural styles, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Virtual Conference (2020), pp. 1000–1008

[Doe16] C. Doersch, Tutorial on variational autoencoders (2016), pp. 1–23. arXiv:1606.05908

[DRC+17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: an open urban driving simulator, in *Proceedings of the Conference on Robot Learning CORL*, Mountain View, CA, USA (2017), pp. 1–16

[DT17] T. DeVries, G.W. Taylor, Dataset augmentation in feature space, in *Proceedings of the International Conference on Learning Representations (ICLR) Workshops*, Toulon, France (2017), pp. 1–12

[DT18] T. DeVries, G.W. Taylor, Learning confidence for out-of-distribution detection in neural networks (2018), pp. 1–12. arXiv:1802.04865

[EEF+18] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song, Robust physical-world attacks on deep learning visual classification, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 1625–1634

[EIS+19] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, B. Tran, A. Madry, Adversarial robustness as a prior for learned representations (2019), pp. 1–25. arXiv:1906.00945

[EMH19a] T. Elsken, J.H. Metzen, F. Hutter, Efficient multi-objective neural architecture search via lamarckian evolution, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA (2019), pp. 1–23

[EMH19b] T. Elsken, J.H. Metzen, F. Hutter, Neural architecture search: a survey. J. Mach. Learn. Res. **20**(55), 1–21 (2019)

[ERT+17] A. Endert, W. Ribarsky, C. Turkay, W. Wong, I. Nabney, I. Díaz Blanco, F. Rossi, The state of the art in integrating machine learning into visual analytics. Comput. Graph. Forum **36**(8), 458–486 (2017)

[ETTS19] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, A rotation and a translation suffice: fooling CNNs with simple transformations, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA (2019), pp. 1–21

[EW18] C. Eastwood, C.K.I. Williams, A framework for the quantitative evaluation of disentangled representations, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada (2018), pp. 1–15

[FAA18] E. Fertig, A. Arbabi, A.A. Alemi, β-VAEs can retain label information even at high compression (2018), pp. 1–6. arXiv:1812.02682

[FAL17] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, NSW, Australia (2017), pp. 1126–1135

[FCSG17] R. Feinman, R.R. Curtin, S. Shintre, A.B. Gardner, Detecting adversarial samples from artifacts (2017), pp. 1–9. arXiv:1703.00410

[FF15] A. Fawzi, P. Frossard, Manitest: are classifiers really invariant? (2015), pp. 1–13. arXiv:1507.06535

[FH19] M. Feurer, F. Hutter, Hyperparameter optimization, in *Automated Machine Learning: Methods, Systems, Challenges* (Springer, Berlin, 2019), pp. 3–33

[FKH18] S. Falkner, A. Klein, F. Hutter, BOHB: robust and efficient hyperparameter optimization at scale, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden (2018), pp. 1436–1445

[FP78] D.F. Frey, R.A. Pimentel, Principal component analysis and factor analysis, in *Quantitative Ethology*, ed. by P.W. Colgan (Wiley, New York, 1978), pp. 219–245

[Fry77] M.J. Fryer, A review of some non-parametric methods of density estimation. J. Inst. Math. Appl. **20**(3), 335–354 (1977)

[FSS+16] M. Fayyaz, M.H. Saffar, M. Sabokrou, M. Fathy, R. Klette, F. Huang, STFCN: Spatio-temporal FCN for semantic video segmentation (2016), pp. 1–17. arXiv:1608.05971

[FV17] R.C. Fong, A. Vedaldi, Interpretable explanations of black boxes by meaningful perturbation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy (2017), pp. 3429–3437

[FXY12] J. Feng, H. Xu, S. Yan, Robust PCA in high-dimension: a deterministic approach (2012), pp. 1–8. arXiv:1206.4628

[FZ10] P.F. Felzenszwalb, R. Zabih, Dynamic programming and graph algorithms in computer vision. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **33**(4), 721–740 (2010)

[GAHW22] S.S. Gannamaneni, M. Akila, C. Heinzemann, M. Woehrle, The good and the bad: using neuron coverage as a DNN validation technique, in *Deep Neural Networks and Data for Automated Driving – Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 413–433

[GBA+19] Y.F.A. Gaus, N. Bhowmik, S. Akçay, P.M. Guillén-Garcia, J.W. Barker, T.P. Breckon, Evaluation of a dual convolutional neural network architecture for object-wise anomaly detection in cluttered x-ray security imagery, in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary (2019), pp. 1–8

[GBR+06] A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, A.J. Smola, A kernel method for the two-sample-problem, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2006), pp. 513–520

[GBR07] T. Gneiting, F. Balabdaoui, A.E. Raftery, P. Forecasts, Calibration and sharpness. J. R. Stat. Soc.: Ser. B (Stat. Methodol.) **69**(2), 243–268 (2007)

[GBR+12] A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, A.J. Smola, A kernel two-sample test. J. Mach. Learn. Res. **13**(25), 723–773 (2012)

[GBY+18] L.H. Gilpin, D. Bau, B.Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining explanations: an overview of interpretability of machine learning, in *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Turin, Italy (2018), pp. 80–89

[GEB15] L.A. Gatys, A.S. Ecker, M. Bethge, A neural algorithm of artistic style (2015), pp. 1–16. arXiv:1508.06576

[GEB16] L.A. Gatys, A.S. Ecker, M. Bethge, Image style transfer using convolutional neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA (2016), pp. 2414–2423

[GG84] S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **6**(6), 721–741 (1984)

[GG16] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, in *Proceedings of the International Conference on Machine Learning (ICML)*, New York, NY, USA (2016), pp. 1050–1059

[GHHW20] C. Gladisch, C. Heinzemann, M. Herrmann, M. Woehrle, Leveraging combinatorial testing for safety-critical computer vision datasets, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2020), pp. 1314–1321. Virtual Conference

[GHK17] Y. Gal, J. Hron, A. Kendall, Concrete dropout, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Long Beach, CA, USA (2017), pp. 3581–3590

[GHL+20] V. Guizilini, R. Hou, J. Li, R. Ambrus, A. Gaidon, Semantically-guided representation learning for self-supervised monocular depth, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2020), pp. 1–14. Virtual Conference

[GHS22] O. Grau, K. Hagn, Q.S. Sha, A variational deep synthesis approach for perception validation, in *Deep Neural Networks and Data for Automated Driving – Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 389–412

[GIG17] Y. Gal, R. Islam, Z. Ghahramani, Deep Bayesian active learning with image data, in *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, NSW, Australia (2017), pp. 1183–1192

[Gir15] R. Girshick, Fast R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile (2015), pp. 1440–1448

[GJG17] R. Gadde, V. Jampani, P.V. Gehler, Semantic video CNNs through representation warping, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy (2017), pp. 4453–4462

[GJZ+18] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, J. Sun, DLFuzz: differential fuzzing testing of deep learning systems, in *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on Foundations of Software Engineering (ESEC/FSE)*, Lake Buena Vista, FL, USA (2018), pp. 739–743

[GK20] T. Ghosh, M. Kirby, Supervised dimensionality reduction and visualization using centroid-encoder (2020), pp. 1–25. arXiv:2002.11934

[GKZ14] M. Ghifary, W.B. Kleijn, M. Zhang, Domain adaptive neural networks for object recognition, in *Proceedings of the Pacific Rim International Conference on Artificial Intelligence (PRICA) – Trends in Artificial Intelligence*, Gold Coast, QLD, Australia (2014), pp. 898–904

[GKZB15] M. Ghifary, W.B. Kleijn, M. Zhang, D. Balduzzi, Domain generalization for object recognition with multi-task autoencoders, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile (2015), pp. 2551–2559

[GLYB14] Y. Gong, L. Liu, M. Yang, L.D. Bourdev, Compressing deep convolutional networks using vector quantization (2014), pp. 1–10. arXiv:1412.6115

[Goo17] I. Goodfellow, NIPS 2016 tutorial: generative adversarial networks (2017), pp. 1–57. arXiv:1701.00160

[GPAM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (2014), pp. 2672–2680

[GPSW17] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, NSW, Australia (2017), pp. 1321–1330

[GR19] P. Gupta, E. Rahtu, CIIDefence: defeating adversarial attacks by fusing class-specific image inpainting and image denoising, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea (2019), pp. 6708–6717

[Gra18] A. Gramacki, *Nonparametric Kernel Density Estimation and its Computational Aspects* (Springer, Berlin, 2018)

[GRCvdM18] C. Guo, M. Rana, M. Cissé, L. van der Maaten, Countering adversarial images using input transformations, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada (2018), pp. 1–12

[GRS22] H. Gottschalk, M. Rottmann, M. Saltagic, Does redundancy in AI perception systems help to test for super-human automated driving performance?, in *Deep Neural Networks and Data for Automated Driving—Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 103–128

[GSS15] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA (2015), pp. 1–11

[GTC+18] R. Garcia, A.C. Telea, B.C. da Silva, J. Tørresen, J.L. Dihl Comba, A task-and-technique centered survey on visual analytics for deep learning model engineering. Comput. & Graph. **77**, 30–49 (2018)

[Guo18] Y. Guo, A survey on methods and theories of quantized neural networks (2018), pp. 1–17. arXiv:1808.04752

[GWY+19] S. Gui, H. Wang, H. Yang, C. Yu, Z. Wang, J. Liu, Model compression with adversarial robustness: a unified optimization framework, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2019), pp. 1283–1294

[HAB19] M. Hein, M. Andriushchenko, J. Bitterwolf, Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 41–50

[Hai16] T. Hailesilassie, Rule extraction algorithm for deep neural networks: a review (2016), pp. 1–6. arXiv:1610.05267

[HAMFs22] A.S. Hashemi, B. Andreas, S. Mozaffari, T. Fingscheidt, Improving transferability of generated universal adversarial perturbations for image classification and segmentation, in *Deep Neural Networks and Data for Automated Driving—Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 195–222

[HAMS20] T.M. Hospedales, A. Antoniou, P. Micaelli, A.J. Storkey, Meta-learning in neural networks: a survey (2020), pp. 1–20. arXiv:2004.05439

[HD19] D. Hendrycks, T. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA (2019), pp. 1–15

[HDHH20] J. Hanhirova, A. Debner, M. Hyyppä, V. Hirvisalo, A machine learning environment for evaluating autonomous driving software (2020), pp. 1–8. arXiv:2003.03576

[HDR18] M. Harradon, J. Druce, B. Ruttenberg, Causal learning and explanation of deep neural networks via autoencoded activations (2018), pp. 1–8. arXiv:1802.00541

[HDY+12] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Proc. Mag. **29**(6), 82–97 (2012)

[HG17] D. Hendrycks, K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France (2017), pp. 1–12

[HG22] K. Hagn, O. Grau, Optimized data synthesis for DNN training and validation by sensor artifact simulation, in *Deep Neural Networks and Data for Automated Driving—Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 149–170

[HGDG17] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy (2017), pp. 2980–2988

[Hin09] G.E. Hinton, Deep belief networks. Scholarpedia **4**(5), 5947 (2009)

[HKD+18] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, in *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, Stockholm, Sweden (2018), pp. 2234–2240

[HKPC18] F. Hohman, M. Kahng, R. Pienta, D.H. Chau, Visual analytics in deep learning: an interrogative survey for the next frontiers (2018), pp. 1–20. arXiv:1801.06889

[HLP+17] G. Huang, Y. Li, G. Pleiss, Z. Liu, J.E. Hopcroft, K.Q. Weinberger, Snapshot ensembles: train 1, get M for free (2017), pp. 1–14. arXiv:1704.00109

[HLS+19] D. Ho, E. Liang, I. Stoica, P. Abbeel, X. Chen, Population based augmentation: efficient learning of augmentation policy schedules, in *Proceedings of the International Conference on Machine Learning (ICML)*, Long Beach, CA, USA (2019), pp. 2731–2741

[HMD16] S. Han, H. Mao, W.J. Dally, Deep compression: compressing deep neural network with pruning, trained quantization and huffman coding, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2016), pp. 1–14

[HMD19] D. Hendrycks, M. Mazeika, T. Dietterich, Deep anomaly detection with outlier exposure, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA (2019), pp. 1–18

[HMJ13] T. Hazan, S. Maji, T. Jaakkola, On sampling from the Gibbs distribution with random maximum a-posteriori perturbations, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Lake Tahoe, NV, USA (2013), pp. 1268–1276

[HMK+19] L. Hoyer, M. Muñoz, P. Katiyar, A. Khoreva, V. Fischer, Grid saliency for context explanations of semantic segmentation, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2019), pp. 6459–6470

[HMP+17] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M.M. Botvinick, S. Mohamed, A. Lerchner, β-VAE: learning basic visual concepts with a constrained variational framework, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France (2017), pp. 1–22

[HO00] A. Hyvärinen, E. Oja, Independent component analysis: algorithms and applications. Neural Netw. **13**(4–5), 411–430 (2000). (June)

[HR90] J. Huber, A. Rüppel, Zuverlässigkeitsschätzung für die Ausgangssymbole von Trellis-Decodern. Archiv für Elektronik und Übertragung (AEÜ) (in German), **44**(1), 8–21 (1990)

[HRF19] Z. He, A.S. Rakin, D. Fan, Parametric noise injection: trainable randomness to improve deep neural network robustness against adversarial attack, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 588–597

[HS97] S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

[HVD14] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops*, Montréal, QC, Canada (2014), pp. 1–9

[HZRS15] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **37**(9), 1904–1916 (2015)

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA (2016), pp. 770–778

[HZS17] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy (2017), pp. 1398–1406

[ICG+18] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, T. Brox, Uncertainty estimates and multi-hypotheses networks for optical flow, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany (2018), pp. 652–667

[Ino18] H. Inoue, Data augmentation by pairing samples for images classification (2018), pp. 1–8. arXiv:1801.02929

[ISO18] ISO/TC 22/SC 32. ISO 26262-1:Road Vehicles – Functional Safety – Part 1: Vocabulary. International Organization for Standardization (ISO) (2018)

[JC16] I.T. Jolliffe, J. Cadima, Principal component analysis: a review and recent developments. Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. **374**(2065), 1–16 (2016)

[JC17] K. Janocha, W.M. Czarnecki, On loss functions for deep neural networks in classification. Schedae Informaticae **25**(9), 49–49 (2017)

[JDCR12] M. Joshi, M. Dredze, W.W. Cohen, C.P. Rosé, Multi-domain learning: when do domains matter?, in *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Jeju Island, Korea (2012), pp. 1302–1312

[JGST19] H.S. Jomaa, J. Grabocka, L. Schmidt-Thieme, Hyp-RL: hyperparameter optimization by reinforcement learning (2019), pp. 1–17. arXiv:1906.11527

[JKC+18] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A.G. Howard, H. Adam, D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 2704–2713

[JLM+18] B. Jiang, R. Luo, J. Mao, T. Xiao, Y. Jiang, Acquisition of localization confidence for accurate object detection, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany (2018), pp. 784–799

[JLS+20] Y. Jia, Y. Lu, J. Shen, Q.A. Chen, H. Chen, Z. Zhong, T. Wei, Fooling detection alone is not enough: adversarial attack against multiple object tracking, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2020), pp. 1–11. Virtual Conference

[JMS96] M.C. Jones, J.S. Marron, S.J. Sheather, A brief survey of bandwidth selection for density estimation. J. Amer. Stat. Assoc. **91**(433), 401–407 (1996)

[JWCF19] X. Jia, X. Wei, X. Cao, H. Foroosh, ComDefend: an efficient image compression model to defend adversarial examples, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 6084–6092

[KAF+08] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, G. Melançon, Visual analytics: definition, process, and challenges, in *Information Visualization: Human-Centered Issues and Perspectives* (Springer, Berlin, 2008), pp. 154–175

[KALL18] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of GANs for improved quality, stability, and variation, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada (2018), pp. 1–26

[KB15] D.P. Kingma, J. Ba, ADAM: a method for stochastic optimization, in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA (2015), pp. 1–15

[KBFs20] M. Klingner, A. Bär, T. Fingscheidt, Improved noise and attack robustness for semantic segmentation by using multi-task training with self-supervised depth estimation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2020), pp. 1299–1309. Virtual Conference

[KBL+17] K. Kamnitsas, C. Baumgartner, C. Ledig, V. Newcombe, J. Simpson, A. Kane, D. Menon, A. Nori, A. Criminisi, D. Rueckert, B. Glocker, Unsupervised domain adaptation in brain lesion segmentation with adversarial networks, in *Proceedings of the International Conference on Information Processing in Medical Imaging (IPMI)*, Boone, NC, USA (2017), pp. 597–609

[KFE18] V. Kuleshov, N. Fenner, S. Ermon, Accurate uncertainties for deep learning using calibrated regression, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden (2018), pp. 2801–2809

[KFs22] M. Klingner, T. Fingscheidt, Improved DNN robustness by multi-task training with an auxiliary self-supervised task, in *Deep Neural Networks and Data for Automated Driving—Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 171–194

[KG17] A. Kendall, Y. Gal, What uncertainties do we need in bayesian deep learning for computer vision?, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Long Beach, CA, USA, 2017), pp. 5574–5584

[KGB17a] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, in *Proceedings of the International Conference on Learning Representations (ICLR) Workshops* (Toulon, France 2017), pp. 1–14

[KGB17b] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Toulon, France, 2017), pp. 1–17

[KGC17] J. Kukačka, V. Golkov, D. Cremers, Regularization for deep learning: a taxonomy (2017), pp. 1–23. arXiv:1710.10686

[KGC18] A. Kendall, Y. Gal, R. Cipolla, Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 7482–7491

[KGLL20] Z. Kong, J. Guo, A. Li, C. Liu, PhysGAN: generating physical-world-resilient adversarial examples for autonomous driving, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 14254–14263. Virtual Conference

[KHKS22] F. Küppers, A. Haselhoff, J. Kronenberger, J. Schneider, Confidence calibration for object detection and segmentation, in *Deep Neural Networks and Data for Automated Driving—Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 255–282

[KK11] P. Krähenbühl, V. Koltun, Efficient inference in fully connected CRFs with Gaussian edge potentials, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Granada, Spain (2011), pp. 109–117

[KKB19] P. Koopman, A. Kane, J. Black, Credible autonomy safety argumentation, in *Proceedings of the Safety-Critical Systems Symposium (SSS)*, Bristol, UK (2019), pp. 1–27

[KKMH20] I. Khemakhem, D.P. Kingma, R.P. Monti, A. Hyvärinen, Variational autoencoders and nonlinear ICA: a unifying framework (2020), pp. 1–27. arXiv:1907.04809

[KKSH20] F. Küppers, J. Kronenberger, A. Shantia, A. Haselhoff, Multivariate confidence calibration for object detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2020), pp. 1322–1330. Virtual Conference

[KLSL18] C.C. Kao, T.Y. Lee, P. Sen, M.Y. Liu, Localization-aware active learning for object detection (2018), pp. 1–35. arXiv:1801.05124

[KLX+17] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, X. Wang, Object detection in videos with tubelet proposal networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA (2017), pp. 727–735

[KM02] D. Klein, C.D. Manning, Fast exact inference with a factored model for natural language parsing, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2002), pp. 3–10

[KMAM18] M. Kesarwani, B. Mukhoty, V. Arya, S. Mehta, Model extraction warning in MLaaS paradigm, in *Proceedings of the Annual Computer Security Applications Conference* (2018), pp. 371–380

[KMN+17] N.S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P.T.P. Tang, On large-batch training for deep learning: generalization gap and sharp minima (2017), pp. 1–16. arXiv:1609.04836

[KMT09] D.A. Keim, F. Mansmann, J. Thomas, Visual analytics: how much visualization and how much analytics? ACM SIGKDD Explor. Newslett. **11**(2), 5–8 (2009)

[KNS+18] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, E.P. Xing, Neural architecture search with Bayesian optimisation and optimal transport, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2018), pp. 2020–2029

[Kok17] I. Kokkinos, Ubernet: training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA (2017), pp. 5454–5463

[KP20] A. Kumar, B. Poole, On implicit regularization in β-VAEs, in *Proceedings of the International Conference on Machine Learning (ICML)* (2020), pp. 5480–5490

[KRBT08] P. Kohli, J. Rihan, M. Bray, P.H.S. Torr, Simultaneous segmentation and pose estimation of humans using dynamic graph cuts. Int. J. Comput. Vis. (IJCV) **79**(3), 285–298 (2008)

[Kri09] A. Krizhevsky, Object classification experiments. Technical report, Canadian Institute for Advanced Research (2009)

[KRPM+18] S. Kohl, B. Romera-Paredes, C. Meyer, J. De Fauw, J.R. Ledsam, K. Maier-Hein, S.M.A. Eslami, D. Jimenez Rezende, O. Ronneberger, A probabilistic U-Net for segmentation of ambiguous images, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2018), pp. 6965–6975

[KS18] H. Kumar, P.S. Sastry, Robust loss functions for learning multi-class classifiers, in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Miyazaki, Japan (2018), pp. 1–6

[KSFF17] M. Kull, T. Silva Filho, P. Flach, Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers, in *Proceedings*

of the International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA (2017), pp. 623–631

[KSH12] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Lake Tahoe, NV, USA (2012), pp. 1106–1114

[KT08] P. Kohli, P.H.S. Torr, Measuring uncertainty in graph cut solutions. Comput. Vis. Image Underst. **112**(1), 30–38 (2008)

[KTMFs20] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, T. Fingscheidt, Self-supervised monocular depth estimation: solving the dynamic object problem by semantic guidance, in *Proceedings of the European Conference on Computer Vision (ECCV)* (2020), pp. 582–600. Virtual Conference

[KTP+20] K. Krishna, G.S. Tomar, A.P. Parikh, N. Papernot, M. Iyyer, Thieves on sesame street! Model extraction of BERT-based APIs, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2020), pp. 1–19. Virtual Conference

[KW14] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, AB, Canada (2014), pp. 1–14

[KWG+18] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, R. Sayres, Interpretability beyond feature attribution: quantitative testing with concept activation vectors (TCAV), in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden (2018), pp. 2668–2677

[KXG17] E. Kodirov, T. Xiang, S. Gong, Semantic autoencoder for zero-shot learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA (2017), pp. 4447–4456

[KYL+20] N. Kapoor, C. Yuan, J. Löhdefink, R. Zimmermann, S. Varghese, F. Hüger, N. Schmidt, P. Schlicht, T. Fingscheidt, A self-supervised feature map augmentation (FMA) loss and combined augmentations finetuning to efficiently improve the robustness of CNNs, in *Proceedings of the ACM Computer Science in Cars Symposium (CSCS)* (2020), pp. 1–8. Virtual Conference

[KZG18] D. Karmon, D. Zoran, Y. Goldberg, Lavan: localized and visible adversarial noise, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden (2018), pp. 2507–2515

[LAL+20] C. Liu, T. Arnon, C. Lazarus, C. Barrett, M.J. Kochenderfer, Algorithms for verifying deep neural networks (2020), pp. 1–126. arXiv:1903.06758

[LBD+89] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)

[LBL+19] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, O. Bachem, Challenging common assumptions in the unsupervised learning of disentangled representations, in *Proceedings of the International Conference on Machine Learning (ICML)*, Long Beach, CA, USA (2019), pp. 4114–4124

[LBS+19] J. Löhdefink, A. Bär, N.M. Schmidt, F. Hüger, P. Schlicht, T. Fingscheidt, On low-bitrate image compression for distributed automotive perception: higher peak SNR does not mean better semantic segmentation, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, Paris, France (2019), pp. 352–359

[LCH+07] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F.J. Huang, Energy-based models, in *Predicting Structured Data*, ed. by G. Baklr, T. Hofmann, B. Schölkopf, A.J. Smola, B. Taskar, S.V.N. Vishwanathan (MIT Press, 2007), pp. 191–246

[LCM+17] L. Junhua, W. Chen, Y. Ma, J. Ke, Z. Li, F. Zhang, R. Maciejewski, Recent progress and trends in predictive visual analytics. Front. Comput. Sci. **11**(2), 192–207 (2017)

[LCPB18] S. Li, Y. Chen, Y. Peng, L. Bai, Learning more robust features with adversarial training (2018), pp. 1–7. arXiv:1804.07757

[LCWJ18] M. Long, Z. Cao, J. Wang, M.I. Jordan, Conditional adversarial domain adaptation, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2018), pp. 1640–1650

[LCY14] M. Lin, Q. Chen, S. Yan, Network in network, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, AB, Canada (2014), pp. 1–10

[LFK+20] J. Löhdefink, J. Fehrling, M. Klingner, F. Hüger, P. Schlicht, N.M. Schmidt, T. Fingscheidt, Self-supervised domain mismatch estimation for autonomous perception, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2020), pp. 1359–1368. Virtual Conference

[LGG+17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy (2017), pp. 2980–2988

[LGH+17] L. Yafeng, R. Garcia, B. Hansen, M. Gleicher, R. Maciejewski, The state-of-the-art in predictive visual analytics. Comput. Graph. Forum **36**(3), 539–562 (2017)

[LGH19] J. Lin, C. Gan, S. Han, Defensive quantization: when efficiency meets robustness, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA (2019), pp. 1–15

[LJD+18] L. Li, K.G. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: a novel bandit-based approach to hyperparameter optimization. J. Mach. Learn. Res. **18**(185), 1–52 (2018)

[LJD19] S. Liu, E. Johns, A.J. Davison, End-to-end multi-task learning with attention, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 1871–1880

[LJL+19] H. Liu, R. Ji, J. Li, B. Zhang, Y. Gao, Y. Wu, F. Huang, Universal adversarial perturbation via prior driven uncertainty approximation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea (2019), pp. 2941–2949

[LK19] M. Lee, Z. Kolter, On physical adversarial patches for object detection (2019), pp. 1–5. arXiv:1906.11897

[LKD+17] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient ConvNets, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France (2017), pp. 1–13

[LLL+17] Z. Liu, X. Li, P. Luo, C.C. Loy, X. Tang, Deep learning markov random field for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **40**(8), 1814–1828 (2017)

[LLL+19] Z. Liu, Q. Liu, T. Liu, N. Xu, X. Lin, Y. Wang, W. Wen, Feature distillation: DNN-oriented JPEG compression against adversarial examples, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 860–868

[LLLS17] K. Lee, H. Lee, K. Lee, J. Shin, Training confidence-calibrated classifiers for detecting out-of-distribution samples (2017), pp. 1–16. arXiv:1711.09325

[LLLS18] K. Lee, K. Lee, H. Lee, J. Shin, A simple unified framework for detecting out-of-distribution samples and adversarial attacks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2018), pp. 7167–7177

[LLS+17] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy (2017), pp. 2755–2763

[LLS18] S. Liang, Y. Li, R. Srikant, Enhancing the reliability of out-of-distribution image detection in neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada (2018), pp. 1–15

[LM19] Z.G. Liu, M. Mattina, Learning low-precision neural networks without straight-through estimator (STE), in *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA (2019), pp. 3066–3072

[LMW+17] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, V. Pascucci, Visualizing high-dimensional data: advances in the past decade. IEEE Trans. Vis. Comput. Graph. **23**(3), 1249–1268 (2017)

[LNH14] C.H. Lampert, H. Nickisch, S. Harmeling, Attribute-based classification for zero-shot visual object categorization. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **36**(3), 453–465 (2014)

[LO17] D. Lopez-Paz, M. Oquab, Revisiting classifier two-sample tests, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France (2017), pp. 1–15

[LPB17] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Long Beach, CA, USA (2017), pp. 6402–6413

[LPWK18] H. Li, S.J. Pan, S. Wang, A.C. Kot, Domain generalization with adversarial feature learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 5400–5409

[LSD15] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA (2015), pp. 3431–3440

[LSFs19] T. Lohrenz, M. Strake, T. Fingscheidt, On temporal context information for hybrid BLSTM-based phoneme recognition, in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Singapore (2019), pp. 516–523

[LSY19] H. Liu, K. Simonyan, Y. Yang, DARTS: differentiable architecture search, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA (2019), pp. 1–13

[LTG+18] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, D. Tao, Deep domain generalization via conditional invariant adversarial networks, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany (2018), pp. 647–663

[LUAD16] M. Lê, J. Unkelbach, N. Ayache, H. Delingette, Sampling image segmentations for uncertainty quantification. Med. Image Anal. **34**, 42–51 (2016)

[LWB+19] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, K.-R. Müller, Unmasking clever hans predictors and assessing what machines really learn. Nat. Commun. **10**(1096), 1–8 (2019)

[LWL17] J.-H. Luo, J. Wu, W. Lin, ThiNet: a filter level pruning method for deep neural network compression, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy (2017), pp. 5068–5076

[LWLZ17] S. Liu, X. Wang, M. Liu, J. Zhu, Towards better analysis of machine learning models: a visual analytics perspective. Vis. Inf. **1**(1), 48–56 (2017)

[LWZ09] M. Li, W. Yan, Q. Zhang, SAR image segmentation based on mixture context and wavelet hidden-class-label Markov random field. Comput. & Math. Appl. **57**(6), 961–969 (2009)

[LYL+19] X. Liu, H. Yang, Z. Liu, L. Song, H. Li, Y. Chen, DPATCH: an adversarial patch attack on object detectors, in *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI)*, Honolulu, HI, USA (2019), pp. 1–8

[LYP+19] R.G. Lopes, D. Yin, B. Poole, J. Gilmer, E.D. Cubuk, Improving robustness without sacrificing accuracy with patch Gaussian augmentation (2019), pp. 1–18. arXiv:1906.02611

[LYSH18] D. Li, Y. Yang, Y.-Z. Song, T.M. Hospedales, Learning to generalize: meta-learning for domain generalization, in *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA (2018), pp. 3490–3497

[LYW+19] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, A. Yuille, Every pixel counts ++: joint learning of geometry and motion with 3D holistic understanding (2019), pp. 1–17. arXiv:1810.06125

[LYZZ19] P. Li, J. Yi, B. Zhou, L. Zhang, Improving the robustness of deep neural networks via adversarial training with triplet loss, in *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, Macau, China (2019), pp. 2909–2915

[LZG+19] Y. Luo, L. Zheng, T. Guan, J. Yu, Y. Yang, Taking a closer look at domain shift: category-level adversaries for semantics consistent domain adaptation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 2507–2516

[LZN+18] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, K. Murphy, Progressive neural architecture search, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany (2018), pp. 19–35

[LZWJ17] M. Long, H. Zhu, J. Wang, M.I. Jordan, Deep transfer learning with joint adaptation networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, NSW, Australia (2017), pp. 2208–2217

[LZY+19] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, T.M. Hospedales, Episodic training for domain generalization, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea (2019), pp. 1446–1455

[Mac98] D.J.C. MacKay, Introduction to Gaussian processes, in *Neural Networks and Machine Learning*. ed. by C.M. Bishop (Springer, Berlin, 1998), pp. 133–166

[Mac03] D.J.C. MacKay, *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, Cambridge, 2003)

[MARC20] M. Mancini, Z. Akata, E. Ricci, B. Caputo, Towards recognizing unseen categories in unseen domains, in *Proceedings of the European Conference on Computer Vision (ECCV)* (2020), pp. 466–483. Virtual Conference

[MBRB18] A.S. Morcos, D.G.T. Barrett, N.C. Rabinowitz, M. Botvinick, On the importance of single directions for generalization (2018), pp. 1–15. arXiv:1803.06959

[MBS13] K. Muandet, D. Balduzzi, B. Schölkopf, Domain generalization via invariant feature representation, in *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, GA, USA (2013), pp. 10–18

[MCKBF17] J.H. Metzen, M. Chaithanya Kumar, T. Brox, V. Fischer, Universal adversarial perturbations against semantic image segmentation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy (2017), pp. 2774–2783

[MD18] W. Mei, W. Deng, D.V.D. Adaptation, A Survey. Neurocomputing **312**, 135–153 (2018)

[MDFF16] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, DeepFool: a simple and accurate method to fool deep neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA (2016), pp. 2574–2582

[MDFFF17] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, Universal adversarial perturbations, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA (2017), pp. 1765–1773

[MDFUF19] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, P. Frossard, Robustness via curvature regularization, and vice versa, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 9078–9086

[MDS+18] M. Mathew, K. Desappan, P.K. Swami, S. Nagori, B.M. Gopinath, Embedded low-power deep learning with TIDL. Technical Report, Texas Instruments (2018)

[MDSN17] M. Mathew, K. Desappan, P.K. Swami, S. Nagori, Sparse, quantized, full frame CNN for low power embedded devices, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Honolulu, HI, USA (2017), pp. 328–336

[MG18] A. Malinin, M. Gales, Predictive uncertainty estimation via prior networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2018), pp. 7047–7058

[MGB17] K.R. Mopuri, U. Garg, R.V. Babu, Fast feature fool: a data independent approach to universal adversarial perturbations, in *Proceedings of the British Machine Vision Conference (BMVC)*, London, UK (2017), pp. 1–12

[MGR+18] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, L. van der Maaten, Exploring the limits of weakly supervised pretraining, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany (2018), pp. 185–201

[MH20a] T. Matsuura, T. Harada, Domain generalization using a mixture of multiple latent domains, in *Proceedings of the AAAI Conference on Artificial Intelligence*, New York, NY, USA (2020), pp. 11749–11756

[MH20b] A. Meinke, M. Hein, Towards neural networks that provably know when they don't know, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2020), pp. 1–18. Virtual Conference

[Mig17] S. Migacz, 8-Bit Inference With TensorRT (2017)

[MKH+19] A. Mustafa, S. Khan, M. Hayat, R. Goecke, J. Shen, L. Shao, Adversarial defense by restricting the hidden space of deep neural networks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea (2019), pp. 3385–3394

[ML11] R. Miotto, G. Lanckriet, A generative context model for semantic music annotation and retrieval. IEEE/ACM Trans. Audio Speech Lang. Proc. **20**(4), 1096–1108 (2011)

[MLB+17] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, K.-R. Müller, Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recogn. **65**(5), 211–222 (2017)

[MLG+18] R. Mackowiak, P. Lenz, O. Ghori, F. Diego, O. Lange, C. Rother, CEREALS – cost-effective region-based active learning for semantic segmentation, in *Proceedings of the British Machine Vision Conference (BMVC)*, Newcastle, UK (2018), pp. 1–21

[MMR+17] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, B. Agüera y Arcas, Communication-efficient learning of deep networks from decentralized data (2017), pp. 1–11. arXiv:1602.05629

[MMS+18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada (2018), pp. 1–10

[MRG20] K. Maag, M. Rottmann, H. Gottschalk, Time-dynamic estimates of the reliability of deep semantic segmentation networks, in *Proceedings of the IEEE International Conference on Tools With Artificial Intelligence (ICTAI)* (2020), pp. 502–509. Virtual Conference

[MSGH16] I. Misra, A. Shrivastava, A. Gupta, M. Hebert, Cross-stitch networks for multi-task learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA (2016), pp. 3994–4003

[MSJ+16] A. Makhzani, J. Shlens, N. Jaitly, I.J. Goodfellow, B. Frey, Adversarial autoencoders, in *Proceedings of the International Conference on Learning Representations (ICLR) Workshops*, San Juan, Puerto Rico (2016), pp. 1–16

[MTK+17] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France (2017), pp. 1–17

[MTRA+12] J. Moreno-Torres, T. Raeder, R. Alaiz, N. Chawla, F. Herrera, A unifying view on dataset shift in classification. Pattern Recogn. **45**(1), 521–530 (2012)

[NC16] M. Naeini, G. Cooper, Binary classifier calibration using an ensemble of near isotonic regression models, in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, Barcelona, Spain (2016), pp. 360–369

[NCH15] M.P. Naeini, G. Cooper, M. Hauskrecht, Obtaining well calibrated probabilities using bayesian binning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Austin, TX, USA (2015), pp. 2901–2907

[NK17] N. Narodytska, S. Kasiviswanathan, Simple black-box adversarial attacks on deep neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Honolulu, HI, USA (2017), pp. 1310–1318

[NS18] D. Nilsson, C. Sminchisescu, Semantic video segmentation by gated recurrent flow propagation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 6819–6828

[NSO20] V. Nguyen, S. Schulze, M.A. Osborne, Bayesian optimization for iterative learning, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (2020), pp. 9361–9371. Virtual Conference

[NWC+11] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops*, Granada, Spain (2011), pp. 1–9

[NYC15] A. Nguyen, J. Yosinski, J. Clune, Deep neural networks are easily fooled: high confidence predictions for unrecognizable images, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA (2015), pp. 427–436

[OAFS18] S.J. Oh, M. Augustin, M. Fritz, B. Schiele, Towards reverse-engineering black-box neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada (2018), pp. 1–20

[OE18] A. Oussidi, A. Elhassouny, Deep generative models: survey, in *Proceedings of the IEEE International Conference on Intelligent Systems and Computer Vision (ISCV)*, Fez, Morocco (2018), pp. 1–8

[OOAG19] A. Odena, C. Olsson, D. Andersen, I. Goodfellow, TensorFuzz: debugging neural networks with coverage-guided fuzzing, in *Proceedings of the International Conference on Machine Learning (ICML)*, Long Beach, CA, USA (2019), pp. 4901–4911

[ORF20] P. Oberdiek, M. Rottmann, G.A. Fink, Detection and retrieval of out-of-distribution objects in semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 1331–1340. Virtual Conference

[ORG18] P. Oberdiek, M. Rottmann, H. Gottschalk, Classification uncertainty of deep neural networks based on gradient information, in *Proceedings of the IAPR TC3 Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR)*, Siena, Italy (2018), pp. 113–125

[Osb16] I. Osband, Risk versus uncertainty in deep learning: bayes, bootstrap and the dangers of dropout, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops*, Barcelona, Spain (2016), pp. 1–5

[OSM19] H. Okamoto, M. Suzuki, Y. Matsuo, Out-of-distribution detection using layerwise uncertainty in deep neural networks (2019). Accessed 18 Nov 2021

[PCYJ19] K. Pei, Y. Cao, J. Yang, S. Jana, DeepXplore: automated whitebox testing of deep learning systems. Commun. ACM **62**(11), 137–145 (2019)

[PDZ18] T. Pang, C. Du, J. Zhu, Max-Mahalanobis linear discriminant analysis networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden (2018), pp. 4016–4025

[PFC+19] J. Postels, F. Ferroni, H. Coskun, N. Navab, F. Tombari, Sampling-free epistemic uncertainty estimation using approximated variance propagation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea (2019), pp. 2931–2940

[PGZ+18] H. Pham, M.Y. Guan, B. Zoph, Q.V. Le, J. Dean, Efficient neural architecture search via parameter sharing, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden (2018), pp. 4092–4101

[PHW22] S. Pavlitskaya, C. Hubschneider, M. Weber, Evaluating mixture-of-expert architectures for network aggregation, in *Deep Neural Networks and Data for Automated Driving - Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 345–364

[PKGB18] O. Poursaeed, I. Katsman, B. Gao, S. Belongie, Generative adversarial perturbations, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 4422–4431

[PL20] A. Pati, A. Lerch, Attribute-based regularization of VAE latent spaces (2020), pp. 1–15. arXiv:2004.05485

[Pla99] John Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in *Advances in Large Margin Classifiers*. ed. by A.J. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (MIT Press, Cambridge, 1999), pp. 61–74

[PMG+17] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z.B. Celik, A. Swami, Practical black-box attacks against machine learning, in *Proceedings of the ACM ASIA Conference on Computer and Communications Security (ASIACSS)*, Abu Dhabi, United Arab Emirates (2017), pp. 506–519

[PTC+17] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, G. Hinton, Regularizing neural networks by penalizing confident output distributions (2017), pp. 1–12. arXiv:1701.06548

[PXD+20] T. Pang, K. Xu, Y. Dong, C. Du, N. Chen, J. Zhu, Rethinking softmax cross-entropy loss for adversarial robustness, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2020), pp. 1–19. Virtual Conference

[PY11] G. Papandreou, A.L. Yuille, Perturb-and-map random fields: using discrete optimization to learn and sample from energy models, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain (2011), pp. 193–200

[RA17] M. Rahmani, G. Atia, C. Pursuit, Fast, simple, and robust principal component analysis. IEEE Trans. Signal Proc. **65**(23), 6260–6275 (2017)

[RAHL19] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution for image classifier architecture search, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA (2019), pp. 4780–4789

[Ras03] C.E. Rasmussen, Gaussian processes in machine learning, in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, ed. by O. Bousquet, U. von Luxburg, G. Rätsch (Springer, Berlin, 2003), pp. 63–71

[RBAS18] S. Ruder, J. Bingel, I. Augenstein, A. Søgaard, Latent multi-task architecture learning (2018), pp. 1–8. arXiv:1705.08142

[RBB18] H. Ritter, A. Botev, D. Barber, A scalable laplace approximation for neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada (2018), pp. 1–15

[RBV18] S.-A. Rebuffi, H. Bilen, A. Vedaldi, Efficient parametrization of multi-domain deep neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 8119–8127

[RCH+20] M. Rottmann, P. Colling, T.-P. Hack, R. Chan, F. Hüger, P. Schlicht, H. Gottschalk, Prediction error meta classification in semantic segmentation: detection via aggregated dispersion measures of softmax probabilities, in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (2020), pp. 1–9. Virtual Conference

[RDG18] S. Roychowdhury, M. Diligenti, M. Gori, Image classification using deep learning and prior knowledge, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, London, UK (2018), pp. 336–343

[RDGF15] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA (2015), pp. 779–788

[REBO22] R. Rombach, P. Esser, A. Blattmann, B. Ommer, Invertible neural networks for understanding semantics of invariances of CNN representations, in *Deep Neural Networks and Data for Automated Driving – Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 223–253

[Ree93] R. Reed, Pruning algorithms - a survey. IEEE Trans. Neural Netw. (TNN) **4**(5), 740–747 (1993)

[RKG18] M. Rottmann, K. Kahl, H. Gottschalk, Deep Bayesian active semi-supervised learning, in *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, Orlando, FL, USA (2018), pp. 158–164

[RL19] M. Rahmani, P. Li, Outlier detection and data clustering via innovation search (2019), pp. 1–18. arXiv:1912.12988

[RMC15] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks (2015), pp. 1–16. arXiv:1511.06434

[RMDC19] S.K. Roy, S. Manna, S.R. Dubey, B.B. Chaudhuri, LiSHT: non-parametric linearly scaled hyperbolic tangent activation function for neural networks (2019), pp. 1–11. arXiv:1901.05894

[RRMH17] R. Rasti, H. Rabbani, A. Mehridehnavi, F. Hajizadeh, Macular OCT classification using a multi-scale convolutional neural network ensemble. IEEE Trans. Med. Imag. **37**(4), 1024–1034 (2017)

[RS19] M. Rottmann, M. Schubert, Uncertainty measures and prediction quality rating for the semantic segmentation of nested multi resolution street scene images, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Long Beach, CA, USA (2019), pp. 1361–1369

[RSFD16] E. Rodner, M. Simon, R.B. Fisher, J. Denzler, Fine-grained recognition in the noisy wild: sensitivity analysis of convolutional neural networks approaches, in *Proceedings of the British Machine Vision Conference (BMVC)*, York, UK (2016), pp. 1–13

[RSFM19] E. Raff, J. Sylvester, S. Forsyth, M. McLean, Barrage of random transforms for adversarially robust defense, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 6528–6537

[RSG16] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": explaining the predictions of any classifier, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, New York, NY, USA (2016), pp. 1135–1144

[RSKR22] T. Riedlinger, M. Schubert, K. Kahl, M. Rottmann, Uncertainty quantification for object detection: output- and gradient-based approaches, in *Deep Neural Networks and Data for Automated Driving – Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 283–306

[RSS18] J. Rabold, M. Siebers, U. Schmid, Explaining black-box classifiers with ILP – empowering LIME with aleph to approximate non-linear decisions with relational rules, in *Proceedings of the International Conference on Inductive Logic Programming (ILP)*, Ferrara, Italy (2018), pp. 105–117

[RTG+19] H. Rezatofighi, N. Tsoi, J.Y. Gwak, A. Sadeghian, I. Reid, S. Savarese, Generalized intersection over union: a metric and a loss for bounding box regression, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 658–666

[RUN18] S. Roy, A. Unmesh, V.P. Namboodiri, Deep active learning for object detection, in *Proceedings of the British Machine Vision Conference (BMVC)*, Newcastle, UK (2018), pp. 1–12

[RZL18] P. Ramachandran, B. Zoph, Q.V. Le, Searching for activation functions, in *Proceedings of the International Conference on Learning Representations (ICLR) Workshops*, Vancouver, BC, Canada (2018), pp. 1–13

[SAMG19] T. Sämann, K. Amende, S. Milz, H.-M. Groß, Robust semantic video segmentation through confidence-based feature map warping, in *Proceedings of the ACM Computer Science in Cars Symposium (CSCS)*, Kaiserslautern, Germany (2019), pp. 1–9

[SBMC17] S. Sarkar, A. Bansal, U. Mahbub, R. Chellappa, UPSET and ANGRI: breaking high performance image classifiers (2017), pp. 1–9. arXiv:1707.01159

[SBS12] P. Sprechmann, A.M. Bronstein, G. Sapiro, Learning robust low-rank representations (2012), pp. 1–15. arXiv:1209.6393

[SCD+20] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: visual explanations from deep networks via gradient-based localization. Int. J. Comput. Vis. (IJCV) **128**, 336–359 (2020)

[Sco15] D.W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization* (Wiley, New York, 2015)

[SCW+15] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.K. Wong, W.C. Woo, Convolutional LSTM network: a machine learning approach for precipitation nowcasting, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2015), pp. 802–810

[SDBR14] J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: the all convolutional net (2014), pp. 1–14. arXiv:1412.6806

[SDF+20] M. Strake, B. Defraene, K. Fluyt, W. Tirry, T. Fingscheidt, Fully convolutional recurrent networks for speech enhancement, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), pp. 6674–6678. Virtual Conference

[SDKF19] H. Song, T. Diethe, M. Kull, P. Flach, Distribution calibration for regression, in *Proceedings of the International Conference on Machine Learning (ICML)*, Long Beach, CA, USA (2019), pp. 5897–5906

[SDW+19] D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, D. Marculescu, Single-path NAS: designing hardware-efficient ConvNets in less than 4 hours, in *Proceedings of the Joint European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, Würzburg, Germany (2019), pp. 481–497

[Set10] B. Settles, Active learning literature survey. Technical Report, University of Wisconsin-Madison (2010)

[SFH17] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Long Beach, CA, USA (2017), pp. 3856–3866

[SGK19] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences (2019), pp. 1–9. arXiv:1704.02685

[SGS15] R.K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2015), pp. 2377–2385

[SGSS07] A.J. Smola, A. Gretton, L. Song, B. Schölkopf, A Hilbert space embedding for distributions, in *Proceedings of the International Conference Algorithmic Learning Theory (ALT)*, Sendai, Japan (2007), pp. 13–31

[SH09] R. Salakhutdinov, G. Hinton, Deep Boltzmann machines, in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Clearwater Beach, FL, USA (2009), pp. 448–455

[She04] S.J. Sheather, Density estimation. Stat. Sci. **19**(4), 588–597 (2004)

[SHK+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(56), 1929–1958 (2014)

[SHK+19] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, R. Ashmore, Structural test coverage criteria for deep neural networks. ACM Trans. Embed. Comput. Syst. (TECS) **18**(5s), 1–23 (2019)

[Shn96] B. Shneiderman, The eyes have it: a task by data type taxonomy for information visualizations, in *Proceedings of the IEEE Symposium on Visual Languages*, Boulder, CO, USA (1996), pp. 336–343

[Sil86] B.W. Silverman, *Density Estimation for Statistics and Data Analysis* (Chapman and Hall/CRC Press, London, 1986)

[SIVA17] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, Inception-ResNet and the impact of residual connections on learning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, San Francisco, CA, USA (2017), pp. 4278–4284

[SJG+20] P. Stock, A. Joulin, R. Gribonval, B. Graham, H. Jégou, And the bit goes down: revisiting the qunatization of neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2020), pp. 1–11. Virtual Conference

[SK19] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning. J. Big Data **60**(6), 1–48 (2019). (July)

[SKF18] H. Song, M. Kull, P. Flach, Non-parametric calibration of probabilistic regression (2018), pp. 1–17. arXiv:1806.07690

[SLC19] I. Seck, G. Loosli, S. Canu, L1-norm double backpropagation adversarial defense, in *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Bruges, Belgium (2019), pp. 1–6

[SLJ+15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA (2015), pp. 1–9

[SLY15] K. Sohn, H. Lee, X. Yan, Learning structured output representation using deep conditional generative models, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2015), pp. 3483–3491

[SMK20] N. Somavarapu, C.-Y. Ma, Z. Kira, Frustratingly simple domain generalization via image stylization (2020), pp. 1–15. arXiv:2006.11207

[SMM+17] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean, Outrageously large neural networks: the sparsely-gated mixture-of-experts layer (2017), pp. 1–19. arXiv:1701.06538

[SOF+19] J. Snoek, Y. Ovadia, E. Fertig, B. Lakshminarayanan, S. Nowozin, D. Sculley, J. Dillon, J. Ren, Z. Nado, Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2019), pp. 13969–13980

[SR20] S. Saito, S. Roy, Effects of loss functions and target representations on adversarial robustness, in *Proceedings of the Conference on Machine Learning and Systems (MLSys) Workshops*, Austin, TX, USA (2020), pp. 1–10

[SRHD16] E. Shelhamer, K. Rakelly, J. Hoffman, T. Darrell, Clockwork convnets for video segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, Amsterdam, The Netherlands (2016), pp. 852–868

[SRK20] P. Sorrenson, C. Rother, U. Köthe, Disentanglement by nonlinear ICA with general incompressible-flow networks (GIN) (2020), pp. 1–23. arXiv:2001.04872

[SRL+22] H. Stage, L. Ries, J. Langner, S. Otten, E. Sax, Analysis and comparison of datasets by leveraging data distributions in latent spaces, in *Deep Neural Networks and Data for Automated Driving – Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 129–148

[SS20a] G. Schwalbe, M. Schels, A survey on methods for the safety assurance of machine learning based systems, in *Proceedings of the European Congress on Embedded Real Time Software and Systems (ERTS)*, Toulouse, France (2020), pp. 1–10

[SS20b] G. Schwalbe, M. Schels, Concept enforcement and modularization as methods for the ISO 26262 safety argumentation of neural networks, in *Proceedings of the European Congress on Embedded Real Time Software and Systems (ERTS)*, Toulouse, France (2020), pp. 1–10

[SSH19] S. Seo, P.H. Seo, B. Han, Learning for single-shot confidence calibration in deep neural networks through stochastic inferences, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 9030–9038

[SSH20] T. Sämann, P. Schlicht, F. Hüger, Strategy to increase the safety of a DNN-based perception for HAD systems (2020), pp. 1–14. arXiv:2002.08935

[SSSS17] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA (2017), pp. 3–18

[SSW+17] T. Schlegl, P. Seeböck, S.M. Waldstein, U. Schmidt-Erfurth, G. Langs, Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, in *Proceedings of the International Conference on Information Processing in Medical Imaging (IPMI)*, Boone, NC, USA (2017), pp. 146–157

[SSZ+17] D. Sacha, M. Sedlmair, L. Zhang, J.A. Lee, J. Peltonen, D. Weiskopf, S.C. North, D.A. Keim, W.Y.S.I.W.Y.C. Change, Human-centered machine learning by interactive visualization. Neurocomputing **268**, 164–175 (2017)

[SSZ19] M. Svatos, G. Sourek, F. Zelezny, Revisiting neural-symbolic learning cycle, in *Proceedings of the International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*, Macao, China (2019), pp. 1–6

[STK+17] D. Smilkov, N. Thorat, B. Kim, F.B. Viégas, M. Wattenberg, SmoothGrad: removing noise by adding noise (2007), pp. 1–10. arXiv:1706.03825

[Sto08] A.J. Storkey, When training and test sets are different: characterising learning transfer, in *Dataset Shift in Machine Learning*, ed. by J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, N.D. Lawrence (MIT Press, 2008), pp. 3–28

[STP17] B. Summa, J. Tierny, V. Pascucci, Visualizing the uncertainty of graph-based 2D segmentation with min-path stability. Comput. Graph. Forum **36**(3), 133–143 (2017)

[STY17] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, NSW, Australia (2017), pp. 3319–3328

[SU15] A.G. Schwing, R. Urtasun, Fully connected deep structured networks (2015), pp. 1–10. arXiv:1503.02351

[SVS19] J. Su, D.V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. (TEVC) **23**(5), 828–841 (2019)

[SVZ14] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: visualising image classification models and saliency maps, in *Proceedings of the International Conference on Learning Representations (ICLR) Workshops*, Banff, AB, Canada (2014), pp. 1–8

[SWR+18] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, D. Kroening, Concolic testing for deep neural networks, in *Proceedings of the ACM/IEEE International Conference on Automated Software Engineering (ASE)*, Montpellier, France (2018), pp. 109–119

[SYPS19] V. Sandfort, K. Yan, P. Pickhardt, R. Summers, Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. Sci. Rep. **9**(1–9), 16884 (2019)

[SYZ+21] X. Sun, Z. Yang, C. Zhang, G. Peng, K.-V. Ling, Conditional Gaussian distribution learning for open set recognition (2021), pp. 1–10. arXiv:2003.08823

[SZ15] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA (2015), pp. 1–14

[SZS+14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, AB, Canada (2014), pp. 1–10

[SZS+17] D. Sacha, L. Zhang, M. Sedlmair, J.A. Lee, J. Peltonen, D. Weiskopf, S.C. North, D.A. Keim, Visual interaction with dimensionality reduction: a structured literature analysis. IEEE Trans. Visual. Comput. Graph. **23**(1), 241–250 (2017)

[SZT17] R. Shwartz-Ziv, N. Tishby, Opening the black box of deep neural networks via information (2017), pp. 1–19. arXiv:1703.00810

[SZYP19] H. Sikka, W. Zhong, J. Yin, C. Pehlevan, A closer look at disentangling in *beta*-VAE (2019), pp. 1–8. arXiv:1912.05127

[TA12] D. Tarlow, R.P. Adams, Revisiting uncertainty in graph cut solutions, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, USA (2012), pp. 2440–2447

[TAGD98] A.B. Tickle, R. Andrews, M. Golea, J. Diederich, The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks. IEEE Trans. Neural Netw. (TNN) **9**(6), 1057–1068 (1998)

[Tay06] B.J. Taylor, *Methods and Procedures for the Verification and Validation of Artificial Neural Networks* (Springer, Berlin, 2006)

[TBF+15] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, Learning spatiotemporal features with 3D convolutional networks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile (2015), pp. 4489–4497

[TBGS19] I. Tolstikhin, O. Bousquet, S. Gelly, B. Schoelkopf, Wasserstein auto-encoders (2019), pp. 1–20. arXiv:1711.01558

[TC05] J.J. Thomas, K.A. Cook, *Illuminating the Path: The Research and Development Agenda for Visual Analytics* (IEEE, 2005)

[TCBZ19] R. Theagarajan, M. Chen, B. Bhanu, J. Zhang, ShieldNets: defending against adversarial attacks using probabilistic adversarial robustness, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 6988–6986

[Thr95] S. Thrun, Extracting rules from artificial neural networks with distributed representations, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Denver, CO, USA 1995, pp. 505–512

[Thr02] S. Thrun, Robotic mapping: a survey. Technical report, Carnegie Mellon University (2002)

[TKTH18] L. Theis, I. Korshunova, A. Tejani, F. Huszár, Faster gaze prediction with dense networks and fisher pruning (2018), pp. 1–18. arXiv:1801.05787

[TPJR18] Y. Tian, K. Pei, S. Jana, B. Ray, DeepTest: automated testing of deep-neural-network-driven autonomous cars, in *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)* (2018), pp. 303–314

[TVRG19] S. Thys, W. Van Ranst, T. Goedemé, Fooling automated surveillance cameras: adversarial patches to attack person detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Long Beach, CA, USA (2019), pp. 49–55

[TWZ+18] M. Teichmann, M. Weber, J.M. Zollner, R. Cipolla, R. Urtasun, MultiNet: real-time joint semantic reasoning for autonomous driving, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China (2018), pp. 1013–1020

[TZJ+16] F. Tramèr, F. Zhang, A. Juels, M.K. Reiter, T. Ristenpart, Stealing machine learning models via prediction APIs, in *Proceedings of the USENIX Security Symposium*, Austin, TX, USA (2016), pp. 601–618

[UEKH19] H. Uzunova, J. Ehrhardt, T. Kepp, H. Handels, Interpretable explanations of black box classifiers applied on medical images by meaningful perturbations using variational autoencoders, in *Proceedings of the SPIE Medical Imaging*, San Diego, CA, USA (2019). 1094911

[UMY+20] S. Uhlich, L. Mauch, K. Yoshiyama, F. Cardinaux, J.A. García, S. Tiedemann, T. Kemp, A. Nakamura, Differentiable quantization of deep neural networks (2020), pp. 1–21. arXiv:1905.11452

[VBB+20] S. Varghese, Y. Bayzidi, A. Bär, N. Kapoor, S. Lahiri, J.D. Schneider, N. Schmidt, F. Hüger, P. Schlicht, T. Fingscheidt, Unsupervised temporal consistency metric for video segmentation in highly-automated driving, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2020), pp. 1369–1378. Virtual Conference

[VHB+22] S. Varghese, C. Hümmer, A. Bär, F. Hüger, T. Fingscheidt, Joint optimization for DNN model compression and corruption robustness, in *Deep Neural Networks and Data for Automated Driving – Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, Berlin, 2022), pp. 435–458

[VJB+19] T.-H. Vu, H. Jain, M. Bucher, M. Cord, P. Perez, ADVENT: adversarial entropy minimization for domain adaptation in semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 2517–2526

[vLSB20] S. van Steenkiste, F. Locatello, J. Schmidhuber, O. Bachem, Are disentangled representations helpful for abstract visual reasoning? (2020), pp. 1–14. arXiv:1905.12506

[WB98] C.K.I. Williams, D. Barber, Bayesian classification with Gaussian processes. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **20**(12), 1342–1351 (1998)

[Wer78] W. Wertz, *Statistical Density Estimation: A Survey* (Vandenhoeck & Ruprecht, 1978)

[Wes04] T.H.-W. Westerveld, Using generative probabilistic models for multimedia retrieval. Dissertation, University of Twente (2004)

[WFZ19] M. Weber, M. Fürst, J.M. Zöllner, Automated focal loss for image based object detection (2019), pp. 1–9. arXiv:1904.09048

[WGH19] M. Woehrle, C. Gladisch, C. Heinzemann, Open questions in testing of learned computer vision functions for automated driving, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, Toulouse, France (2019), pp. 333–345

[WGSM17] S.C. Wong, A. Gatt, V. Stamatescu, M.D. McDonnell, Understanding data augmentation for classification: when to warp?, in *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Gold Coast, QLD, Australia (2017), pp. 1–6

[WHLX19] H. Wang, Z. He, Z.C. Lipton, E.P. Xing, Learning robust representations by projecting superficial statistics out, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA (2019), pp. 1–16

[WJZ+20] H. Wu, P. Judd, X. Zhang, M. Isaev, P. Micikevicius, Integer quantization for deep learning inference: principles and empirical evaluation (2020), pp. 1–20. arXiv:2004.09602

[WKP13] C. Wang, N. Komodakis, N. Paragios, M.R.F. Modeling, Inference & learning in computer vision & image understanding: a survey. Comput. Vis. Image Underst. **117**(11), 1610–1627 (2013)

[WLDG20] Z. Wu, S.-N. Lim, L. Davis, T. Goldstein, Making an invisibility cloak: real world adversarial attacks on object detectors, in *Proceedings of the European Conference on Computer Vision (ECCV)* (2020), pp. 1–17. Virtual Conference

[WR96] C.K.I. Williams, C.E. Rasmussen, Gaussian processes for regression, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Denver, CO, USA (1996), pp. 514–520

[WSC+20] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, M. Yadong, M. Tan, X. Wang et al., Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **43**(10), 3349–3364 (2020)

[WSRA20] O. Willers, S. Sudholt, S. Raafatnia, S. Abrecht, Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP) Workshops* (2020), pp. 336–350

[WT11] M. Welling, Y.W. Teh, Bayesian learning via stochastic gradient Langevin dynamics, in *Proceedings of the International Conference on Machine Learning (ICML)*, Bellevue, WA, USA (2011), pp. 1–11

[WTPFW19] J. Wu, S. Toscano-Palmerin, P.I. Frazier, A.G. Wilson, Practical multi-fidelity bayesian optimization for hyperparameter tuning, in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, Tel Aviv, Israel (2019), pp. 1–11

[WYKN20] Y. Wang, Q. Yao, J.T. Kwok, M.N. Lionel, Generalizing from a few examples: a survey on few-shot learning. ACM Comput. Surv. (CSUR) **53**(3), 1–34 (2020)

[WZM+16] X.-M. Wang, T.-Y. Zhang, Y.-X. Ma, J. Xia, W. Chen, A survey of visual analytic pipelines. J. Comput. Sci. Technol. **31**(4), 787–804 (2016)

[XCKW19] Y. Xu, P. Cao, Y. Kong, Y. Wang, L_{DMI} : a novel information-theoretic loss function for training deep nets robust to label noise, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2019), pp. 6222–6233

[XCS10] H. Xu, C. Caramanis, S. Sanghavi, Robust PCA via outlier pursuit, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada (2010), pp. 2496–2504

[XGD+17] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA (2017), pp. 1492–1500

[XHM+11] X. Xie, J.W.K. Ho, C. Murphy, G. Kaiser, B. Xu, T.Y. Chen, Testing and validating machine learning classifiers by metamorphic testing. Syst. Softw. **84**(4), 544–558 (2011)

[XLH+20] H. Xu, D. Luo, R. Henao, S. Shah, L. Carin, Learning autoencoders with relational regularization (2020), pp. 1–18. arXiv:2002.02913

[XLZ+18] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, J. Sun, Unified perceptual parsing for scene understanding, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany (2018), pp. 432–448

[XW19] Y. Xiao, W.Y. Wang, Disentangled representation learning with Wasserstein total correlation (2019), pp. 1–10. arXiv:1912.12818

[XWvdM+19] C. Xie, Y. Wu, L. van der Maaten, A.L. Yuille, K. He, Feature denoising for improving adversarial robustness, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 501–507

[XZL+20] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, X. Lin, Adversarial T-shirt! evading person detectors in a physical world, in *Proceedings of the European Conference on Computer Vision (ECCV)* (2020), pp. 665–681. Virtual Conference

[XZZ+19] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, A.L. Yuille, Improving transferability of adversarial examples with input diversity, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 2730–2739

[YDL+17] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, W. Zuo, Mind the class weight bias: weighted maximum mean discrepancy for unsupervised domain adaptation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA (2017), pp. 945–954

[YLLW18] J. Ye, X. Lu, Z. Lin, J. Wang, Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada (2018), pp. 1–11

[YLW+20] H. Yan, Z. Li, Q. Wang, P. Li, X. Yong, W. Zuo, Weighted and class-specific maximum mean discrepancy for unsupervised domain adaptation. IEEE Trans. Multimedia **22**(9), 2420–2433 (2020)

[YXL+19] S. Ye, K. Xu, S. Liu, H. Cheng, J.H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, X. Lin, Adversarial robustness vs. model compression, or both?, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea (2019), pp. 111–120

[YZS+18] G. Yang, H. Zhao, J. Shi, Z. Deng, J. Jia, SegStereo: exploiting semantic information for disparity estimation, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany (2018), pp. 636–651

[ZCAW17] L.M. Zintgraf, T.S. Cohen, T. Adel, M. Welling, Visualizing deep neural network decisions: prediction difference analysis (2017), pp. 1–12. arXiv:1702.04595

[ZCG+19] B. Zoph, E.D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, Q.V. Le, Learning data augmentation strategies for object detection (2013), pp. 1–13. arXiv:1906.11172

[ZCY+19] X. Zhang, X. Chen, L. Yao, C. Ge, M. Dong, Deep neural network hyperparameter optimization with orthogonal array tuning, in *Proceedings of the International Conference on Neural Information Processing (ICONIP)*, Sydney, NSW, Australia (2019), pp. 287–295

[ZE01] B. Zadrozny, C. Elkan, Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers, in *Proceedings of the International Conference on Machine Learning (ICML)*, Williamstown, MA, USA (2001), pp. 609–616

[ZE02] B. Zadrozny, C. Elkan, Transforming classifier scores into accurate multiclass probability estimates, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Edmonton, AB, Canada (2002), pp. 694–699

[ZF14] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Zurich, Switzerland (2014), pp. 818–833

[ZGY+19] F. Zhu, R. Gong, F. Yu, X. Liu, Y. Wang, Z. Li, X. Yang, J. Yan, Towards unified INT8 training for convolutional neural network (2019), pp. 1–14. arXiv:1912.12607

[Zha12] B. Zhang, Reliable classification of vehicle types based on cascade classifier ensembles. IEEE Trans. Intell. Trans. Syst. (TITS) **14**(1), 322–332 (2012)

[ZHD+19] R. Zhao, Y. Hu, J. Dotzel, C. De Sa, Z. Zhang, Improving neural network quantization without retraining using outlier channel splitting, in *Proceedings of the International Conference on Machine Learning (ICML)*, Long Beach, CA, USA (2019), pp. 7543–7552

[ZHML20] J.M. Zhang, M. Harman, L. Ma, Y. Liu, Machine learning testing: survey, landscapes and horizons. IEEE Trans. Softw. Eng. 1–37 (2020). Early access

[ZJRP+15] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, P.H.S. Torr, Conditional random fields as recurrent neural networks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile (2015), pp. 1529–1537

[ZLMJ16] J.R. Zilke, E. Loza Mencía, F. Janssen, DeepRED – rule extraction from deep neural networks, in *Proceedings of the International Conference on Discovery Science (DS)*, Bari, Italy (2016), pp. 457–473

[ZLZ+20] H. Zhou, W. Li, Y. Zhu, Y. Zhang, B. Yu, L. Zhang, C. Liu, Deepbillboard: systematic physical-world testing of autonomous driving systems, in *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)* (2020), pp. 347–358. Virtual Conference

[ZP17] C. Zhou, R.C. Paffenroth, Anomaly detection with robust deep autoencoders, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Halifax, NS, Canada (2017), pp. 665–674

[ZPIE17] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy (2017), pp. 2242–2251

[ZQF+18] H. Zhuo, X. Qian, Y. Fu, H. Yang, X. Xue, SCSP: spectral clustering filter pruning with soft self-adaption manners (2018), pp. 1–14. arXiv:1806.05320

[ZSLG16] S. Zheng, Y. Song, T. Leung, I. Goodfellow, Improving the robustness of deep neural networks via stability training, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA (2016), pp. 4480–4488

[ZSR+19] Y. Zhu, K. Sapra, F.A. Reda, K.J. Shih, S. Newsam, A. Tao, B. Catanzaro, Improving semantic segmentation via video propagation and label relaxation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA (2019), pp. 8856–8865

[ZVSL18] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA (2018), pp. 8697–8710

[ZWN+18] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, Y. Zou, DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients (2018), pp. 1–13. arXiv:1606.06160

[ZZK+20] Z. Zhong, L. Zheng, G. Kang, S. Li, Y. Yang, Random erasing data augmentation, in *Proceedings of the AAAI Conference on Artificial Intelligence*, New York, NY, USA (2020), pp. 13001–13008

[ZZW+18] H. Zhao, S. Zhang, G. Wu, J.M.F. Moura, J.P. Costeira, G.J. Gordon, Adversarial multiple source domain adaptation, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Montréal, QC, Canada (2018), pp. 8568–8579

Recent Advances in Safe AI for Automated Driving

Does Redundancy in AI Perception Systems Help to Test for Super-Human Automated Driving Performance?

Hanno Gottschalk, Matthias Rottmann, and Maida Saltagic

Abstract While automated driving is often advertised with better-than-human driving performance, this chapter reviews that it is nearly impossible to provide direct statistical evidence on the system level that this is actually the case. The amount of labeled data needed would exceed dimensions of present-day technical and economical capabilities. A commonly used strategy therefore is the use of redundancy along with the proof of sufficient subsystems' performances. As it is known, this strategy is efficient especially for the case of subsystems operating independently, i.e., the occurrence of errors is independent in a statistical sense. Here, we give some first considerations and experimental evidence that this strategy is not a free ride as the errors of neural networks fulfilling the same computer vision task, at least in some cases, show correlated occurrences of errors. This remains true, if training data, architecture, and training are kept separate or independence is trained using special loss functions. Using data from different sensors (realized by up to five 2D projections of the 3D MNIST dataset) in our experiments is more efficiently reducing correlations, however not to an extent that is realizing the potential of reduction of testing data that can be obtained for redundant and statistically independent subsystems.

H. Gottschalk and M. Rottmann have equal contribution.

H. Gottschalk (✉) · M. Rottmann · M. Saltagic
School of Mathematics and Science and IZMD, University of Wuppertal, Gaußstr. 20, Wuppertal, Germany
e-mail: hanno.gottschalk@uni-wuppertal.de

M. Rottmann
e-mail: rottmann@uni-wuppertal.de

M. Saltagic
e-mail: maida.saltagic@gmx.de

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_2

1 Introduction

The final report of the ethics committee on automated and connected driving [FBB+17] at the German Federal Ministry of Transportation and Digital Infrastructure starts with the sentences[1] "Partially and fully automated traffic systems serve first and foremost to improve the safety of all road users. [...] Protecting people takes precedence over all other utilitarian considerations. The goal is to reduce harm up to complete prevention. The approval of automated systems is only justifiable if, in comparison with human driving performance, they promise at least a reduction of damage in the sense of a positive risk balance". This pronounced statement sets the highest safety goals. In this chapter, we contemplate the feasibility of a justification based on direct empirical evidence.

If it comes to automated driving, the elephant in the room is the outrageous amount of data that is needed to empirically support the safety requirement set up by the ethics committee with direct measurement. This chapter, however, is not the elephant's first sighting; see, e.g., [KP16], where it is shown that hundreds of millions to billions of test kilometers are required for statistically valid evidence of better-than-human driving performance by automated vehicles. While in this chapter the basic statistical facts on the measurement of the probability of rare events are revisited and adapted to a German context, we slightly extend the findings by estimating the data required for testing AI-based perception functionality using optical sensors along with an estimate of the labeling cost for a sufficient test database.

What is new in this chapter is a statistical discussion and preliminary experimental evidence on redundancy as a potential solution to the aforementioned problem. The decomposition of the system into redundant subsystems, each one capable to trigger the detection of other road users without fusion or filtering, largely reduces the amount of data needed to test each subsystem. However, this is only true if the failure of the subsystems is statistically independent of the other subsystems. This leads to the question of (a) how to measure independence and (b) whether the actual behavior of neural networks supports the independence assumption.

A study on the role of independence in ensembles of deep neural networks was presented in [LWC+19], where the goal was rather (1) to improve performance by selecting ensemble members according to different diversity scores and (2) to obtain robustness against adversarial attacks. In [WLX+20], a number of different consensus algorithms, i.e., ensemble voting algorithms, are compared according to different evaluation metrics. Also in that work, networks are trained independently and selected afterwards.

In our own studies of independence of the occurrence of error events in the prediction of neural networks, we provide experiments for classification with deep neural networks on the academic datasets EMNIST [CATvS17], CIFAR10 [Kri09], and 3D-MNIST.[2] We consider networks with an increasing degree of diversity with respect to training data, architecture, and weight initialization. Even the most diverse networks

[1] Translated from German.

[2] https://www.kaggle.com/daavoo/3d-mnist.

exhibit a Pearson correlation close to 0.60, which clearly contradicts the hypothesis of independence. Also, re-training committees of up to 5 networks with special loss functions to increase independence between committee members by far does not achieve the k-out-of-n performance predicted for independent subsystems [Zac92, ME14]. While it is possible to bring the mean correlation down to zero by special loss functions in the training, this, at least in our preliminary experiments, at the same time deteriorates the performance of the committee members. As the main take-away, redundancy does not necessarily provide a solution to the testing problem.

In this chapter, we do not aim at presenting final results, but only to provide a contribution to an inevitable debate.

Our chapter is organized as follows: In the next section, we evaluate some numbers from the traffic by motor vehicles in the last pre-pandemic year in Germany, 2019. In Sect. 3, we recall some basic facts on the statistics of rare events of an entire system or a system of redundant subsystems. While independent redundant subsystems largely reduce the amount of data required for testing, we also consider the case of correlated subsystems for which the data requirements scale down less neatly. Also, we discuss the amount of data required to actually prove sufficiently low correlation. In Sect. 4, we test neural networks for independence or correlation for simple classification problems. Not surprisingly, we find that such neural networks actually provide correlated error schemes, and the system performance falls far behind the theoretically predicted performance for the error of statistically independent classifiers. This holds true even if we train networks to behave independently or feed the networks with different (toy) sensors. This demonstrates that independence cannot be taken for granted and it might be even hard to achieve through training methods. We give our conclusions and provide a brief outlook on other approaches that have the potential to resolve the issue of outrageous amounts of labeled data for a direct assurance case in the final Sect. 5.

2 How Much Data is Needed for Direct Statistical Evidence of Better-Than-Human Driving?

We focus on the loss of human life as the most important safety requirement. Our frame of reference is set by the traffic in Germany in the year 2019. For this year, the Federal Ministry of Transport and Digital Infrastructure reports 3,046 fatalities which results in 4.0 fatalities per billion kilometers driven on German streets in total and 1.1 fatalities per billion kilometers on motorways; see [Ger20, p. 165] for these and more detailed data.

If we neglect that some accidents do not involve human drivers, that in deadly accidents oftentimes more than one person is killed, and that a large number of those killed did not cause the fatal accident, we obtain a lower bound of at least 250 million kilometers driven per fatality caused by the average human driver. Research on how much this is underestimating the actual distance is recommended but beyond the

scope of this chapter. For an upper bound, we multiply this number by an ad hoc safety factor of 10.

Assuming an average velocity in the range of 50–100 km/h, this amounts to an average time of about 2.5–50 million hours or 285–5,700 years of permanent driving until the occurrence of a single fatal accident. If a camera sensor works at a frame rate of 20–30 fps, $(1.8–54)\times 10^{11}$ frames are processed by the AI-system in this time, corresponding to 0.18–5.4 exabyte (1 exabyte $= 1 \times 10^{18}$ bytes) of data, assuming 1 megabyte per high-resolution image.

Several factors can be identified that would leverage or discount the amount of data required for a direct test of better-than-human safety. We do not claim that our selection of factors is exhaustive and new ideas might change the figures in the future. Nevertheless, here we present some factors that certainly are of importance.

First, due to the strong correlation of consecutive frames, the frame rate of 20–30 fps presumably can be reduced for labeled data. Here, we take the approach that correlation probably is high if the automated car has driven less than one meter, but after 10 m driven there is probably not much correlation is left that one could infer the safety of the automated car and its environment from the fact that it was safe 10 m back. At the same time, this approach eliminates the effect of the average traveling speed.

One could argue further that on many frames, no safety-relevant instance of other road users is given and one could potentially avoid labeling such frames. However, from the history of fatal accidents with the involvement of autopilots we learn that such accidents could even be triggered in situations considered to be non-safety-critical from a human perspective; see, e.g., [Nat20]. As direct measurement of safety should not be based on assumptions, unless they can be supported by evidence, we do not suggest introducing a discounting factor as we would have to assume without proof that we could separate hazardous from non-hazardous situations. This of course does not exclude that a refined methodology is developed in the future that is capable to provide this separation and we refer to the extensive literature on corner cases; see, e.g., [BBLFs19, BTLFs20, BTLFs21, HBR+21].

On the other hand, as we will present in Sect. 3, a statistically valid estimate of the frequency of rare events requires a leverage factor of at least 3–10 applied on the average number of frames per incident; see Sect. 3.1 for the details and precise values.

Further, in the presence of a clear trend of the reduction of fatalities in street traffic [Ger20] (potentially, partially due to AI-based assistance systems already), a mere reproduction of the present-day level of safety in driving does not seem to be sufficient. Without deeper ethical or scientific justification, we assume that humans expect at least 10–100 times lower failure rate of robots than they would concede themselves, while at the same time, we recommend further ethical research and political debate on this critical number. Even with a reduction number of 100, the approximately 30 fatalities due to autonomous vehicles would well exceed the risk

of being struck by lightning causing ∼4 fatalities per year in Germany,[3] which often is considered as a generally acceptable risk.

In addition, several failure modes exist aside from AI-based perception that cause fatalities in transportation. We therefore must not reserve the entire cake of acceptable risk to perception-related errors, only. Instead, here we suggest a fraction of $\frac{1}{10}$ to $\frac{1}{2}$ of the entire acceptable risk for perception-related root causes of fatalities. Also at this place, we recommend more serious research, ethical consideration, and public debate.

Drawing all this together, we obtain a total number of frames that ranges from 1.50 trillion frames in the best-case scenario to 23,000 trillion frames, or 1.5–23,000 exabyte (in the year of reference 2019, the entire Internet contained 33,000 exabyte[4] of data). This computation is summarized from Table 1.

Note that replacing fatalities with injuries reduces the amount of data by roughly a factor of one hundred (exact number for 2019 4/509) [Ger20].

Direct measurement of reliability of an AI-perception system requires labeled data, and the time to annotate a single frame by a human ranges from a couple of minutes for bounding box labeling up to 90 min for a fully segmented image [Sch19]. Working with the span of 5–90 min per annotated frame and a span of wages from a minimum wage of 9.19 Euro for Germany in our year of reference 2019 as lower bound to 15 EUR as upper bound, the cost of labeling a single image ranges from 0.775 to 22.5 EUR.

The total cost for labeling of test data to produce direct statistical evidence therefore ranges from 1.16 trillion in the best case to 51,800 trillion Euro in the worst case. This compares to 3.5 trillion Euro of Germany's gross domestic product in 2019.

We conclude in agreement with [KP16] that direct and assumption-free statistical evidence of safety of the AI-based perception function of an automated vehicle that complies with the safety requirements derived from the ethics committee's final report is largely infeasible with the present-day technology.

Certainly, this does not say anything about whether an AI-based system for automated driving actually *would be* driving better than human. Many experts, including the authors, believe it *could be*, at least in the long run. But the subjective belief of technology-loving experts—in the absence of direct evidence—is certainly insufficient to give credibility to the *promise* of enhanced safety due to automated driving in the sense of the ethics committee's introductory statement.

This of course does not exclude that safety arguments which are based on indirect forms of evidence are reasonable and possible, if they are based on assumptions that can be and are carefully checked. In fact, in the following, we discuss one such potential strategy based on redundancy and report some problems and some progress with this approach applied to small, academic examples.

[3] https://www.vde.com/de/blitzschutz/infos/bitzunfaelle-blitzschaeden#statistik.

[4] Here, for clarity, we use powers of 10, e.g., 1000, instead of powers of 2, e.g., 1024.

Table 1 Factors and numbers that influence the data requirement for a statistically sound assurance case by direct testing. Safety factors (↑) multiply and reduction factors (↓) divide the number of frames/cost

Description	↑↓	Quantity Lower Bound	Quantity Upper Bound	Unit	Source	Frames Lower Bound	Frames Upper Bound
Meters to fatal accident (2019)		2.50×10^{11}	2.50×10^{12}	m	[Ger20] & ad hoc		
Meters driven per frame	↓	1	10	m/f	ad hoc assumption	2.50×10^{10}	2.50×10^{12}
Factor for stat. evidence	↑	2.99 $\alpha = 5\%$	9.21 $\alpha = 0.01\%$	factor	Sect. 3.1	7.49×10^{10}	2.30×10^{13}
Add. safety by autom. driving	↑	10	100	factor	ad hoc assumption	7.49×10^{11}	2.30×10^{15}
Fraction of perception risk from total risk	↓	$\frac{1}{10}$	$\frac{1}{2}$	factor	ad hoc assumption	1.50×10^{12}	2.30×10^{16}
						Cost (EUR) lower bound	Cost (EUR) upper bound
Labeling time per frame	↑	5	90	min	[Sch19] & ad hoc		
Hourly wages	↑	9.19	15	EUR/h	minimum wage GER 2019 & ad hoc		
Cost per frame	↑	0.775	22.5	EUR/f	2 rows above		
Total cost				EUR	frames× cost per frame	1.16×10^{12}	5.18×10^{17}

3 Measurement of Failure Probabilities

3.1 Statistical Evidence for Low Failure Probability

In this subsection, we provide the mathematical reasoning for the leverage factor of 2.99–9.21 that accounts for statistical evidence. Let us denote by p the actual probability of a fatal accident for one single kilometer of automated driving. We are looking for statistical evidence that $p \leq p_{tol} = f_{tol} \cdot p_{human}$, where f_{tol} is a debit factor for

enhanced safety of robots and multiple technical risks. From Table 1, we infer that $f_{\text{tol}} \in [\frac{1}{1000}, \frac{1}{20}]$ taking into account the fraction of perception risk from total risk and the additional safety due to automated driving, cf. Sect. 2. Here, $\text{p}_{\text{human}} \approx \frac{1}{250{,}000{,}000}$ is the (estimated, upper bound) probability of a fatal accident caused by a human driver per driven kilometer. With $\hat{p} = \frac{N_{\text{obs}}}{N_{\text{test}}}$, we denote the estimated probability of a fatal accident per kilometer driven for the autonomous vehicle based on the observed number of fatal accidents N_{obs} on N_{test} kilometers of test driving. We want to test for the alternative hypothesis H_1 that p is below p_{tol} at a level of confidence $1 - \alpha$ with $\alpha \in (0, 1)$ a small number, e.g., $\alpha = 5\%, 1\%, 0.1\%, 0.01\%$ or even smaller. We thus assume the null hypothesis H_0 that $\text{p} \geq \text{p}_{\text{tol}}$ using that under the null hypothesis $N_{\text{obs}} \sim B(N_{\text{test}}, \text{p}_{\text{tol}})$ is Bernoulli-distributed with probability p_{tol} and N_{test} repetitions. The exact one-sided Bernoulli test rejects the null hypothesis and accepts H_1 provided that

$$\begin{aligned} 1 - \alpha \leq \quad & \text{P}_{N \sim B(N_{\text{test}}, \text{p}_{\text{tol}})}(N > N_{\text{obs}}) = 1 - \text{P}_{N \sim B(N_{\text{test}}, \text{p}_{\text{tol}})}(N \leq N_{\text{obs}}) \\ & = 1 - \sum_{j=0}^{N_{\text{obs}}} \binom{N_{\text{test}}}{j} (\text{p}_{\text{tol}})^j \left(1 - \text{p}_{\text{tol}}\right)^{N_{\text{test}} - j}. \end{aligned} \tag{1}$$

The reasoning behind (1) is the following: Assume H_0, i.e., the true probability of a fatal accident due to the autonomous vehicle would be higher than p_{tol}. Then, with a high probability of at least $1 - \alpha$ we would have seen more fatal accidents than just N_{obs}, which we actually observed. This puts us in front of the alternative to either believe that in our test campaign we just observed an extremely rare event of probability α, or to discard the hypothesis H_0 that the safety requirements are *not* fulfilled.

Let us suppose for the moment that the outcome of the test campaign is ideal, i.e., no fatal accidents are observed at all, i.e., $N_{\text{obs}} = 0$. In this ideal case, (1) is equivalent to

$$\alpha \geq \left(1 - \text{p}_{\text{tol}}\right)^{N_{\text{test}}} \quad \Leftrightarrow \quad -\frac{\ln(\alpha)}{N_{\text{test}}} \leq -\ln\left(1 - \text{p}_{\text{tol}}\right) \approx \text{p}_{\text{tol}}, \tag{2}$$

where we used the first-order Taylor series expansion of the natural logarithm at 1, which is highly precise as p_{tol} is small. Thus, even in the ideal case of zero fatalities observed, $N_{\text{test}} \geq -\frac{\ln(\alpha)}{\text{p}_{\text{tol}}}$ is required. For α ranging from 5% to 0.01%, $-\ln(\alpha)$ roughly ranges from 3 (numerical value 2.9976) to 10 (numerical value 9.2103). This explains the back of the envelope estimates in Sect. 2.

Note that the approach of [KP16] differs as it is based on a rate estimate for the Poisson distribution. Nevertheless, as binominal and Poisson distribution for low probabilities approximate each other very well, this difference is negligible, as the difference is essentially proportional to the event of two or more fatal incidents in one kilometer driven.

3.2 Test Data for Redundant Systems

Assuming independence of subsystems: Let $(\mathbf{x}, y)$ be a pair of random variables, where $\mathbf{x} \in \mathcal{X}$ represents the input data presented to two neural networks h_1 and h_2, and $y \in \mathcal{Y}$ denotes the corresponding ground truth label. We assume that $(\mathbf{x}, y)$ follows a joint distribution P possessing a corresponding density p. For each neural network, the event of failure is described by

$$\mathcal{F}_i := \{h_i(\mathbf{x}) \neq y\}\,, \quad i = 1, 2,$$

with $1_{\mathcal{F}_i}$ being their corresponding indicator variables that are equal to one for an event in $\mathcal{F}_i$ and zero else. If and only if we assume independence of the events $\mathcal{F}_i$, we obtain

$$\mathbb{E}[1_{\mathcal{F}_1} \cdot 1_{\mathcal{F}_2}] = \mathrm{P}(\mathcal{F}_1 \cap \mathcal{F}_2) = \mathrm{P}(\mathcal{F}_1) \cdot \mathrm{P}(\mathcal{F}_2) = \mathbb{E}[1_{\mathcal{F}_1}] \cdot \mathbb{E}[1_{\mathcal{F}_2}]\,,$$

which implies that the covariance fulfills

$$\mathrm{COV}(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2}) = \mathbb{E}[1_{\mathcal{F}_1} \cdot 1_{\mathcal{F}_2}] - \mathbb{E}[1_{\mathcal{F}_1}] \cdot \mathbb{E}[1_{\mathcal{F}_2}] = 0\,.$$

This is easily extended to n neural networks $h_i(\mathbf{x})$, $i \in \mathcal{I} = \{1, \ldots, n\}$ and their corresponding failure sets $\mathcal{F}_i$. Under the hypothesis of independence of the family of events $\mathcal{F}_i$, we obtain

$$\mathrm{p}_{\mathrm{system}} = \mathbb{E}\left[\prod_{i \in \mathcal{I}} 1_{\mathcal{F}_i}\right] = \prod_{i \in \mathcal{I}} \mathrm{P}(\mathcal{F}_i) = \prod_{i \in \mathcal{I}} \mathrm{p}_{\mathrm{sub},i}\,,$$

where $\mathrm{p}_{\mathrm{system}}$ is the probability of failure of a system of $\#\mathcal{I} = n$ redundant neural networks working in parallel, where failure is defined as all networks being wrong at the same time [ME14] and $\mathrm{p}_{\mathrm{sub},i} = \mathrm{P}(\mathcal{F}_i)$ is the probability of failure for the ith subsystem $h_i(\mathbf{x})$.

Let us suppose for convenience that the probability for the subsystems $h_i(\mathbf{x})$ are all equal, $\mathrm{p}_{\mathrm{sub},i} = \mathrm{p}_{\mathrm{sub}}$. Then $\mathrm{p}_{\mathrm{system}} = \mathrm{p}_{\mathrm{sub}}^n$. In order to give evidence that $\mathrm{p}_{\mathrm{system}} < \mathrm{p}_{\mathrm{tol}}$, it is thus enough to provide evidence for $\mathrm{p}_{\mathrm{sub}} = \mathrm{p}_{\mathrm{sub},i} < \mathrm{p}_{\mathrm{tol}}^{\frac{1}{n}}$ for $i \in \mathcal{I}$. Subsystem testing to a confidence of $(1 - \alpha)$ on the system level requires a higher confidence at the subsystem level, which, by a simple Bonferroni correction [HS16], can be conservatively estimated as $(1 - \frac{\alpha}{n})$. Consequently, by (2) the amount of data for testing the subsystem $h_i(\mathbf{x})$ is given by

$$-\frac{\ln\left(\frac{\alpha}{n}\right)}{N_{\mathrm{test},i}} > \mathrm{p}_{\mathrm{tol}}^{\frac{1}{n}} \quad \Leftrightarrow \quad N_{\mathrm{test},i} > -\frac{\ln\left(\frac{\alpha}{n}\right)}{\mathrm{p}_{\mathrm{tol}}^{\frac{1}{n}}}. \tag{3}$$

As $\mathrm{p}_{\mathrm{tol}}^{\frac{1}{n}}$ is much larger than $\mathrm{p}_{\mathrm{tol}}$, the amount of testing data required is radically reduced, even if one employs n separate test sets for all n subsystems. By comparison of (2) and (3), the factor of reduction is roughly

$$\gamma_n = \frac{N_{\mathrm{test}}}{n N_{\mathrm{test},i}} = \frac{1}{n\, p_{\mathrm{tol}}^{1-\frac{1}{n}} \left(1 - \frac{\ln(n)}{\ln(\alpha)}\right)}.$$

For example, for $n = 2$ in the best-case scenario, $\alpha = 5\%$, and $f_{\mathrm{tol}} = \frac{1}{20}$ the reduction factor is $\gamma_2 = 28{,}712$, which reduces the corresponding 1.5×10^{12} frames to 52.2 million frames. This already is no longer an absolutely infeasible number. For the worst-case scenario, $\alpha = 0.01\%$ and $f_{\mathrm{tol}} = \frac{1}{1000}$, the reduction factor is $\gamma_2 = 232{,}502$, resulting in 98.9 billion frames, which seems out of reach, but not to the extent of 2.3×10^{16} frames.

Keeping the other values fixed, $n = 3$ even yields a reduction factor of $\gamma_3 = 974{,}672$ in the best-case scenario and $\gamma_3 = 11{,}818{,}614$ in the worst-case scenario, resulting in 1,54 million frames in the best and 1,95 billion frames in the worst-case scenario. These numbers look almost realizable, given the economic interests at stake.

However, the strategy based on redundant subsystems comes with a catch. It is only applicable, *if* the subsystems are independent. But this is an assumption that is not necessarily true. We therefore investigate what happens, if the errors of subsystems are not independent.

Assuming no independence of subsystems: In this case, the covariance of the error indicator variables $1_{\mathcal{F}_i}$ is not equal to zero and can be regarded as a measure of the joint variability for the random variables $1_{\mathcal{F}_i}$. The normalized version of the covariance is the Pearson correlation

$$\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2}) = \frac{\mathrm{COV}(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})}{\sigma(1_{\mathcal{F}_1}) \cdot \sigma(1_{\mathcal{F}_2})} \in [-1, 1],$$

where $\sigma(1_{\mathcal{F}_i}) = \sqrt{\mathrm{p}_{\mathrm{sub},i}(1 - \mathrm{p}_{\mathrm{sub},i})}$ denotes the standard deviation of $1_{\mathcal{F}_i}$, which is supposed to be greater than zero for $i = 1, 2$. The correlation measures the linear relationship between the random variables $1_{\mathcal{F}_i}$ and takes values ± 1 if the relationship between the random variables is deterministic.

Let us first consider a system with two redundant subsystems in parallel, $h_i(\mathbf{x})$, $i = 1, 2$, where we however drop the assumption of independence. Then we obtain

$$\begin{aligned} \mathrm{p}_{\mathrm{system}} &= \mathbb{E}[1_{\mathcal{F}_1} \cdot 1_{\mathcal{F}_2}] \\ &= \mathrm{COV}(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2}) + \mathbb{E}[1_{\mathcal{F}_1}] \cdot \mathbb{E}[1_{\mathcal{F}_2}] \\ &= \rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2}) \sqrt{\mathrm{p}_{\mathrm{sub},1}(1 - \mathrm{p}_{\mathrm{sub},1})} \sqrt{\mathrm{p}_{\mathrm{sub},2}(1 - \mathrm{p}_{\mathrm{sub},2})} + \mathrm{p}_{\mathrm{sub},1}\mathrm{p}_{\mathrm{sub},2}. \end{aligned} \tag{4}$$

Assuming again equal failure probabilities for the subsystems $\mathrm{p}_{\mathrm{sub}} = \mathrm{p}_{\mathrm{sub},1} = \mathrm{p}_{\mathrm{sub},2}$ and using $1 - \mathrm{p}_{\mathrm{sub}} \approx 1$ as a good approximation as $\mathrm{p}_{\mathrm{sub}}$ for a safe system is very small, we obtain from (4)

$$\mathrm{p}_{\mathrm{system}} \approx \rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})\mathrm{p}_{\mathrm{sub}} + \mathrm{p}_{\mathrm{sub}}^2, \tag{5}$$

i.e., we can only expect a reduction of the frames needed for testing which is comparable to the case, where statistical independence holds, if $\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})$ is of the same small order of magnitude as $\mathrm{p}_{\mathrm{sub}}$. If, e.g., we assume an extremely weak correlation of $\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2}) = 0.01$, we can essentially neglect the $\mathrm{p}_{\mathrm{sub}}^2$-term as $\mathrm{p}_{\mathrm{sub}} \ll 0.01$ and realize that the reduction factor essentially is $\frac{1}{\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})} = 100$, only. Thus, even for such a pretty uncorrelated error scheme, the number of frames required for testing would be lower bounded by 1.5×10^{10} to 2.3×10^{14} frames, even neglecting Bonferroni correction and independent test sets which make up a multiplication factor $B_2 = n(1 - \frac{\log(n)}{\log(\alpha)})$ yielding $B_2 = 2.46$ and $B_2 = 2.15$, respectively. With these effects taken into account, we arrive at 36.9 billion frames in the best-case scenario and 4.94×10^{14} frames in the worst case, where even the lower number of frames seems hardly feasible.

A related computation for $n = 3$ yields to leading order, using (5) and approximating the complement of small probabilities with one and neglecting terms of order $\mathrm{p}_{\mathrm{sub}}^2$,

$$\begin{aligned}
\mathrm{p}_{\mathrm{system}} = & \quad \mathbb{E}\left[1_{\mathcal{F}_1} \cdot 1_{\mathcal{F}_2} \cdot 1_{\mathcal{F}_3}\right] \\
\approx & \quad \rho(1_{\mathcal{F}_1 \cap 1_{\mathcal{F}_2}}, 1_{\mathcal{F}_3})\sqrt{\mathbb{E}\left[1_{\mathcal{F}_1} \cdot 1_{\mathcal{F}_2}\right]\mathrm{p}_{\mathrm{sub}}} + \mathbb{E}\left[1_{\mathcal{F}_1} \cdot 1_{\mathcal{F}_2}\right]\mathrm{p}_{\mathrm{sub}} \\
\approx & \quad \rho(1_{\mathcal{F}_1 \cap 1_{\mathcal{F}_2}}, 1_{\mathcal{F}_3})\sqrt{\left(\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})\mathrm{p}_{\mathrm{sub}} + \mathrm{p}_{\mathrm{sub}}^2\right)\mathrm{p}_{\mathrm{sub}}} \\
& \quad + \left(\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})\mathrm{p}_{\mathrm{sub}} + \mathrm{p}_{\mathrm{sub}}^2\right)\mathrm{p}_{\mathrm{sub}} \\
\approx & \quad \rho(1_{\mathcal{F}_1 \cap 1_{\mathcal{F}_2}}, 1_{\mathcal{F}_3})\sqrt{\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})}\mathrm{p}_{\mathrm{sub}}\,.
\end{aligned} \tag{6}$$

If we thus assume that both correlations $\rho(1_{\mathcal{F}_1 \cap 1_{\mathcal{F}_2}}, 1_{\mathcal{F}_3})$ between the failure of subsystem $h_3(\mathbf{x})$ and the composite redundant subsystem from $h_3(\mathbf{x})$ and $h_2(\mathbf{x})$ are both equal to 0.01, we obtain a total reduction factor of roughly $\frac{1}{\rho(1_{\mathcal{F}_1 \cap 1_{\mathcal{F}_2}}, 1_{\mathcal{F}_3})\sqrt{\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})}} =$ 1,000, which still leads to roughly $1.50 \times 10^9 - 2.30 \times 10^{13}$ frames, even without Bonferroni correction and independent test sets for subsystems. With both taken into account, the amount of data ranges from 5,04 billion frames in the best-case scenario to 9.43×10^{13} frames in the worst case, where only the figure obtained in the best case, based on problematic choices, seems remotely feasible. However, in the presence of domain shifts in time and location, it seems questionable if the road of testing weakly correlated subsystems is viable (supposed they *are* weekly correlated).

We also note that correlation coefficients as low as $\rho = 0.01$ are rarely found in nature and in addition, it requires empirical testing to provide evidence for a low correlation. The correlations we measure in Sect. 4 for the case of simple classification

problems miss this low level by at least an order of magnitude, leading to an extra factor of at least 10 in the above considerations.

3.3 Data Requirements for Statistical Evidence of Low Correlation

In the preceding section, we have analyzed that low correlation between subsystems efficiently reduces the data required for testing better-than-human safety of autonomous vehicles. However, to achieve this, e.g., in the case of two redundant subsystems, the correlation has to be in the order of magnitude of $\mathrm{p}_{\mathrm{sub}} = \mathrm{p}_{\mathrm{tol}}^{\frac{1}{2}}$. Statistical evidence for such a low level of correlation requires data itself. Let us suppose the ideal situation once more that a correlation coefficient is strictly zero $\rho = 0$ and we would like to compute the number of pairs of observations of the random variables $(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})$ that is needed to prove that ρ is in the order of magnitude $\mathrm{p}_{\mathrm{sub}}$, as required for a decent downscaling of the number of test data frames. In other words, we have to estimate a number of samples needed to provide statistical evidence at a given significance level α that $\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})$ is less than $\mathrm{p}_{\mathrm{sub}} = \mathrm{p}_{\mathrm{tol}}^{\frac{1}{2}}$.

As shown by Raymond Fisher and others, see, e.g., [LL88], the quantity $\hat{Z} = \frac{1}{2}\log\left(\frac{1+\hat{\rho}}{1-\hat{\rho}}\right)$ is asymptotically normally distributed with expected value $\frac{1}{2}\log(\frac{1+\rho}{1-\rho}) = 0$ in our case, where we assumed $\rho = 0$. The standard deviation is given by $\sqrt{\frac{1}{N_{\mathrm{test}}-3}}$. Here, $\hat{\rho}$ stands for the empirical correlation coefficient of the pair of observations [HS16].

A two-sided confidence interval for a given level of confidence $1-\alpha$ for the observed value $\hat{z}$ of $\hat{Z}$ thus is given by

$$\hat{z}_{\pm} = \hat{z} \pm z_{1-\frac{\alpha}{2}} \sqrt{\frac{1}{N_{\mathrm{test}} - 3}}, \tag{7}$$

where $z_{1-\frac{\alpha}{2}}$ is the $1-\frac{\alpha}{2}$ quantile of the standard normal distribution. Transforming back (7), we obtain lower and upper bounds

$$\hat{\rho}_{\pm} = \frac{\exp(2\hat{z}_{\pm}) - 1}{\exp(2\hat{z}_{\pm}) + 1}. \tag{8}$$

Under our best-case hypothesis $\rho = 0$, the boundaries $\hat{z}_{\pm}$ of the confidence interval converge to zero; we may apply the δ-rule with the derivative $\frac{d}{dz}\frac{\exp(2z)-1}{\exp(2z)+1}\Big|_{z=0} = \frac{4\exp(2z)}{(\exp(2z)+1)^2}\Big|_{z=0} = 1$. Let us consider the width W of the confidence interval for ρ for the best possible outcome obtained for $\hat{z} = 0$. By (8) and the δ-rule, asymptotically for large N_{test} it is given by $W = 2z_{1-\frac{\alpha}{2}}\sqrt{\frac{1}{N_{\mathrm{test}}-3}}$. Even if this is the case, to infer

that $|\rho| \leq \mathrm{p}_{\mathrm{sub}}$ with confidence $1-\alpha$, one requires, for the case of two subsystems, $z_{1-\frac{\alpha}{2}}\sqrt{\frac{1}{N_{\mathrm{test}}-3}} \leq \mathrm{p}_{\mathrm{sub}} = \mathrm{p}_{\mathrm{tol}}^{\frac{1}{2}}$. If N_{test} is large, we can neglect the -3 term and obtain for the best case

$$N_{\mathrm{test}} \approx \frac{z_{1-\frac{\alpha}{2}}^2}{p_{\mathrm{tol}}}. \tag{9}$$

Not unexpectedly, this brings back the bad scaling behavior observed in (2) and the related problematic data requirements, which are essentially the same as for the non-redundant, direct approach. The numbers for $z_{1-\frac{\alpha}{2}}^2$ for $\alpha = 5\% \ldots \alpha = 0.01\%$ range from 2.706 to 13.831 which essentially confirms the range of roughly 3 …10 for the statistical safety factor obtained from (1) and (2).

4 Correlation Between Errors of Neural Networks in Computer Vision

As of now, deep neural networks (DNNs) for perception tasks are far away from being perfect. Motivated by common practices in reliability engineering, redundancy, i.e., the deployment of multiple system components pursuing the same task in parallel, might be one possible approach toward improving the reliability of a perception system.

Redundancy can enter into perception systems in many ways. Assume a system setup with multiple sensors, e.g., camera, LiDAR, and RaDAR. There are multiple options to design a deep-learning-driven perception system processing the different sensors' data. A non-exhaustive list of designs may look as follows:

1. Only a single sensor is processed; this is done by a single DNN;
2. Only a single sensor is processed; this is done by a committee of DNNs;
3. All sensors are processed by a single-sensor-fusing DNN;
4. All sensors are processed by a committee of sensor-fusing DNNs;
5. Each sensor is processed by a separate DNN; the results are fused afterwards;
6. Each sensor is processed by a committee of DNNs; the results are fused afterwards.

Except for the first design, all other designs incorporate redundancy. Herein, there are two types of redundancy, redundancy via multiple sensors (all pursuing the same task of perceiving the environment) and redundancy via multiple DNNs.

Certainly, approach one is only eligible for direct testing, see Sect. 3.1, and the same is true for the "early fusion" approach 3. All the other approaches could potentially benefit from redundancy, if independence or low correlation of the errors can be assumed. Therefore, the degree of independence, which can be understood and quantified as the degree of uncorrelatedness, is a quantity of interest for safety and also for testing; see Sect. 3.2. However, as explained in Sect. 3.2, in order to use redundancy as a part of the solution to the testing problem outlined in Sect. 2, correlation has to be extremely low.

For the case of simple DNNs processing the same sensor's data, we give evidence that such low correlation in general does not hold. The evidence we find rather points in the opposite direction that it is hard to obtain a correlation that is below 0.5, even if the training datasets and network architecture are kept well separated. On the other hand, one could try to train DNNs such that their failure events are uncorrelated.

In this section, we show preliminary numerical results on MNIST (handwritten digits), EMNIST (also containing handwritten letters), and 3D-MNIST for

- training DNNs for independence/less correlated failure events;
- the role of different sensors (by viewing 3D-MNIST examples from different directions).

Although our findings do not directly apply to physically more diverse sensors like Camera, LiDAR, and RaDAR, these preliminary results indicate that the independence of DNN-based perception systems cannot be taken simply for granted, even if different sources of data are employed.

4.1 Estimation of Reliability for Dependent Subsystems

Most commonly, the so-called active parallel connection of n subsystems is used, wherein the entire system (meaning the ensemble of DNNs) is assumed to be functional iff at least one subsystem (which corresponds to one committee member h_i) is functional. However, the active parallel connection is not the only decision rule of interest which can be applied to the committee h_i, $i = 1, \ldots, n$. For instance, considering a pedestrian detection performed by a committee h_i that detects a pedestrian if at least one of the DNNs does so. For increasing n, we would expect an increase in false positive detections, therefore facing the typical trade-off of false positives and false negatives. In order to steer this trade-off, we use k-out-of-n systems that are functional iff at least k out of n components h_i are functional, i.e., at most $n - k$ components fail. Hence, we are interested in the event

$$\left\{ \sum_{i=1}^{n} 1_{\mathcal{F}_i} < n - k \right\}$$

and its probability which is the probability of the k-out-of-n system being functional. The reliability of k-out-of-n system can be expressed analytically in terms of the reliability of its components; see also [Zac92]. For $k = 1$, this boils down to the active parallel connection.

If we assume independence of the failure events $\mathcal{F}_i$ and that all networks fail with equal probability $\mathrm{P}(\mathcal{F}_i) = \mathrm{p}_{\mathrm{sub},i} = \mathrm{p}_{\mathrm{sub}}$, then the probability that at least k-out-of-n networks are functional can be calculated via

$$P\left(\sum_{i=1}^{n} 1_{\mathcal{F}_i} < n-k\right) = \sum_{j=k}^{n} \binom{n}{j} (1-\mathrm{p}_{\mathrm{sub}})^j \cdot \mathrm{p}_{\mathrm{sub}}^{n-j} = 1 - F_B(k-1; n, 1-\mathrm{p}_{\mathrm{sub}}), \tag{10}$$

where F_B denotes the distribution function of the binomial distribution. This quantity serves as a reference in our experiments.

4.2 Numerical Experiments

In this section, we conduct the first experiments using the datasets EMNIST, CIFAR10, and 3D-MNIST. The original dataset MNIST [LCB98] contains 60,000 grayscale images of size 28×28 displaying handwritten digits (10 classes). EMNIST is an extension of MNIST that contains handwritten digits and characters of the same 28×28 resolution. Of that dataset, we only considered the characters (26 classes) of which there are 145,600 available. We used 60,000 images for training, 20,000 for validation, and the rest for testing. CIFAR10 contains 60,000 RGB images of size 32×32 categorized into 10 classes (from the categories animals and machines). We used the default split of 50,000 training and 10,000 test examples. We reserved a quarter of the training set for validation. 3D-MNIST contains point clouds living on a 16^3-lattice. The dataset contains 10,000 training and 2,000 test examples.

We used convolutional DNNs implemented in `Keras` [C+15] with simple architectures; if not stated otherwise they contain 2 convolutional layers with 32 and 64 3×3-filters, respectively, each of them followed by a leakyReLU activation and 2×2 max pooling, and finally a single dense layer. For training, we used a batch size of 256, weight decay of 10^{-4}, and the Adam optimizer [KB15] with default parameters, except for the learning rate. We started with a learning rate of 10^{-2}, trained until stagnation, and repeated that with a learning rate of 10^{-3}.

Reducing correlations with varying training data, architecture, and weight initializers: First, we study to what extent independence can be promoted by varying the training data, architecture, and weight initializers in an active parallel system with $n = 2$ DNNs. To this end, we split the training data and validation data into two disjoint chunks; consider another network, where we add a third convolutional layer with 128 filters, again followed by leakyReLU and max pooling, and use the Glorot uniform and Glorot normal initializers with default parameters. For the sake of brevity, we introduce a short three-bit notation indicating the boolean truth values corresponding to the questions

$$\text{(same data? same architecture? same initializer?).} \tag{11}$$

For instance, 101 stands for two DNNs being trained with the same data, having different architectures, and using the same initializer. We report results in terms of *average accuracy* estimating $\frac{1}{n}\sum_{i=1}^{n}(1-\mathrm{P}(\mathcal{F}_i))$ and *joint accuracy* estimating $1-\mathrm{P}(\bigcap_{i=1}^{n}\mathcal{F}_i)$.

Table 2 Correlation coefficients and average accuracies for EMNIST and CIFAR10. The configurations read as defined in (11). All runs have been performed 10 times, all numbers are averaged over these 10 runs, and the standard deviation over these 10 runs are given

Configurations	EMNIST		CIFAR10	
	$\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})$	avg. acc. (%)	$\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})$	avg. acc (%)
111	0.71±0.01	91.17±0.05	0.73±0.01	72.27±0.45
110	0.71±0.00	91.17±0.06	0.74±0.01	72.19±0.22
101	0.71±0.01	91.13±0.04	0.74±0.01	72.08±0.43
100	0.71±0.01	91.13±0.05	0.74±0.01	72.14±0.36
011	0.58±0.01	89.65±0.14	0.66±0.02	66.10±0.74
010	0.58±0.01	89.76±0.08	0.65±0.01	66.08±0.35
001	0.57±0.01	89.74±0.07	0.66±0.01	66.29±0.65
000	0.58±0.01	89.62±0.13	0.65±0.01	66.30±0.38

Table 2 shows in all cases correlation coefficients much greater than zero. Corresponding χ^2 tests with significance level $\alpha = 0.05$ in all cases rejected the hypothesis that the DNNs are independent. Noteworthily, varying the initializer or the network architecture barely changes the results while changing the training data seems to have the biggest impact, clearly reducing the correlation coefficient. A reduction in correlation also reduces the average performance of the networks. For the sake of comparability, all networks were only trained with half of the training data, since otherwise the configurations with equal data would have the advantage of working with twice the amount of training data compared to the configuration with different training data.

Next, we study in this setting whether we can achieve at least conditional independence. To this end, we aim at conditioning the difficulty of the task by conditioning to softmax entropy quantiles. More precisely, we compute the entropy of the softmax distribution of all data points of both networks. We then sum the entropy values for each data point over the two networks and group all examples into 8 equally sized bins according to ascending summed entropy.

Table 3 shows that the higher the softmax entropy gets, the less the DNNs failures are correlated. This goes down to correlation coefficients of 0.25 for EMNIST and 0.23 for CIFAR10, when considering the softmax entropy bin no. 8 with the highest entropy values. Still, χ^2 tests reveal that the correlations are too strong to assume independent failures.

Training two networks for enhanced independence: Since the measures considered so far do not lead to success, we explicitly try to decorrelate the failures of the DNNs. To this end, we incorporate an additional loss function into training, added to the typically used empirical cross-entropy loss. Let $\mathrm{p}(y|\mathbf{x}, h_i)$ denote the probability estimated by the DNN h_i that the class y is the correct one given the input $\mathbf{x}$. One possible approach is to explicitly enforce a prediction of h_1 different from that of h_2

Table 3 Correlation coefficients $\rho(1_{\mathcal{F}_1}, 1_{\mathcal{F}_2})$ for different quantiles of softmax entropy computed on EMNIST and CIFAR10. The configurations read as defined in (11). The experiments have been repeated 10 times; the corresponding standard errors are of the order of 0.01

	EMNIST								CIFAR10							
Entropy bin	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Config.	Correlation coefficients															
111	1.0	1.0	1.0	1.0	0.99	0.94	0.71	0.45	1.0	0.99	0.94	0.79	0.64	0.57	0.45	0.26
110	1.0	1.0	1.0	1.0	0.99	0.95	0.71	0.45	1.0	1.0	0.96	0.79	0.67	0.55	0.44	0.31
101	1.0	1.0	1.0	1.0	0.99	0.94	0.71	0.43	1.0	0.99	0.94	0.78	0.67	0.55	0.43	0.36
100	1.0	1.0	1.0	0.99	1.0	.095	0.71	0.45	1.0	0.99	0.95	0.79	0.64	0.51	0.46	0.36
011	1.0	1.0	0.99	0.98	0.90	0.71	0.40	0.28	0.99	0.94	0.81	0.65	0.6	0.44	0.37	0.27
010	1.0	1.0	1.0	0.98	089	0.73	0.43	0.26	1.0	0.95	0.82	0.64	0.5	0.44	0.40	0.25
001	1.0	1.0	0.98	0.96	0.90	0.73	0.40	0.25	0.99	0.94	0.83	063	0.54	0.43	0.37	0.27
000	1.0	1.0	0.99	0.98	0.90	0.72	0.42	0.27	0.99	0.95	0.80	0.65	0.53	0.45	0.35	0.23

if the latter fails and vice versa. This can be achieved by minimizing the following quantity:

$$-\mathbb{E}_{(\mathbf{x},y)\sim \mathrm{P}}[1_{h_i(\mathbf{x})\neq y} \log(1-\mathrm{p}(h_i(\mathbf{x})|\mathbf{x}, h_j)) + 1_{h_j(\mathbf{x})\neq y} \log(1-\mathrm{p}(h_j(\mathbf{x})|\mathbf{x}, h_i)]$$

with its empirical counterpart

$$J_{i,j}(\{(\mathbf{x}_m, y_m)\}_{m=1,\ldots,M}) = -\frac{1}{M}\sum_{m=1}^{M} 1_{h_i(\mathbf{x}_m)\neq y_m} \log(1-\mathrm{p}(h_i(\mathbf{x}_m)|\mathbf{x}_m, h_j)) + 1_{h_j(\mathbf{x}_m)\neq y_m} \log(1-\mathrm{p}(h_j(\mathbf{x}_m)|\mathbf{x}_m, h_i))\,, \tag{12}$$

where $\{(\mathbf{x}_m, y_m)\}_{m=1,\ldots,M}$ denotes a sample of M data points $(\mathbf{x}_m, y_m)$. In our experiments, we use a penalization coefficient/loss weight λ and then we add $\lambda \cdot J_{1,2}(\{(\mathbf{x}_m, y_m)\}_{m=1,\ldots,M})$ to the empirical cross-entropy loss. In further experiments not presented here, we also used other loss functions that explicitly enforce anti-correlated outputs or even independence of the softmax distributions. However, these loss functions led to uncontrollable behavior of the training pipeline, in particular when trying to tune the loss weight λ. Therefore, we present results for the loss function in (12).

Figure 1 (top) depicts the correlation as well as the average accuracy for different values of λ, ranging from 0 to extreme values of 200. For increasing values of λ, we observe a clear drop in performance from an initial average accuracy of more than 91% for $\lambda = 0$ down to roughly 72% for $\lambda = 200$. At the same time, the correlation decreased to values below 0.3 which, however, is still not enough to assume independence as being confirmed by our χ^2 tests. While the average accuracy monotonously decreases, it can be observed that the joint accuracy peaks around $\lambda = 7.5$; see Fig. 1 (bottom). The prediction of the DNN committee is pooled by summing up softmax

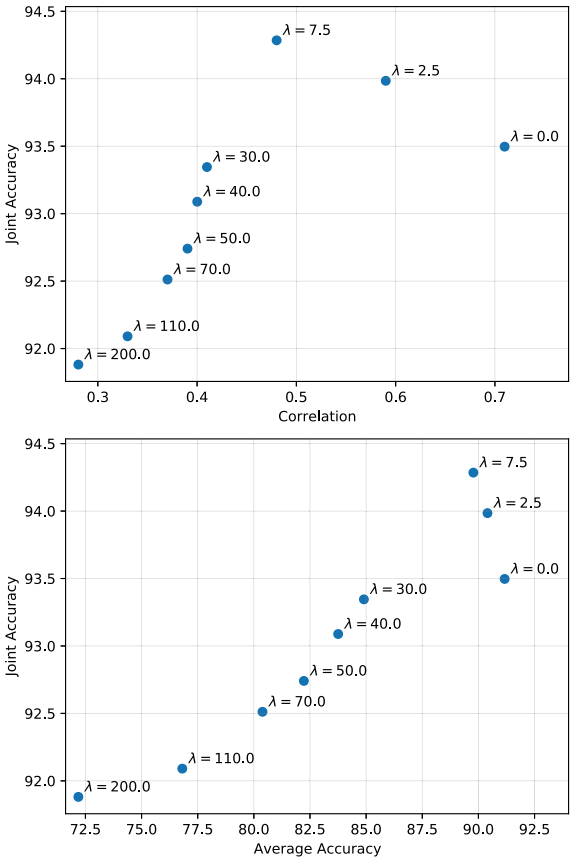


Fig. 1 Study of the influence of the loss weight λ on averaged accuracy and joint accuracy on the EMNIST dataset

probabilities over both DNN class-wise and then selecting the class with maximal sum. The joint accuracy is given by the accuracy of the committee with that decision rule. While the joint accuracy for an ordinarily trained committee with $\lambda = 0$ is about 93.5%, this can be improved by tenderly decorrelating the DNNs to a correlation coefficient around 0.5 which yields an increase of almost 1% point. At the same time, there is a mild decrease in average accuracy.

Concluding this section, we again study conditional independence for 8 softmax entropy bins (chosen according to equally distributed softmax entropy quantiles), this time comparing independence training $\lambda = 7.5$ with ordinary training $\lambda = 0$; see Table 4. We observe a considerable decrease in correlation in bin no. 8 (representing the highest softmax entropy) for the EMNIST data down to 0.07. In comparison, the decrease for CIFAR10 is rather mild. However, also this small correlation coefficient of 0.07 is not sufficient for assuming conditional independence.

Table 4 Correlation coefficients under independence training for different quantiles of softmax entropy computed on EMNIST and CIFAR10

	EMNIST								CIFAR10							
Entropy bin	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
λ	Correlation coefficients															
7.5	1.0	1.0	0.99	0.96	0.90	0.59	0.35	0.07	0.89	0.76	0.63	0.53	0.47	0.39	0.32	0.21
0.0	1.0	1.0	1.0	1.0	0.99	0.95	0.72	0.44	1.0	0.99	0.94	0.80	0.64	0.57	0.42	0.33

Training several networks for independence: We now present results when training an ensemble of $n = 5$ networks. To this end, we sum up the cross-entropy losses of the $n = 5$ models and add the penalization term

$$\lambda \cdot \frac{2}{n-1} \sum_{i=2}^{n} \sum_{j=1}^{i-1} J_{i,j}(\{(\mathbf{x}_m, y_m)\}_{m=1,\ldots,M}) \tag{13}$$

which is a straightforward combinatorial generalization of the previously used penalty term. Therein, λ again denotes the loss weight which varies during our experiments.

Besides k-out-of-5 accuracies, we consider also accuracies from single networks and ensemble accuracies. The corresponding ensemble prediction is obtained by summing up the softmax probabilities (via the class-wise sum over the first k ensemble members) and then taking the argmax.

Figure 2 depicts the results of our training for independence with 5 models in terms of k-out-of-5 accuracy. When stating mean accuracy, this refers to the average

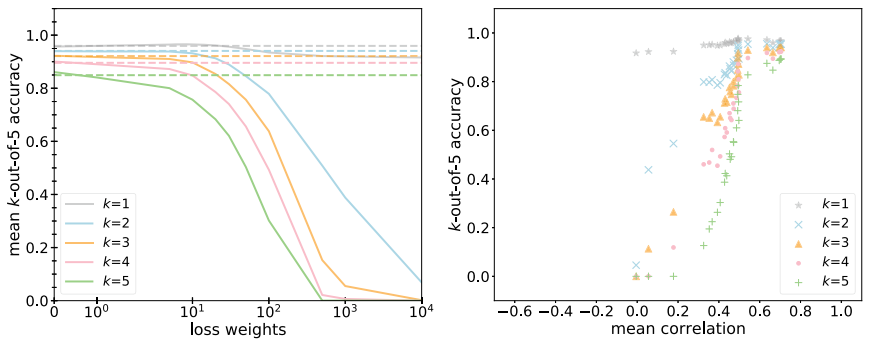


Fig. 2 Experiments with the EMNIST dataset. Left: mean k-out-of-5 accuracy (averaged over 30 repetitions) for an ensemble of five networks as a function of the loss weight λ. The solid lines depict the accuracies of the ensembles trained for independence with loss weight λ, the dashed lines depict baseline ensemble accuracies trained without incorporating the loss from (13). Right: k-out-of-5 accuracy for a single run as a function of the mean correlation (of each network with each other network)

Table 5 Correlation coefficients for an ensemble of 5 networks trained on EMNIST for independence and a baseline ensemble where each network was trained individually. All results are averaged over 30 runs

Loss weight λ		0	10^{-1}	10^{0}	10^{1}	10^{2}	Baseline model	Theoretical
Mean correlation		0.70	0.69	0.65	0.53	0.43	0.67	0.00
Single network accuracy	$k = 1$	0.92	0.92	0.91	0.88	0.73	0.91	
	$k = 2$	0.92	0.92	0.91	0.91	0.91	0.91	
	$k = 3$	0.92	0.92	0.91	0.88	0.60	0.91	
	$k = 4$	0.92	0.92	0.91	0.87	0.51	0.91	
	$k = 5$	0.92	0.92	0.91	0.85	0.39	0.91	
Ensemble accuracy/Mean k-out-of-5 accuracy	$k = 1$	0.92/0.96	0.92/0.96	0.91/0.96	0.88/0.97	0.73/0.93	0.91/0.96	1.00
	$k = 2$	0.92/0.94	0.92/0.94	0.92/0.94	0.92/0.93	0.91/0.78	0.92/0.94	0.99
	$k = 3$	0.92/0.92	0.92/0.92	0.92/0.92	0.92/0.90	0.90/0.64	0.92/0.92	0.96
	$k = 4$	0.93/0.90	0.93/0.90	0.92/0.89	0.92/0.85	0.90/0.49	0.92/0.90	0.72
	$k = 5$	0.93/0.86	0.93/0.86	0.92/0.84	0.92/0.76	0.90/0.30	0.92/0.85	0.32

of over 30 runs. For a loss weight of $\lambda = 10^1$, we observe in the left panel that the 1-out-of-5 accuracy of our independence training is slightly above the accuracy of the baseline ensemble which was trained regularly, i.e., each network was trained independently. Note that this is different from $\lambda = 0$, where all networks are still trained jointly with a common loss function which is the sum of the cross-entropy losses. The right-hand panel shows that the 1-out-of-5 accuracy peaks at a mean correlation (the average correlation over the errors of all networks $i < j$) of 0.4. Decorrelating the networks' errors further toward zero-mean correlation is possible, however, the 1-out-of-5 accuracy decreases. The k-out-of-5 accuracy for $k > 1$ suffers even more from decorrelating the networks' errors. Note that, in practice, 1-out-of-5, e.g., for a pedestrian detection, might additionally suffer from overproduction of false positives and could be impractical. That hypothesis is indeed supported by Table 5, which shows that for larger loss weights $\lambda \geq 1$ the individual network accuracies become heterogeneous. In particular, network 5 suffers from extremely low accuracy at $\lambda = 10^2$, which is, however, still far away from zero-mean correlation. For the sake of completeness, we give two examples of correlations between the individual models since we only reported mean correlations so far; see Fig. 3. Comparing the right-hand panel with Table 5, we see that the well-performing network no. 2 exhibits comparatively small correlation coefficients with the other networks' errors. Surprisingly, the worse performing models' errors show higher correlation coefficients which reveals that in that case they often err jointly on the same input

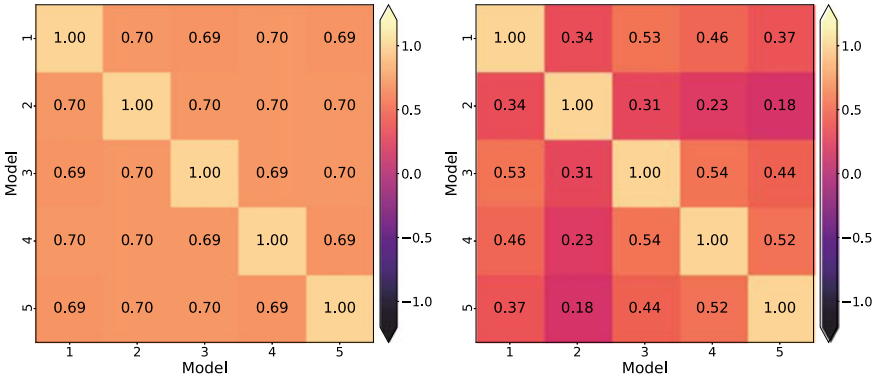


Fig. 3 Correlation coefficients of the networks' errors for all combinations of 2 out of 5 models. Left: loss weight $\lambda = 10^{-1}$. Right: loss weight $\lambda = 10^{2}$

examples. Additionally, we provide theoretical k-out-of-5 accuracies according to (10) where we choose p_{sub} equal to 1 minus the average ensemble accuracy (which is 92.45%). In particular, the k-out-of-5 accuracies of the ensemble for $k = 1$ and 2 are clearly below the theoretical ones.

We conclude that independence training may help slightly to improve the performance, however, the benefit seems to be limited when all networks are trained with the same input. Besides these tests, we considered an additional loss function that explicitly penalizes the correlation of the networks' errors. That approach, being actually more directed toward our goal, even achieved negative correlation coefficients of the networks' errors. It was also able to slightly improve the ensembles' performance in terms of k-out-of-n accuracy over the baseline, however, this improvement was even less pronounced than that one reported in this section. Thus, we do not report those results here.

Training of independence for different input sensors: In order to conduct further experiments on the scale of the EMNIST and CIFAR10 datasets, we consider the 3D-MNIST dataset that provides synthesized 3D representations of handwritten digits. We apply rotations around the x-axis with chosen angles. To this end, we create 5 rotated variants of the dataset with a randomly chosen but fixed angles $\theta = a\pi/9$, $a = 1, \ldots, 9$; see Fig. 4 for an illustrative example. Each of our $k = 1, \ldots, 5$ networks obtains one of the 5 rotated variants of the data with a given angle θ_k, and is then trained. For the baseline, all networks are again trained independently. For independence training, all networks are trained with common loss functions, as described previously, and the same handwritten digit, however, viewed from a different angle θ_k, is presented to the network k, $k = 1, \ldots, 5$.

Figure 5 shows results of numerical experiments conducted analogously to those presented in Fig. 2. Comparing both figures with each other, we observe that the baseline k-out-of-5 accuracies are much lower than when presenting identical images to the networks. Indeed, although MNIST classification constitutes a much simpler

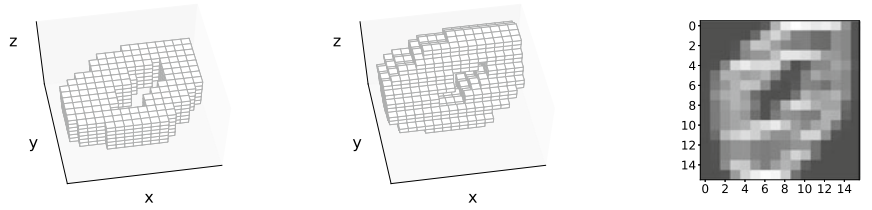


Fig. 4 A 3D-MNIST example data point. Left: original input data. Center: input data rotated by $\pi/3$ along the x-axis. Right: 2D projection of the rotated data, which is obtained by summation and normalization along the y-axis

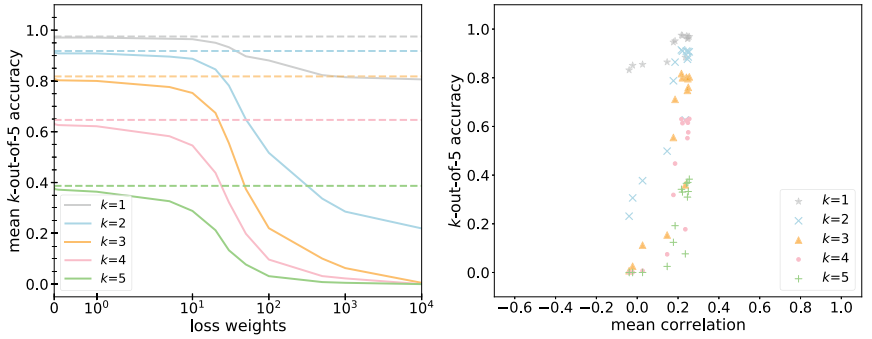


Fig. 5 Experiments with the 3D-MNIST dataset. Left: mean k-out-of-5 accuracy (averaged over 30 repetitions) for an ensemble of five networks as a function of the loss weight λ. The solid lines depict the accuracies of the ensembles trained for independence with loss weight λ, the dashed lines depict baseline ensemble accuracies trained without incorporating the loss from (13). Right: k-out-of-5 accuracy for a single run as a function of the mean correlation (of each network with each other network)

task than classifying the letters from EMNIST; this 3D variant shows much lower individual network performances, not only indicated by the left-hand panel of Fig. 5, but also by Table 6. On the other hand, the highest mean correlation (obtained for the loss weight $\lambda = 0$) for 3D-MNIST depicted by the right-hand panel of Fig. 5 is below 0.3 and therefore much lower than the one depicted by Fig. 2.

Neglecting the reduced performance on 3D-MNIST, these results show that an ensemble, wherein each network obtains a different input, can be clearly improved by increasing the number of ensemble members. The networks only exhibit small correlations among each other. However, it seems that independence training cannot contribute to the performance of the ensemble anymore in the given setup. It remains open, whether independence training may help in a realistic setting with different sensors for perception in automated driving and networks conducting way more difficult tasks such as object detection, instance segmentation, semantic segmentation, or panoptic segmentation. Note that the correlation strengths reported in our experiments in accordance with Sect. 3 do not suffice to substantially reduce the data requirements into a feasible regime. Recalling the discussion in Sect. 3.2, even

Table 6 Results for an ensemble of 5 networks trained on 3D-MNIST for independence and a baseline ensemble where each network was trained individually. All results are averaged over 30 runs

Loss weight λ		0	10^{-1}	10^{0}	10^{1}	10^{2}	Baseline model	Theoretical
Mean correlation		0.25	0.25	0.24	0.22	0.15	0.24	0.00
Single network accuracy	$k = 1$	0.70	0.70	0.70	0.66	0.38	0.70	
	$k = 2$	0.74	0.73	0.74	0.73	0.75	0.75	
	$k = 3$	0.73	0.72	0.72	0.67	0.25	0.74	
	$k = 4$	0.81	0.80	0.79	0.73	0.17	0.82	
	$k = 5$	0.72	0.73	0.71	0.64	0.20	0.74	
Ensemble accuracy/mean k-out-of-5 accuracy	$k = 1$	0.70/0.97	0.70/0.97	0.70/0.97	0.66/0.96	0.38/0.88	0.70/0.97	1.00
	$k = 2$	0.82/0.91	0.82/0.91	0.82/0.91	0.81/0.89	0.77/0.52	0.82/0.92	0.98
	$k = 3$	0.84/0.81	0.84/0.80	0.84/0.80	0.83/0.75	0.78/0.22	0.85/0.82	0.90
	$k = 4$	0.87/0.63	0.86/0.63	0.86/0.62	0.85/0.55	0.79/0.10	0.87/0.65	0.63
	$k = 5$	0.88/0.38	0.88/0.37	0.88/0.37	0.86/0.29	0.79/0.03	0.89/0.39	0.24

if the subsystem performance remained unaffected by independence training, based on the lowest correlation observed in our experiments, we could at most expect a reduction of the required data amount by one order of magnitude.

5 Conclusion and Outlook

Summary and main take-away messages: In this chapter, we argued that obtaining statistical evidence from brute-force testing leads to the requirement of infeasible amounts of data. In Sects. 2 and 3, we estimated upper and lower bounds on the amount of data required to test a given perception system with statistical validity for being significantly safer than a human driver. We restricted our considerations to fatalities and arrived at data amounts that are infeasible from both storage and labeling cost perspectives, as already found in [KP16]. In Sect. 3.1, we showed that perfectly uncorrelated redundant AI perception systems could be used to resolve the problem. However, as we have seen in Sect. 3.2, redundant subsystems that are not perfectly uncorrelated require extremely low correlation coefficients between error events produced by neural networks to yield substantial reductions in data requirement. In Sect. 3.3, we furthermore found the amount of data needed to prove

a sufficiently low correlation as large as the amount of data needed for the original test.

In Sect. 4, we present numerical results on the correlation of redundant subsystems. We studied correlation coefficients between error events of redundant neural networks dependent on network architecture, training data, and weight initializers. Furthermore, we trained ensembles of neural networks to become decorrelated. Besides studying correlation coefficients, we considered the system's performance in terms of 1-out-of-n as well as k-out-of-n performance. In the numerical experiments, we obtained correlation coefficients that would allow for a reduction in the amount of data required by at most one order of magnitude. For the testing problem, this would still represent an infeasible data requirement. However, redundancy could contribute to a moderate reduction of the amount of data required for testing and potentially be combined with other approaches.

There are alternative approaches to test the safety of perception systems other than brute-force testing. Here, we give a short outlook on two of them that are very actively developed in the research community.

Outlook on testing with synthetic data: One possibility to obtain vast amounts of data for testing is to consider synthetic sources of data. The question of whether synthetic data can be used for testing has already been addressed, e.g., in [RBK+21]. In principle, arbitrary amounts of test data can be created from a driving simulation such as CARLA [DRC+17]. The domain shift between synthetic and real data can be bridged with generative adversarial networks (GANs). The latter have been proven to be learnable in the large sample limit [BCST20, AGLR21], meaning that for increasing amounts of data and capacity of the generator and discriminator, the learned map from synthetic to real data is supposed to converge to the true one. Combining synthetic data and adversarial learning is therefore a promising candidate for testing DNNs. However, in this setup there remain other gaps to be bridged (number of assets, environmental variability, and infeasibility of the empirically risk-minimizing generator).

Outlook on testing with real data: There exists a number of helpful approaches to estimating the performance of unlabeled data. A DNN of strong performance (or ensembles of those) can be utilized to compute pseudo ground truth. The model to be equipped with an AI perception system can be learned in a teacher-student fashion [AHKS19, BHSFs19], and the discrepancies between the student model and the teachers can be compared with errors on a moderate subset with ground truth, for instance, in terms of correlations of errors and discrepancies. Furthermore, in order to process the vast amounts of recorded data and perform testing more efficiently, well-performing uncertainty quantification methods [KG17, KRPM+18, RCH+20] in combination with corner case detection methods [BBLFs19, HBR+21] can help to pre-select data for testing. Besides that, many additional approaches toward improving the reliability of DNN exist. However, while a big number of tools already exist, their proper application to DNN testing and inference of statistically relevant statements on the system's safety still require thorough research.

These approaches and other upcoming research might be part of the solution to the testing problem in future.

Concluding remark: As a concluding remark, this chapter does not intend to discourage safety arguments, as also conceptualized in this volume. We do not deny the value of empirical evidence in order to, e.g., prove the relative superiority of one AI system over another with regards to safety. The inherent difficulty to provide direct evidence for the better-than-human safety of automated driving as required by the ethics committee of the German Ministry of Transportation and Digital Infrastructure [FBB+17] should not be mistaken as an excuse for a purely experimental approach. Bringing automated vehicles to the street without prior risk assessment implies that risks would be judged a posteriori based on the experience with a large fleet of automated vehicles.

Such matters attain urgency in the light of recent German legislation on the experimental usage of automated driving under human supervision[5] which only refers to the technical equipment for the sensing of automated vehicles, but does not specify the minimal performance for the AI-based perception based on the sensor information. Related regulations in other countries face similar problems.[6]

The debate on how to ensure a safe transition to automated driving that complies with high ethical standards, therefore, remains of imminent scientific and public interest.

Acknowledgements The authors thank Lina Haidar for support and numerical results in Section 4.2. Financial support by the German Federal Ministry of Economic Affairs and Energy (BMWi) via Grant No. 19A19005R as a part of the Safe AI for Automated Driving consortium is gratefully acknowledged.

References

[AGLR21] H. Asatryan, H. Gottschalk, M. Lippert, M. Rottmann, A convenient infinite dimensional framework for generative adversarial learning, pp. 1–29 (2021). arXiv:2011.12087

[AHKS19] S. Abbasi, M. Hajabdollahi, N. Karimi, S. Samavi, Modeling teacher-student techniques in deep neural networks for knowledge distillation, pp. 1–6 (2019). arXiv:1912.13179

[BBLFs19] J.-A. Bolte, A. Bär, D. Lipinski, T. Fingscheidt, Towards corner case detection for autonomous driving, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 438–445, Paris, France (2019)

[BCST20] G. Biau, B. Cadre, M. Sangnier, U. Tanielian, Some theoretical properties of GANs. Ann. Stat. **48**(3), 1539–1566 (2020)

[BHSFs19] A. Bär, F. Hüger, P. Schlicht, T. Fingscheidt, On the robustness of redundant teacher-student frameworks for semantic segmentation, in *Proceedings of the IEEE/CVF*

[5] https://dserver.bundestag.de/btd/19/274/1927439.pdf.

[6] Framework for Automated Driving System Safety, No. NHTSA-2020-0106, 49 CFR Part 571 (Nov. 19, 2020).

Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 1380–1388, Long Beach, CA, USA (2019)

[BTLFs20] J. Breitenstein, J.-A. Termöhlen, D. Lipinski, T. Fingscheidt, Systematization of corner cases for visual perception in automated driving, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 1257–1264, Virtual Conference (2020)

[BTLFs21] J. Breitenstein, J.-A. Termöhlen, D. Lipinski, T. Fingscheidt, Corner cases for visual perception in automated driving: some guidance on detection approaches, pp. 1–8 (2021). arXiv:2102.05897

[C+15] F. Chollet, et al. Keras (2015). Accessed 18 Nov 2021

[CATvS17] G. Cohen, S. Afshar, J. Tapson, A. van Schaik, EMNIST: an extension of MNIST to handwritten letters, pp. 1–10 (2017). arXiv:1702.05373

[DRC+17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: an open urban driving simulator, in *Proceedings of the Conference on Robot Learning CORL*, pp. 1–16, Mountain View, CA, USA (2017)

[FBB+17] U. di Fabio, M. Broy, R.J. Brüngger, U. Eichhorn, A. Grunwald, D. Heckmann, E. Hilgendorf, H. Kagermann, A. Losinger, M. Lutz-Bachmann, A. Markl, K. Müller, K. Nehm, Ethik-Kommission - Automatisiertes und vernetztes Fahren. Technical report, Federal Ministry of Transport and Digital Infrastructure (2017)

[Ger20] German Federal Ministry for Transortation & Infrastructure. Verkehr in Zahlen 2020/2021. Deutsches Zentrum für Luft- und Raumfahrt (2020)

[HBR+21] F. Heidecker, J. Breitenstein, K. Rösch, J. Löhdefink, M. Bieshaar, C. Stiller, T. Fingscheidt, B. Sick, An application-driven conceptualization of corner cases for perception in highly automated driving, pp. 1–8 (2021). arXiv:2103.03678

[HS16] J. Hedderich, L. Sachs, *Angewandte Statistik (in German)* (Springer, Berlin, 2016)

[KB15] D.P. Kingma, J. Ba, ADAM: a method for stochastic optimization, in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–15, San Diego, CA, USA (2015)

[KG17] A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision? in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, pp. 5574–5584, Long Beach, CA, USA (2017)

[KP16] N. Kalra, S.M. Paddock, Driving to safety: how many miles of driving would it take to demonstrate autonomous vehicle reliability? Transp. Res. Part A: Policy Pract. **94**, 182–193 (2016)

[Kri09] A. Krizhevsky, Object classification experiments. Technical Report, Canadian Institute for Advanced Research (2009)

[KRPM+18] S. Kohl, B. Romera-Paredes, C. Meyer, J. De Fauw, J.R. Ledsam, K. Maier-Hein, S.M. Ali Eslami, D. Jimenez Rezende, O. Ronneberger, A probabilistic U-Net for segmentation of ambiguous images, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, pp. 6965–6975, Montréal, QC, Canada (2018)

[LCB98] Y. LeCun, C. Cortes, C.J.C. Burges, The MNIST database of handwritten digits (1998). Assessed 18 Nov 2021

[LL88] G.R. Loftus, E.F. Loftus, *Essence of Statistics* (Brooks/Cole Publishing Co, Salt Lake City, 1988)

[LWC+19] L. Liu, W. Wei, K.-H. Chow, M. Loper, E. Gursoy, S. Truex, Y. Wu, Deep neural network ensembles against deception: ensemble diversity, accuracy and robustness, in *Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 274–282, Monterey, CA, USA (2019)

[ME14] W.Q. Meeker, L.A. Escobar, *Statistical Methods for Reliability Data* (Wiley, New York, 2014)

[Nat20] National Transportation Safety Board NSTB. Collision Between a Sport Utility Vehicle Operating With Partial Driving Automation and a Crash Attenuator (2020). Accessed 18 Nov 2021

[RBK+21] J. Rosenzweig, E. Brito, H.-U. Kobialka, M. Akila, N.M. Schmidt, P. Schlicht, J.D. Schneider, F. Hüger, M. Rottmann, S. Houben, T. Wirtz, Validation of simulation-based testing: bypassing domain shift with label-to-image synthesis, pp. 1–8 (2021). arXiv:2106.05549

[RCH+20] M. Rottmann, P. Colling, T.-P. Hack, R. Chan, F. Hüger, P. Schlicht, H. Gottschalk, Prediction error meta classification in semantic segmentation: detection via aggregated dispersion measures of softmax probabilities, in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, Virtual Conference (2020)

[Sch19] F.A. Schmidt, *Crowdproduktion von Trainingsdaten: Zur Rolle von Online-Arbeit beim Trainieren autonomer Fahrzeuge*, vol. 417 of *Study (in German)* (Hans-Böckler-Stiftung, 2019)

[WLX+20] Y. Wu, L. Liu, Z. Xie, J. Bae, K.-H. Chow, W. Wei, Promoting high diversity ensemble learning with ensemblebench, in *Proceedings of the IEEE International Conference on Cognitive Machine Intelligence (CogMI)*, pp. 208–217, Atlanta, GA, USA (2020)

[Zac92] S. Zacks, *Introduction to Reliability Analysis: Probability Models and Statistical Methods* (Springer, Berlin, 1992)

Analysis and Comparison of Datasets by Leveraging Data Distributions in Latent Spaces

Hanno Stage, Lennart Ries, Jacob Langner, Stefan Otten, and Eric Sax

Abstract Automated driving is widely seen as one of the areas, where key innovations are driven by the application of deep learning. The development of safe and robust deep neural network (DNN) functions requires new validation methods. A core insufficiency of DNNs is the lack of generalization for out-of-distribution datasets. One path to overcome this insufficiency is through the analysis and comparison of the domains of training and test datasets. This is important because otherwise, deep learning cannot advance automated driving. Variational autoencoders (VAEs) are able to extract meaningful encodings from datasets in their latent space. This chapter examines various methods based on these encodings and presents a broad evaluation on different automotive datasets and potential domain shifts, such as weather changes or new locations. The used methods are based on the distance to the nearest neighbors between datasets and leverage several network architectures and metrics. Several experiments with different domain shifts on different datasets are conducted and compared with a reconstruction-based method. The results show that the presented methods can be a promising alternative to the reconstruction error for detecting automotive-relevant domain shifts between different datasets. It is also shown that VAE loss variants that focus on a disentangled latent space can improve the stability of the domain shift detection quality. Best results were achieved with nearest neighbor methods using VAE and JointVAE, a VAE variant with a discrete and a continuous

H. Stage (✉) · L. Ries · J. Langner · S. Otten · E. Sax
FZI Research Center for Information Technology, Haid-und-Neu-Str. 10-14,
76131 Karlsruhe, Germany
e-mail: stage@fzi.de

L. Ries
e-mail: ries@fzi.de

J. Langner
e-mail: langner@fzi.de

S. Otten
e-mail: otten@fzi.de

E. Sax
e-mail: sax@fzi.de

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_3

latent space, in combination with a metric based on the well-known z-score, and with the NVAE, a VAE variant with optimizations regarding reconstruction quality, in combination with the deterministic reconstruction error.

1 Introduction

Artificial intelligence (AI) and deep learning are key enablers in automated driving [RMM21]. One important task in the development of automated cars is the perception of the vehicle environment. While several sensors can be used to solve this task, the most promising approaches in industry and research rely at least partly on camera-based perception [RNFP19]. In the perception sub-tasks of object detection and semantic segmentation, deep learning-based approaches already outperform classical methods [GLGL18, ZZXW19].

However, those deep learning-based approaches suffer from several insufficiencies. Following Sämann et al., one central insufficiency is the ability to generalize from given training data [SSH20]. If the underlying distribution of the input data differs from the distribution of the training dataset, the prediction accuracy may decrease.

In the context of automated driving, a decrease in prediction accuracy can have dramatic consequences, e.g., fatal accidents due to traffic participants which were not detected, e.g., Uber accident [SSWS19]. One possibility to decrease the probability of these consequences is the optimization of used datasets. A high coverage over all possible situations in the training data is crucial to improve the quality of trained neural networks. However, since the input spaces (e.g., image data) are very high-dimensional and complex, the rule-based assessment of coverage is not feasible [BLO+17].

Hence, in this work, the capabilities of variational autoencoders (VAEs) of extracting meaningful distributions from datasets are used to compare different datasets in an unsupervised fashion. Since the VAE works directly with raw data, no assumptions over relevant dataset dimensions for the comparison are necessary. Additionally, no already trained perception function is necessary. The underlying data distributions found by the VAE are analyzed to find similarities and novelties, using the latent space of the VAE. This makes it possible to estimate whether a particular dataset is in-domain or out-of-domain of a second dataset, thus allowing the detection of *domain shifts* between datasets.

During early function development, the presented methods can be used to extract datasets with manageable sizes, still maintaining high coverages over relevant information in the complete data pool. These smaller datasets lead to faster training and validation and allow faster iterations. Moreover, the method can detect an unknown domain shift between the training, validation, and test datasets, which could lead to wrong conclusions about the performance of the function. In later development stages, the dataset comparison enables a prediction, if an already trained function

can be safely used in a different environment. Additionally, for domain shift detections with very high confidences, an online parallel execution of a perception neural network is thinkable, acting as a warning system for possibly unknown input data.

A domain shift can be caused by a huge variety of factors, some of them potentially unknown to the developer. This makes the application of rule-based approaches impossible. So, the idea of the presented dataset comparison method is to learn a multivariate distribution for all relevant aspects of one dataset and find useful metrics to compare these distributions. The used metrics in this investigation are related to known novelty detection techniques for VAEs.

The main contribution of this chapter is the application of various novelty detection methods and various VAE architectures in several experiments, each testing a different kind of domain shift. The applied novelty detection methods comprise five different nearest neighbor approaches applied to the latent spaces. As a benchmark, the reconstruction error is also computed [VGM+20]. The applicability of these methods is investigated on three datasets, each providing one or more inherent domain shifts (e.g., weather, location, or daytime). The presented broad evaluation of different approaches can give insights into the practical application of such approaches in the development of safe perception functions for automated driving.

2 Related Works

In this section, we provide an overview of related topics. There are various ways to useVAEs for anomaly detection in datasets [AC15, SWXS18, VGM+20]. An et al. compute the reconstruction probability of a given test sample by inspecting several samples drawn from the latent distribution of test samples [AC15]. The underlying assumption is that anomalous data samples have a higher variance and will therefore show a lower reconstruction quality. Vasilev et al. propose several methods and metrics to compute novelty scores using VAEs [VGM+20]. They describe 19 different methods for novelty calculation, including latent space-based and reconstruction-based approaches. Their evaluation on a magnetic resonance imaging dataset and on MNIST showed that a variety of metrics outperforms classical approaches such as nearest neighbor distance or principal component analysis (PCA) in the feature space.

In the domain of skin disease detection, VAE-based novelty detection methods were applied by Lu et al. [LX18]. They decompose the VAE loss into the reconstruction term and the KL term and use these as a novelty metric. Their results show better performance of the reconstruction-based approaches.

In the automotive domain, the issue of designing meaningful test datasets is addressed on various levels. Bach et al. address the selection of well-adjusted datasets for testing with a visual method for time series in recorded real-world data [BLO+17]. Langner et al. enhanced this approach with a method for dataset design, using the novelty detection capabilities of classical autoencoders [LBR+18]. A related approach

uses methods from natural language processing (NLP) to describe sequential contexts in automotive datasets, allowing similarity analysis and novelty detection [RSBS20].

These approaches can only indirectly find domain shifts for perception functions, since they are not designed to work in the feature space itself. For the detection of domain shifts directly in raw data, Löhdefink et al. propose the analysis of the reconstruction error of an autoencoder to predict the quality of a semantic segmentation network [LFK+20]. Sundar et al. address the issue of the diversity of automotive data by training various β-VAEs using different labels such as weather or daytime [SRR+20].

There is a lot of research in relation to application domains, proposed metrics and network architectures in the area of dataset analysis with VAEs. However, most of the related approaches restrict their analysis on datasets with only a few degrees of freedom (e.g., [VGM+20] and [LX18]) or test their method on only one type of domain shift (e.g., different locations). In this work, a broad spectrum of approaches (different network architectures, different metrics) were evaluated on three datasets including five different domain shifts.

3 Building Blocks of Our Approach

In this work, four different VAE architectures and five different metrics for anomaly detection are examined on their applicability to compare automotive datasets. Four methods use a nearest neighbor approach in the latent space for detection of domain shifts.

In this section, the VAE DNN is introduced. Afterwards, the used novelty detection methods are presented, and the section concludes with the explanation for dataset comparison and domain shift detection.

3.1 Variational Autoencoder DNN

The difference between the original autoencoder and the VAE is that the VAE uses a stochastic model. The VAE consists of a stochastic encoder, a latent space, and a stochastic decoder. The general structure of a VAE can be seen in Fig. 1. Simplified, one could say that the VAE transfers the input data into a low-dimensional space to then reconstruct the input data from this space. The underlying stochastic model is now explained.

The probabilistic generative model of the VAE $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is a deep neural network (DNN) with network parameters $\boldsymbol{\theta}$ mapping $\mathbf{z}$ into $\mathbf{x}$ and is named decoder [VGM+20, Dup18]. This decoder model learns a joint distribution of the data $\mathbf{x}$ and a continuous latent variable $\mathbf{z} \in \mathbb{R}^{d_{\text{lat}}}$ [CHK21]. In other words, the decoder describes the distribution of the decoded variable $\hat{\mathbf{x}}$ given the encoded one $\mathbf{z}$. The inference model of the

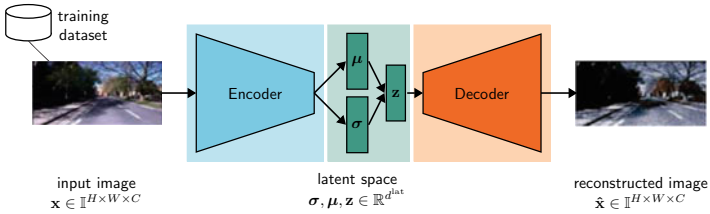


Fig. 1 Basic variational autoencoder structure, consisting of an encoder to transform data into the latent space and a decoder to reconstruct images from sampled values **z**. The encoder receives an image **x** as input, where H is the height of the image, W is the width of the image, and C is the number of color channels

VAE is trained to approximate the posterior distribution $\mathrm{q}_\phi(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}),\ \sigma^2(\mathbf{x}))$ with parameters ϕ mapping **x** into **z** and is named encoder [VGM+20]. The encoder describes the distribution of the encoded variable given the decoded one.

The probabilistic generated model and the inference model then give rise to the first loss term, since the aim for the input data of the encoder is to be reconstructed by the decoder. The reconstruction error loss is defined as follows: $\mathbb{E}_{\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})}[\log \mathrm{p}_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]$ [Dup18]. In order to improve the disentanglement properties of $\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})$, the VAE has a loss optimization that tries to fit $\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})$ to $\mathrm{p}(\mathbf{z})$ [HMP+17]. This loss optimization can be both controlling the capacity of the latent information bottleneck and embodying statistical independence. This is achieved by defining $\mathrm{p}(\mathbf{z})$ as an isotropic unit Gaussian distribution $\mathrm{p}(\mathbf{z}) = \mathcal{N}(0, I)$ [HMP+17]. In order to train the disentanglement in the bottleneck, the VAE uses the Kullback–Leibler (KL) divergence of the approximate from the true posterior $D_{KL}(\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})||\mathrm{p}(\mathbf{z}))$. Before we define the loss term, we introduce the variable β, which serves as a weighting factor. This allows us to define the loss function for the VAE and β-VAE as follows:

$$J(\boldsymbol{\theta}, \phi) = \mathbb{E}_{\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})}[\log \mathrm{p}_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - \beta D_{KL}(\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})||\mathrm{p}(\mathbf{z})). \tag{1}$$

The first term of the loss function is the reconstruction error, which is responsible for the VAE encoding the informative latent variable **z** and allowing the input data **x** to be reconstructed [CHK21]. The second term regulates the posterior distribution by adjusting the distribution of the encoded latent variables $\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})$ to the prior $\mathrm{p}(\mathbf{z})$ [CHK21]. If $\beta > 1$ is selected, the loss function of the β-VAE as shown in (1) is used, and if $\beta = 1$ is selected, (1) breaks down to the standard VAE loss [HMP+17]. The variable β scales the regularization term for the posterior distribution, with $\beta > 1$ enforcing the β-VAE to encode more disentangled latent variables by matching the encoded latent variables with the prior by higher pressure on $D_{KL}(\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})||\mathrm{p}(\mathbf{z}))$ [HMP+17, CHK21].

The VAE and β-VAE only use continuous latent variables to model the latent variable **z** [CHK21]. The next VAE architecture that we use for our experiments additionally uses discrete variables to disentangle the generative factors of the observed data [CHK21]. The JointVAE uses the discrete variable **c** to decode the generative

factors of the observed data [Dup18, CHK21]. The loss function of the JointVAE is an extension of the loss function of the β-VAE and results in (2).

$$J(\theta, \phi) = \mathbb{E}_{\mathrm{q}_\phi(\mathbf{z},\mathbf{c}|\mathbf{x})}[\log \mathrm{p}_\theta(\mathbf{x}|\mathbf{z}, \mathbf{c})] - \gamma\left|D_{KL}(\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})||\mathrm{p}(\mathbf{z})) - C_\mathbf{z}\right| - \gamma\left|D_{KL}(\mathrm{q}_\phi(\mathbf{c}|\mathbf{x})||\mathrm{p}(\mathbf{c})) - C_\mathbf{c}\right|. \tag{2}$$

The term $D_{KL}(\mathrm{q}_\phi(\mathbf{c}|\mathbf{x})||\mathrm{p}(\mathbf{c})))$ is the Kullback–Leibler divergence for the discrete latent variables and $D_{KL}(\mathrm{q}_\phi(\mathbf{z}|\mathbf{x})||\mathrm{p}(\mathbf{z}))$ is the Kullback–Leibler divergence for the continuous latent variables [Dup18]. The variables γ, $C_\mathbf{z}$, and $C_\mathbf{c}$ are hyperparameters [Dup18]. The hyperparameters $C_\mathbf{z}$ and $C_\mathbf{c}$ are gradually increased during training and both hyperparameters control the amount of information the model can encode [Dup18]. The hyperparameter γ forces the KL divergence to match the capacity $C_\mathbf{z}$ and $C_\mathbf{c}$. As with the standard VAE, $\mathrm{q}_\phi(\mathbf{z}, \mathbf{c}|\mathbf{x})$ is the inference model which maps $\mathbf{x}$ into $\mathbf{z}$ and $\mathbf{c}$.

In contrast to β-VAE and JointVAE which are focused on optimized latent space distributions, the goal of the Nouveau VAE (NVAE) is to reconstruct the best possible images [VK21]. Compared to the other used VAEs, the NVAE is a hierarchical VAE. This means that the NVAE has a hierarchical encoder and decoder structure and multiple latent spaces on different stages. Another special property is that this VAE has a bidirectional encoder [VK21]. This means that the information does not only pass from one encoder to the other but also the information from a deeper encoder passed back to the higher encoder. A focus during the development of the NVAE was to design expressive neural networks for VAEs [VK21].

In this chapter, the VAE [KW14], β-VAE [HMP+17], JointVAE [Dup18], and Nouveau VAE (NVAE) [VK21] are considered for the five experiments.

3.2 Methods for Novelty Detection

Nearest neighbor approaches for novelty detection have the assumption that the distributions of new and unknown data are different from the distributions of normal data. So, the latent space distance between a test sample distribution p_t and the closest normal sample can be used as a novelty score. Since latent spaces consist of high-dimensional distributions, the selection of a feasible metric to compute meaningful distances is non-trivial. According to Sect. 3.1, the used normal distributions p_n are defined by their parameters $\boldsymbol{\mu}_n$ and $\boldsymbol{\sigma}_n$. As the JointVAE has different latent spaces, for the calculation of the nearest neighbor only the continuous latent space was used.

In this chapter, five metrics in the latent space and one metric in the pixel space are evaluated. The five metrics in the latent space can be further divided into two groups. Group one includes the metrics that only use the means ($\boldsymbol{\mu}$) of the distribution, and metric group two uses both means ($\boldsymbol{\mu}$) and variance ($\boldsymbol{\sigma}$) for the calculation.

First, the metrics from group one are presented that only use means for the calculation. The cosine distance is a widely used metric to determine the similarity of

vectors [NB10, Ye11]. Since it is designed for vectors, it can not directly be applied to distributions. Therefore, only the means $\boldsymbol{\mu}$ of the normal distributions are used for calculating the cosine distance. The cosine distance can be computed by

$$m_{\cos}(\mathrm{p}_n, \mathrm{p}_t) = \frac{\boldsymbol{\mu}_n \boldsymbol{\mu}_t^T}{\|\boldsymbol{\mu}_n\| \|\boldsymbol{\mu}_t\|}, \tag{3}$$

where $\boldsymbol{\mu}_n = (\mu_n(\delta))$ and $\boldsymbol{\mu}_t = (\mu_t(\delta))$ describe the d-dimensional mean vectors of two distributions p_n and p_t, related to a normal and a test sample.

The Euclidean distance is the distance that can be measured with a ruler between two points and is defined as

$$m_{\mathrm{euc}}(\mathrm{p}_n, \mathrm{p}_t) = \sqrt{\sum_{\delta=1}^{d} (\mu_n(\delta) - \mu_t(\delta))^2}. \tag{4}$$

Here and in the following definitions, d is the dimensionality of the latent space. The Manhattan distance is inspired by the quadratic layout of Manhattan, New York City, and measures the distance between two vectors as the sum of their components' absolute differences [BR14].

$$m_{\mathrm{man}}(\mathrm{p}_n, \mathrm{p}_t) = \sum_{\delta=1}^{d} |\mu_n(\delta) - \mu_t(\delta)|. \tag{5}$$

The former presented metrics only used the means $\boldsymbol{\mu}$. Next, two metrics are introduced that use both means $\boldsymbol{\mu}$ and standard deviations $\boldsymbol{\sigma}$ for the calculation of the nearest neighbor. The next metric was defined by the authors of this chapter, using the well-known z-score. From the test samples' distribution, a vectorial value $\mathbf{z}_t = (z_t(\delta))$ is sampled. Then the distance is computed via

$$m_{\mathrm{zsc}}(\mathrm{p}_n, \mathrm{p}_t) = \sum_{\delta=1}^{d} \left\| \frac{z_t(\delta) - \mu_n(\delta)}{\sigma_n(\delta)} \right\|^2, \tag{6}$$

where the difference between $\mathbf{z}_t$ and the normal sample mean $\boldsymbol{\mu}_n$ being element-wise divided by its standard deviation $\boldsymbol{\sigma}_n$. Since we are only interested in the absolute distance of the distributions, we compute the squared L2 norm. The Bhattacharyya distance

$$m_{\mathrm{bha}}(\mathrm{p}_n, \mathrm{p}_t) = \sum_{\delta=1}^{d} \frac{1}{4} \cdot \log\left(\frac{1}{4} \left(\frac{\sigma_n^2(\delta)}{\sigma_t^2(\delta)} + \frac{\sigma_t^2(\delta)}{\sigma_n^2(\delta)} + 2 \right) \right) + \frac{1}{4} \left(\frac{(\mu_n(\delta) - \mu_t(\delta))^2}{\sigma_n^2(\delta) + \sigma_t^2(\delta)} \right), \tag{7}$$

as proposed by Vasilev et al. [VGM+20], is also tested. The Bhattacharyya distance measures the similarity of two distributions [CA79].

To consider also a metric which does not use the latent space, the reconstruction error in the pixel space is considered. Since the VAE is trained only with normal data, the reconstruction quality should decrease, when new data samples are processed. Specifically, we chose the deterministic reconstruction error

$$m_{\text{dre}}(\mathbf{x}, \hat{\mathbf{x}}) = \left\| \mathbf{x} - \hat{\mathbf{x}} \right\|^2, \tag{8}$$

as proposed by Vasilev et al. [VGM+20], since it yielded the best results in their experiments.

3.3 Dataset Comparison Approach

In this section, we explain how the methods described in Sect. 3.2 can be leveraged to compare the whole datasets by explaining the data pipeline, also visible in Fig. 2. As starting points of the pipeline, a training dataset $\mathcal{D}^{\text{tr}}$, a validation dataset $\mathcal{D}^{\text{val}}$, which is in the distribution of the training dataset, and a test dataset $\mathcal{D}^{\text{dom}}$ are needed. $\mathcal{D}^{\text{dom}}$ is the dataset, for which the method tests, whether the dataset is in-domain or out-of-domain of the respective training dataset . The presented metrics should therefore be able to separate $\mathcal{D}^{\text{val}}$ and $\mathcal{D}^{\text{dom}}$. Dataset $\mathcal{D}^{\text{val}}$ most probably will have single images with higher novelty scores and $\mathcal{D}^{\text{dom}}$ can have already known images. Therefore, the metrics most probably cannot perfectly separate $\mathcal{D}^{\text{val}}$ and $\mathcal{D}^{\text{dom}}$. This is solved by an analysis of the novelty metric results on a dataset level.

The first step of the pipeline is the training of a VAE with the $\mathcal{D}^{\text{tr}}$ dataset. For the novelty detection, the decoder can be discarded after training. As shown in Fig. 2, the trained encoder can now be used to generate the latent space representations of $\mathcal{D}^{\text{tr}}$, $\mathcal{D}^{\text{val}}$, and $\mathcal{D}^{\text{dom}}$, spanned by the vectors $\boldsymbol{\mu}^{\text{tr}}$, $\boldsymbol{\sigma}^{\text{tr}}$, $\boldsymbol{\mu}^{\text{val}}$, $\boldsymbol{\sigma}^{\text{val}}$, $\boldsymbol{\mu}^{\text{dom}}$, and $\boldsymbol{\sigma}^{\text{dom}}$.

Now, the novelty scores provided in Sect. 3.2 can be applied to every latent space representation of all data samples in $\mathcal{D}^{\text{val}}$ and $\mathcal{D}^{\text{dom}}$. Hence, for the decision, whether $\mathcal{D}^{\text{dom}}$ has a large gap to $\mathcal{D}^{\text{val}}$, the novelty scores have to be com-

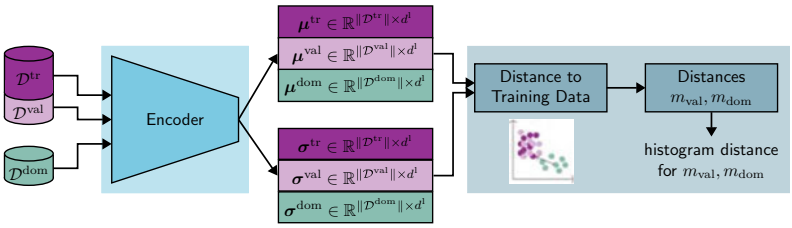


Fig. 2 Approach for dataset comparison based on latent space-based novelty scores. After training, the encoder is used to transform images into their latent space. In this space, different metrics can be applied to calculate nearest neighbor distances between different datasets. Domain shifts can then be detected by analyzing the histograms deviations

pared. In case of the nearest neighbor approaches, the resulting novelty scores $m_{\text{val}} = \{(\text{val}_1, w_{\text{val}_1}), \ldots, (\text{val}_n, w_{\text{val}_n})\}$ and $m_{\text{dom}} = \{(\text{dom}_1, w_{\text{dom}_1}), \ldots, (\text{dom}_n, w_{\text{dom}_n})\}$ are represented as a histogram where $n \in \mathbb{N}$ is the number of buckets, val_a and dom_b are bucket representatives with $0 < a, b \leq n$, with $a, b \in \mathbb{N}$, and w_{val_a}, w_{dom_b} being the weights of the bucket [RTG00]. To compare the histograms we use the earth mover's distance (EMD) [RTG00]. The earth mover's distance can calculate the difference between the histogram m_{val} and m_{dom} by calculating the earth amount f_{ab} which is to be moved from bucket val_a to dom_b [RTG00]. To calculate the distance, the EMD also needs the ground distance matrix $\mathbf{D} = (d_{ab})$, where d_{ab} is the ground distance between bucket val_a and dom_b [RTG00]. With these terms, the EMD can be defined as follows:

$$EMD(m_{\text{val}}, m_{\text{dom}}) = \frac{\sum_{a=1}^{n} \sum_{b=1}^{n} d_{ab} f_{ab}}{\sum_{a=1}^{n} \sum_{b=1}^{n} f_{ab}}. \tag{9}$$

Here, $\sum_{a=1}^{n} \sum_{b=1}^{n} d_{ab} f_{ab}$ is the flow between m_{val} and m_{dom}, that minimizes the overall cost with the following constraints [RTG00]:

$$f_{ab} \geq 0 \qquad 1 \leq a \leq n, 1 \leq b \leq n, \tag{10}$$

$$\sum_{b=1}^{n} f_{ab} \leq w_{\text{val}_a} \quad 1 \leq a \leq n, \tag{11}$$

$$\sum_{a=1}^{n} f_{ab} \leq w_{\text{dom}_b} \quad 1 \leq b \leq n, \tag{12}$$

$$\sum_{a=1}^{n} \sum_{b=1}^{n} f_{ab} = \min\left(\sum_{a=1}^{n} w_{\text{val}_a}, \sum_{b=1}^{n} w_{\text{dom}_b} \right). \tag{13}$$

The used histograms are normalized on the x-axis to a range of values between 0 and 1. Additionally, the y-axis is also normalized, such that the histogram bars sum to 1. With this normalization, comparability between different experiments is achieved, since the results are independent of absolute metric values and dataset sizes.

For the reconstruction error, the histogram approach is comparable with the distances to the nearest neighbors. The histograms then are defined over the deterministic reconstruction error instead of the latent space distance.

4 Experimental Setup

In this section, first the selection of the datasets is explained. Afterwards, the experiments are explained and the hyperparameter values are given. The chapter is concluded with the training parameters for the experiments.

4.1 Datasets

For the evaluation of our method, we selected different datasets which all have different focuses. The first dataset is the Oxford RobotCar Dataset [MPLN17]. In this dataset, the same route was driven multiple times over the time period of 1 year, except for a few small deviations. The special feature of this dataset is that different weather conditions and seasons were recorded on the same route. This allows various domain shift analyses, such as changing weather conditions, e.g., sunny weather vs. snow. In the training, validation, snow test, rain test, and night test data splits there are 9130, 3873, 4094, 9757, and 9608 images, respectively. For all experiments, the test split is created to test whether the data is in-domain or out-of-domain.

As a second dataset, the Audi autonomous driving dataset (A2D2) [GKM+20] was used. This dataset is relatively new and was acquired in different German cities. For our evaluation, we compared the images from the city of Gaimersheim with the images from the city of Munich. The assumption with this dataset is that it shows relatively little variation because it was recorded in two German cities in the state of Bavaria. Thus, it should result in a very low dataset distance. The small difference is expected because Munich is larger than Gaimersheim, and therefore it is a comparison between an urban and a suburban city. The dataset is divided into training dataset of 12550 images, validation dataset of 3138 images, and test dataset of 5490 images.

As a third dataset, the nuScenes dataset [CBL+20] was selected. This dataset was selected because it was recorded in Boston and Singapore. The interesting point in the comparison is that there is right-hand traffic in Boston and left-hand traffic in Singapore. A number of 30013 images are used for training, 40152 images for validation, and 20512 images for test.

All datasets consist of a number of unconnected video sequences. To avoid similar images in different datasets splits, the split between $\mathcal{D}^{\text{tr}}$, $\mathcal{D}^{\text{val}}$, and $\mathcal{D}^{\text{dom}}$ was done sequence-wise. This ensures, that every scene only occurs in exactly one dataset. Only for the A2D2 dataset, the split between $\mathcal{D}^{\text{tr}}$, $\mathcal{D}^{\text{val}}$ was inside the Gaimersheim sequence, resulting in a few very similar images just before and after the split. However, this should not have a big effect, since the similar images take only a very small percentage of the whole dataset.

4.2 Experiments

For the evaluation of our method, we selected three different datasets, which all have a different focus. With these datasets, five different domain shifts were generated. These domain shifts can be seen in Table 1.

The five different domain shifts were investigated with four different VAE variants. Each variant was tested with 6 different metrics, leading to 120 experiments in total.

Experiments 1–3 were performed with the Oxford dataset. For training, images were taken on sunny days and as validation data, images were selected that were

Table 1 All our datasets and experiments

Experiment	Dataset $\mathcal{D}^{\text{tr}}$	Dataset $\mathcal{D}^{\text{val}}$	Dataset $\mathcal{D}^{\text{dom}}$
Experiment 1	Oxford	Oxford alternate route	Snow
Experiment 2	Oxford	Oxford alternate route	Rain
Experiment 3	Oxford	Oxford alternate route	Night
Experiment 4	Gaimersheim	Gaimersheim (holdout)	Munich
Experiment 5	Boston	Boston (holdout)	Singapore

taken on an alternative route under the same weather conditions. This results in the following experiments for the Oxford dataset. Experiment 1 tests if the latent space can be used to distinguish sunny days from snowy days. The second experiment tests how rain can be distinguished from sun, and the third experiment investigates how sunny days differ from night scenes (Fig. 3).

Experiment 4 uses the A2D2 dataset. This experiment only has a very small domain shift, since we compare data recorded in the sub-urban area of the small German city Gaimersheim compared to the urban area of Munich. Experiment 5 uses the nuScenes dataset. This experiment used images from Boston as training data and, as in Experiment 4, used a holdout of the data as validation data. Images from Singapore were used as a test set. Since in Singapore also night scenes were recorded, it has been ensured that no night scenes are in the test dataset.

To allow comparison of the experiments and metrics, baseline experiments were conducted in addition to the domain shift experiments. This is necessary to find out which value the metric has for in-domain data. For this purpose, the validation dataset $\mathcal{D}^{\text{val}}$ was divided into $\mathcal{D}^{\text{val}_0}$ and $\mathcal{D}^{\text{val}_1}$. This makes it possible to define a baseline for each experiment by setting $\mathcal{D}^{\text{val}} = \mathcal{D}^{\text{val}_0}$ and $\mathcal{D}^{\text{dom}} = \mathcal{D}^{\text{val}_1}$.

4.3 VAE Hyperparameters

To ensure comparability between the VAE, β-VAE, and JointVAE, we used the same layer architecture and hyperparameters. The goal of this chapter is to be able to give an initial indication of whether a domain shift can be detected with simply chosen hyperparameters. It is to be expected, that with optimization of hyperparameters on multiple datasets, better results can be achieved.

To evaluate the performance of the presented methods, we have used the standard VAE, β-VAE, JointVAE, and the NVAE. The architecture was the same for the VAE, β-VAE, and the JointVAE. We use five hidden layers with dimensions 32, 64, 128, 256, and 512. After the hidden layers, we add three residual blocks. The residual blocks follow a latent space with 100 neurons. Both β and γ were set to 10.5 and the input size of the images was specified to a format of 512×256. For the NVAE, we adopted the architecture from the paper [VK21].

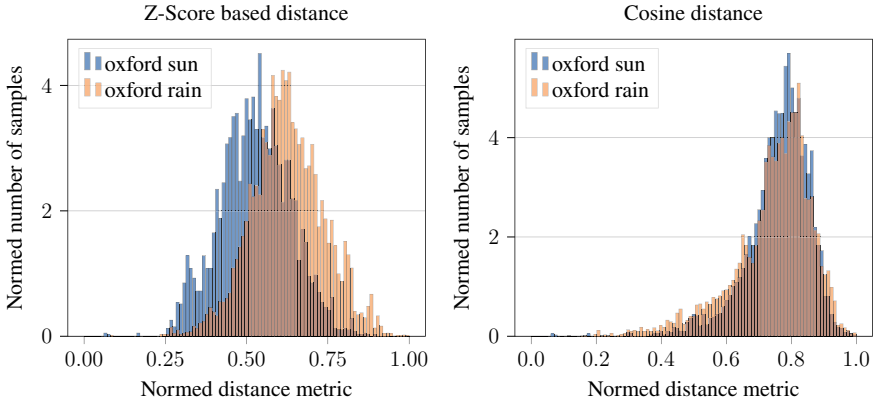


Fig. 3 These histograms show the results for Experiment 2 and the JointVAE with the z-score and cosine distance. Training was done with the sunny Oxford images (oxford sun), and it was tested if rainy images (oxford rain) are out-of-domain. You can clearly see that the histogram on the left shows a difference between rain and sun, and the histograms on the right are rather on top of each other and poorly separated

4.4 Training

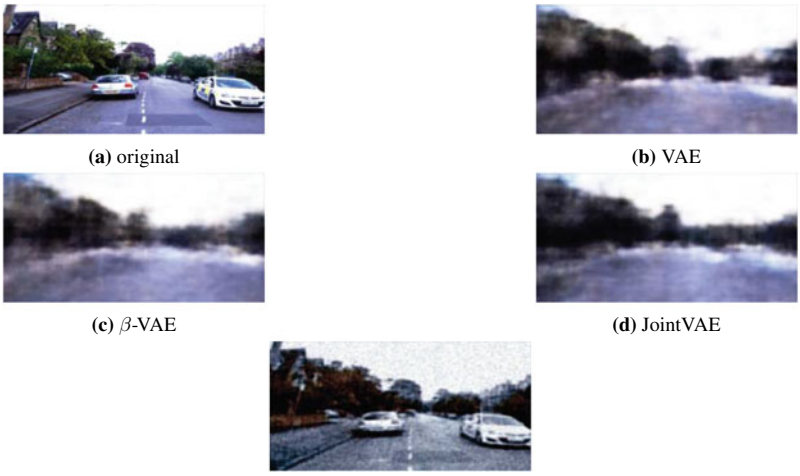
The VAE models used for the domain shift analysis were trained and implemented using `Pytorch` [PGC+17]. All VAE architectures were trained for 50 epochs with a batch size of 128 images. In addition, a learning rate of 0.0001 in combination with the Adam optimizer [KB15] was used. Figure 4 shows the original image and the reconstruction of the different VAEs. The selected image is from the validation dataset.

5 Experimental Results and Discussion

In this section, the results are presented. Section 5.1 first explains the results with the nearest neighbor method and after that in Sect. 5.2 the results will be discussed.

5.1 Results

To make the results of the different architectures and the different metrics comparable, we calculated the ratio of the in-domain data with the out-of-domain data for each experiment and architecture. This is done by



(a) original **(b)** VAE **(c)** β-VAE **(d)** JointVAE **(e)** NVAE

Fig. 4 Figure **a** shows an image from the Oxford validation dataset $\mathcal{D}^{\text{val}}$. Figures **b**–**e** show the reconstruction of the VAE architectures used in this chapter. It is clear that the NVAE has the best reconstruction quality. As expected, the reconstruction of the VAE (**b**), β-VAE (**c**), and JointVAE (TheVAEmodelsu) is worse compared to the NVAE. However, there are differences in reconstruction quality of (**b**)–(**d**), e.g., in the reconstruction of the JointVAE, compared to the VAE and β-VAE, you can imagine the car on the right edge. However, the fact that the reconstruction is worse is not a problem, because the focus of our method lies on the latent space

$$Q_{EMD} = \frac{EMD(\mathcal{D}^{\text{val}}, \mathcal{D}^{\text{dom}})}{EMD(\mathcal{D}^{\text{val}_0}, \mathcal{D}^{\text{val}_1})} \tag{14}$$

and leads to $Q_{EMD} > 1$ for a detected domain shift. The intuition behind this is that the earth mover's distance (meaning the distribution of latent space distances) between two in-domain datasets has to be very low, whereas the latent space distances to out-of-domain data should be higher. The results for the Oxford data can be seen in Tables 2, 3, and 4. It can be seen that the NVAE achieves the best results in all experiments. The reconstruction error works with the NVAE in all three experiments, even if the z-score and the NVAE work slightly better in the rain images.

The values in the A2D2 experiment are lower compared to the Oxford values. This was expected, as the domain shift is significantly smaller compared to the Oxford domain shifts. Different architectures and metrics are close to each other in this experiment. The VAE with the z-score has the best result.

In the nuScenes dataset, the ratio is higher than in the A2D2 experiment. For this experiment, different combinations are again nearly the same, but the VAE with the Manhattan distance is the best.

An important aspect of the experiment results is the stability of results, meaning a combination of architecture and metric should be able to correctly detect every kind of domain shift. For most combinations, this is not fulfilled, as can be seen

Table 2 The result of Q_{EMD} (14) from Oxford alternate route and $\mathcal{D}^{\text{dom}}$ from Oxford snow

Architecture	Bhattacharyya distance	Cosine distance	Euclidean distance	Manhattan distance	Z-score	Reconstruction error
VAE	1.18	1.49	0.52	0.47	1.01	1.32
β-VAE	7.29	4.12	1.03	0.85	1.95	1.10
JointVAE	3.77	0.50	0.37	0.45	3.00	1.25
NVAE	1.57	1.29	1.23	1.33	0.80	8.92

Table 3 The result of Q_{EMD} (14) from Oxford alternate route and $\mathcal{D}^{\text{dom}}$ from Oxford rain

Architecture	Bhattacharyya distance	Cosine distance	Euclidean distance	Manhattan distance	Z-score	Reconstruction error
VAE	1.45	0.77	1.00	1.06	1.01	1.22
β-VAE	1.88	1.24	0.76	0.89	1.72	1.14
JointVAE	1.27	0.68	0.97	1.10	2.52	1.25
NVAE	2.14	0.86	6.08	3.80	5.40	5.00

Table 4 The result of Q_{EMD} (14) from Oxford alternate route and $\mathcal{D}^{\text{dom}}$ from Oxford night

Architecture	Bhattacharyya distance	Cosine distance	Euclidean distance	Manhattan distance	Z-score	Reconstruction error
VAE	1.60	1.86	1.55	1.68	1.45	5.88
β-VAE	1.00	2.06	0.69	0.81	1.76	6.74
JointVAE	2.27	1.91	1.54	1.58	2.72	5.83
NVAE	0.00	0.43	0.85	0.60	0.60	10.42

Table 5 The result of Q_{EMD} (14) from Gaimersheim (holdout) and $\mathcal{D}^{\text{dom}}$ from Munich

Architecture	Bhattacharyya distance	Cosine distance	Euclidean distance	Manhattan distance	Z-score	Reconstruction error
VAE	0.61	0.41	0.72	0.80	1.76	0.65
β-VAE	0.54	0.28	0.40	0.42	0.68	0.85
JointVAE	0.48	0.32	0.55	0.74	1.31	0.61
NVAE	1.29	1.70	1.31	1.69	1.25	1.57

in Tables 2, 3, 4, 5, and 6. One example of this is the VAE with the Bhattacharyya distance: A combination with values over one in four of the five experiments is given (Tables 2, 3, 4, 6), while failing to detect a domain shift for the A2D2 experiment in Table 5.

Table 6 The result of Q_{EMD} (14) from Boston (holdout) and $\mathcal{D}^{\text{dom}}$ from Singapore

Architecture	Bhattacharyya distance	Cosine distance	Euclidean distance	Manhattan distance	Z-score	Reconstruction error
VAE	3.16	2.49	4.39	4.78	2.50	1.86
β-VAE	1.34	2.42	4.38	4.03	1.87	1.81
JointVAE	1.05	2.19	4.69	4.52	2.97	1.79
NVAE	0.76	2.69	1.43	4.43	1.07	1.32

5.2 Discussion

Since this chapter aims to provide insights into the applicability of latent space methods for dataset comparison, some central findings are presented here. One goal was to use various experiments to find a combination that finds all domain shifts, if possible, irrespective of the extent of the domain shift. To achieve the goal, different latent space methods were used, as there are critics who say that a nearest neighbor search does not work in high-dimensional space [AHK01]. In the work of Goos et al. [AHK01], it was said that the Euclidean distance does not work in high-dimensional spaces as well, and that the Manhattan distance achieves better results. For the NVAE, the latent space dimension of 4096 is significantly higher than for the other VAE architectures with 100 dimensions. The NVAE has this high number of latent space dimensions because we used the parameters from the NVAE paper [VK21]. The parameters were adopted to achieve the best possible reconstruction without hyperparameter optimization runs. Our results show a similar pattern. The Manhattan distance can work better than the Euclidean distance. However, both metrics have the problem that they do not always give reliable results and have multiple values less than one in the ratio calculation. In addition to investigating whether the nearest neighbor method works, we also wanted to find an architecture and metric that works stable, meaning that they are able to correctly detect all tested domain shifts. Three architectures and two metrics were found that work and are shown graphically in Fig. 5. The three architectures found have in common that they do not have a ratio with a value smaller than one in all experiments. It can be seen that the NVAE with the reconstruction error works best with strong domain shifts, resulting in large visual differences between images such as rain, snow, and night. If the domain shifts consist of different locations, resulting in a variety of only small dataset differences, the z-score with the VAE or JointVAE works better compared to the NVAE with the reconstruction error. Due to the additional categorical latent space of the JointVAE, the discrete latent space is built up better. This leads to the JointVAE functioning more stable with the z-score than the VAE. The VAE is significantly worse with the Oxford dataset and nuScenes. However, in rain and snow, the VAE works not so good, as can be seen in Fig. 5. The ratio 1.01 is so close to the in-domain/out-of-domain boundary that one can say that the VAE should rather not be used for weather domain shifts. The fact that the JointVAE works better than the

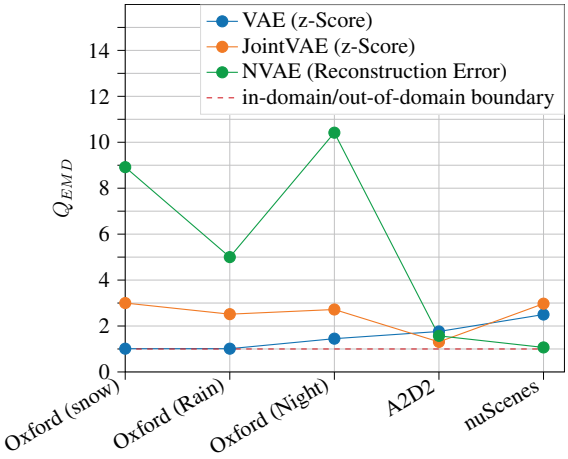


Fig. 5 In this plot, the stability of the three combinations of VAE and metrics can be seen. These three combinations have a value greater than one for each of the five experiments and therefore give stable correct results. The red dashed line indicates the in-domain/out-of-domain boundary

Table 7 The training time of the different VAE architectures with the nuScenes dataset (30013 images). Training was done on an `Nvidia A100` with 50 epochs. Additionally, we also report the time needed to transform an image into the latent space

Architecture	Training Time (hh:mm:ss)	Image to Latent Space (in seconds)
VAE	00:49:17	0.0022
β-VAE	00:50:16	0.0019
JointVAE	00:49:36	0.0022
NVAE	03:31:45	0.0101

VAE may not only be due to the latent space but also to the loss, which has more pressure on the latent spaces and is thus better structured for a nearest neighbor search. In Fig. 5, the β-VAE is missing. This is due to the A2D2 dataset, where the domain shift was not detected with each of the metrics. This could maybe be improved with a better choice of β. The optimal parametrization of each network structure was beyond the scope of this chapter. As the results show, at least in some experiments, the NVAE outperforms the other approaches. However, due to the more complex architecture, the duration of one training is four times higher than for the other architectures (see Table 7). Also, the inference time, here meaning the time to execute the encoder part, is significantly longer with factors between 4 and 5. This aspect should definitely be considered, since a practical application of such approaches probably would implicate frequent retraining and encoding of large amounts of images.

In summary, the JointVAE with the z-score works as a fast alternative for the NVAE with the reconstruction error.

6 Conclusions

This chapter investigated whether the latent space of a VAE can be used to find domain shifts in the (image) context of the automotive industry. One goal was to find a combination of metrics and VAE architectures that work in general, i.e., for different datasets and domain shifts.

The novelty of our chapter is the different types of VAE used to evaluate the shift of image data in different datasets and under different conditions.

To find out which combination is the most stable and best compared to the others, we conducted 120 experiments. In these experiments, four different VAE architectures were trained with three different datasets and five different domain shifts were evaluated. The VAE architectures differ in that three of them have the same layer structure and only the loss changes, and one architecture focuses on reconstructing the images.

Six different metrics were used to detect the domain shifts, including the reconstruction error and four different nearest neighbor searches.

The most stable results were obtained with the JointVAE and the z-score, and with the NVAE and the reconstruction error. With "stable results" we mean that some ratio of earth movers distances between the histograms of the nearest neighbor distances for in-domain and domain shift data is greater than one for all experiments. The VAE with the z-score was also above one in all experiments, but in the Oxford dataset, it was so minimally above one that this combination of metric and VAE cannot be recommended without limitations. In the experiments, the combination JointVAE and z-score performed better for more subtle domain shifts like the change of location, whereas the combination NVAE with reconstruction error performed better for visually present domain shifts like snow or rain.

The results of the experiments show that JointVAE with the latent spatial metric can be an alternative to NVAE with the reconstruction error.

In the current state, it is possible to support the function development with the selection of the best out-of-domain data. This selection can help to select useful data subsets with all relevant data present.

Better results can be expected if the function for which domain shifts are being searched is included in the training. If the information from the function, e.g., inner representations of a neural network is used to build the latent space, a performance drop of this neural network can probably be predicted better. However, this approach would be no longer function-agnostic, since it requires an initially trained function.

As a conclusion, the presented methods can be a useful tool to compare and optimize datasets with only few assumptions and can therefore be one component in a strategy to develop functions with better generalization capabilities. Yet, for a complete approach, it has to be complemented with other methods such as rule-based or function-specific dataset optimizations to mutually compensate each other's weaknesses.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)." The authors would like to thank the consortium for the successful cooperation.

References

[AC15] J. An, S. Cho, Variational autoencoder based anomaly detection using reconstruction probability. Technical report, SNU Data Mining Center (2015)

[AHK01] C. Aggarwal, A. Hinneburg, D. Keim, On the surprising behavior of distance metrics in high dimensional spaces, in *Proceedings of the International Conference on Database Theory*, pp. 420–434, London, UK (2001)

[BLO+17] J. Bach, J. Langner, S. Otten, E. Sax, M. Holzäpfel, Test scenario selection for system-level verification and validation of geolocation-dependent automotive control systems, in *Proceedings of the ICE/IEEE International Conference on Engineering, Technology and Innovation (ITMC)*, pp. 203–210, Madeira, Portugal (2017)

[BR14] S. Banerjee, A. Roy, *Linear Algebra and Matrix Analysis for Statistics* (CRC Press, Boca Raton, 2014)

[CA79] G.B. Coleman, H.C. Andrews, Image segmentation by clustering. Proc. IEEE **67**(5), 773–785 (1979)

[CBL+20] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, nuScenes: a multimodal dataset for autonomous driving, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11621–11631, Virtual Conference (2020)

[CHK21] J. Choi, G. Hwang, M. Kang, Discond-VAE: disentangling continuous factors from the discrete, pp. 1–14 (2021). arXiv:2009.08039

[Dup18] E. Dupont, Learning disentangled joint continuous and discrete representations, pp. 1–16 (2018). arXiv:1804.00104

[GKM+20] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A.S. Chung, L. Hauswald, V.H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, P. Schuberth, A2D2: Audi autonomous driving dataset, pp. 1–10 (2020). arXiv:2004.06320

[GLGL18] Y.Y. Guo, T.G. Liu, M.S. Lew, A review of semantic segmentation using deep neural networks. Int. J. Multimed. Inf. Retr. **7**(2), 87–93 (2018)

[HMP+17] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M.M. Botvinick, S. Mohamed, A. Lerchner, β-VAE: learning basic visual concepts with a constrained variational framework, in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–22, Toulon, France (2017)

[KB15] D.P. Kingma, J. Ba, ADAM: a method for stochastic optimization, in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–15, San Diego, CA, USA (2015)

[KW14] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–14, Banff, AB, Canada (2014)

[LBR+18] J. Langner, J. Bach, L. Ries, S. Otten, M. Holzäpfel, E. Sax, Estimating the uniqueness of test scenarios derived from recorded real-world-driving-data using autoencoders, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 1860–1866, Changshu, China (2018)

[LFK+20] J. Löhdefink, J. Fehrling, M. Klingner, F. Hüger, P. Schlicht, N.M. Schmidt, T. Fingscheidt, Self-supervised domain mismatch estimation for autonomous perception, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1359–1368, Virtual Conference (2020)

[LX18] Y. Lu, P. Xu, Anomaly detection for skin disease images using variational autoencoder, pp. 1–8 (2018). arXiv:1807.01349

[MPLN17] W. Maddern, G. Pascoe, C. Linegar, P. Newman, 1 year, 1000 km: the Oxford RobotCar dataset. Int. J. Robot. Res. **36**(1), 3–15 (2017)

[NB10] H.V. Nguyen, L. Bai, Cosine similarity metric learning for face verification, in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pp. 709–720, Queenstown, New Zealand (2010)

[PGC+17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops*, pp. 1–4, Long Beach, CA, USA (2017)

[RMM21] M. Rabe, S. Milz, P. Mäder, Development methodologies for safety critical machine learning applications in the automotive domain: a survey, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 129–141, Virtual Conference (2021)

[RNFP19] F. Rosique, P.J. Navarro, C. Fernández, A. Padilla, A systematic review of perception system and simulators for autonomous vehicles research. Sensors **19**(3), 1–29 (2019)

[RSBS20] L. Ries, M. Stumpf, J. Bach, E. Sax, Semantic comparison of driving sequences by adaptation of word embeddings, in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 1–7, Virtual Conference (2020)

[RTG00] Y. Rubner, C. Tomasi, L.J. Guibas, The Earth Mover's distance as a metric for image retrieval. Int. J. Comput. Vis. (IJCV) **40**, 99–121 (2000)

[SRR+20] V.K. Sundar, S. Ramakrishna, Z. Rahiminasab, A. Easwaran, A. Dubey, Out-of-distribution detection in multi-label datasets using latent space of β-VAE, in *Proceedings of the IEEE Symposium on Security and Privacy (SP) Workshops*, pp. 250–255, Virtual Conference (2020)

[SSH20] T. Sämann, P. Schlicht, F. Hüger, Strategy to increase the safety of a DNN-based perception for HAD systems, pp. 1–14 (2020). arXiv:2002.08935

[SSWS19] N.A. Stanton, P.M. Salmon, G.H. Walker, M. Stanton, Models and methods for collision analysis: a comparison study based on the Uber collision with a pedestrian. Saf. Sci. **120**(12), 117–128 (2019)

[SWXS18] J. Sun, X. Wang, N. Xiong, J. Shao, Learning sparse representation with variational auto-encoder for anomaly detection. IEEE Access **6**, 33353–33361 (2018)

[VGM+20] A. Vasilev, V. Golkov, M. Meissner, I. Lipp, E. Sgarlata, V. Tomassini, D.K. Jones, D. Cremers, Q-space novelty detection with variational autoencoders, in *Proceedings of the International Conference on Medical Image Computing & Computer-Assisted Intervention (MICCAI) Workshops*, pp. 113–124, Virtual Conference (2020)

[VK21] A. Vahdat, J. Kautz, NVAE: a deep hierarchical variational autoencoder, pp. 1–21 (2021). arXiv:2007.03898

[Ye11] J. Ye, Cosine similarity measures for intuitionistic fuzzy sets and their applications. Math. Comput. Modelling **53**(1–2), 91–97 (2011)

[ZZXW19] Z.-Q. Zhao, P. Zheng, X. Shou-Tao, W. Xindong, Object detection with deep learning: a review. IEEE Trans. Neural Netw. Learn. Syst. (TNNLS) **30**(11), 3212–3232 (2019)

Optimized Data Synthesis for DNN Training and Validation by Sensor Artifact Simulation

Korbinian Hagn and Oliver Grau

Abstract Synthetic, i.e., computer-generated imagery (CGI) data is a key component for training and validating deep-learning-based perceptive functions due to its ability to simulate rare cases, avoidance of privacy issues, and generation of pixel-accurate ground truth data. Today, physical-based rendering (PBR) engines simulate already a wealth of realistic optical effects but are mainly focused on the human perception system. Whereas the perceptive functions require realistic images modeled with sensor artifacts as close as possible toward the sensor, the training data has been recorded. This chapter proposes a way to improve the data synthesis process by application of realistic sensor artifacts. To do this, one has to overcome the domain distance between real-world imagery and the synthetic imagery. Therefore, we propose a measure which captures the generalization distance of two distinct datasets which have been trained on the same model. With this measure the data synthesis pipeline can be improved to produce realistic sensor-simulated images which are closer to the real-world domain. The proposed measure is based on the Wasserstein distance (earth mover's distance, EMD) over the performance metric mean intersection-over-union (mIoU) on a per-image basis, comparing synthetic and real datasets using deep neural networks (DNNs) for semantic segmentation. This measure is subsequently used to match the characteristic of a real-world camera for the image synthesis pipeline which considers realistic sensor noise and lens artifacts. Comparing the measure with the well-established Fréchet inception distance (FID) on real and artificial datasets demonstrates the ability to interpret the generalization distance which is inherent asymmetric and more informative than just a simple distance measure. Furthermore, we use the metric as an optimization criterion to adapt a synthetic dataset to a real dataset, decreasing the EMD distance between a synthetic and the Cityscapes dataset from 32.67 to 27.48 and increasing the mIoU of our test algorithm (`DeeplabV3+`) from 40.36 to 47.63%.

K. Hagn (✉) · O. Grau
Intel Deutschland GmbH, Lilienthalstraße 15, 85579 Neubiberg, Germany
e-mail: korbinian.hagn@intel.com

O. Grau
e-mail: oliver.grau@intel.com

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_4

1 Introduction

Validation of deep neural networks (DNNs) is increasingly resorting toward computer-generated imagery (CGI) due to its mitigation of certain issues. First, synthetic data can avoid privacy issues found with recordings of members of the public and, on the other hand, can automatically produce vast amounts of data at high quality with pixel-accurate ground truth data and reliability than costly manually labeled data. Moreover, simulations allow synthesis of rare cases and the systematic variation and explanation of critical constellations [SGH20]—a requirement for validation of products targeting safety-critical applications, such as automated driving. Here, the creation of corner cases and scenarios which otherwise could not be recorded in a real-world scenario without endangering other traffic participants is the key argument for the validation of perceptive AI with synthetic images.

Despite the advantages of CGI methods, training and validation with synthetic images still have challenges: Training with these images does not guarantee a similar performance on real-world images and validation is only valid if one can verify that the found weaknesses in the validation do not stem from the synthetic-to-real distribution shift seen in the input.

To measure and mitigate this domain shift, metrics have been introduced with various applications in the field of domain adaptation or transfer learning. In domain adaptation, the metrics such as FID, kernel inception distance (KID), and maximum mean discrepancy (MMD) are applied to train generative adversarial networks (GANs) to adapt on a target feature space [PTKY09] or to re-create the visual properties of a dataset [SGZ+16]. However, the problem of training and validation with synthetic imagery is directly related to the predictive performance of a perception algorithm on the target data, and these kinds of metrics struggle to correlate with the predictive performance [RV19]. Additionally, applications of domain adaptation methods often resort to specifically trained DNNs, e.g., GANs, which adapt one domain to the other and therefore add an extra layer of complexity and uncontrollability. This is especially unwanted if a validation goal is tested, e.g., to detect all pedestrians, and the domain adaption by a GAN would add additional objects into the scene (e.g., see [HTP+18]) making it even harder to attribute detected faults of the model to certain specifics of the tested scene. Here, the creation of images via a synthesis process allows to understand domain distance influence factors more directly as all parameters are under direct control.

Camera-recorded images inherently show visual imperfections or artifacts, such as sensor noise, blur, chromatic aberration, or image saturation, as can be seen in an image example from the A2D2 [GKM+20] dataset in Fig. 1. CGI methods, on the other hand, are usually based on idealized models; for example, the pinhole camera model [Stu14] which is free of sensor artifacts.

In this chapter, we present an approach to decrease the domain divergence of synthetic to real-world imagery for perceptive DNNs by realistically modeling sensor lens artifacts to increase the viability of CGI for training and validation. To achieve this, we first introduce a model of sensor artifacts whose parameters are extracted

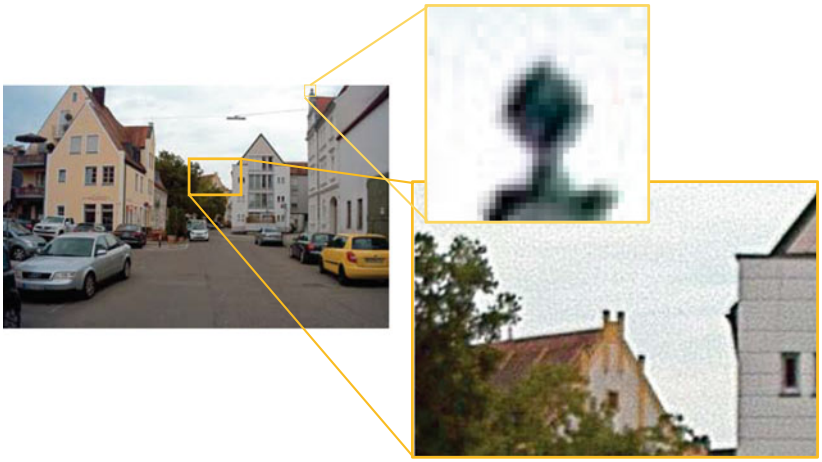


Fig. 1 Real-world images (here A2D2) exhibit sensor lens artifacts which have to be closely modeled by an image synthesization process to decrease the domain distance of synthetic to real-world datasets to make them viable for training and validation

from a real-world dataset and then apply it on a synthetic dataset for training and measuring the remaining domain divergence via validation. Therefore, a new interpretation of the domain divergence by generalization of the distance of two datasets by the per-image performance comparison over a dataset utilizing the Wasserstein or earth mover's distance (EMD) is presented. Next, we demonstrate how this model is able to decrease the domain divergence further by optimization of the initial extracted sensor camera simulation parameters as depicted in Fig. 6. Additionally, we compare our results with randomly chosen parameters as well as with randomly chosen and optimized parameters. Last, we strengthen the case for the usability of our EMD domain divergence measure by comparison with the well-known Fréchet inception distance (FID) on a set of real-world and synthetic datasets and highlight the advantage of our asymmetric domain divergence against the symmetric distance.

2 Related Works

This chapter is related to two areas: domain distance measures, as used in the field of domain adaptation and synthetic data generation for training and validation.

Domain distance measures: A key challenge in domain adaptation approaches is the expression of a distance measure between datasets, also called domain shift. A number of methods were developed to mitigate this shift (e.g., see [LCWJ15, GL15, THSD17, THS+18]).

To measure the domain shift or domain distance, the inception score (IS) has been proposed [SGZ+16], where the classification output of an `InceptionV3-`

based [SVI+16] discriminator network trained on the ImageNet dataset [DDS+09] is used. The works of [HRU+17, BSAG21] rely on features extracted from the `InceptionV3` network to tune domain adaptation approaches, i.e., the FID and KID. However, these metrics cannot predict if the classification performance increases when adapted data is applied as training data for a discriminator [RV19].

Therefore, to measure performance directly, it is essential to train with the adapted or synthetic data and validate on the target data, i.e., cross-evaluation as done by [RSM+16, WU18, SAS+18].

Performance metrics: The mean intersection-over-union (mIoU) is a widely used performance metric for benchmarking semantic segmentation [COR+16, VSN+18]. Adaptations and improvements of the mIoU have been proposed which put more weight on the segmentation contour as in [FMWR18, RTG+19]. Performance metrics as the mIoU are computed over the whole validation dataset, i.e., the whole confusion matrix, but there are propositions to apply the mIoU calculation on a per-image basis and compare the resulting empirical distributions [CLP13].

A per-image comparison mitigates several shortcomings of a single evaluation metric on the whole dataset when used for comparison of classifiers on the same dataset. First, one can distinguish multimodal and unimodal distributions, i.e., strong classification on one half and weak classification on the other half of a set can lead to the same mean as an average classification on all samples. Second, unimodal distributions with the same mean but different shape are also indiscernible under a single dataset averaged metric. This justification led to our choice of a per-image-based mIoU metric as it allows for deeper investigations which are especially helpful when one wants to understand the characteristics that increase or decrease a domain divergence.

Sensor simulation for synthetic image training: The use of synthesized data for development and validation is an accepted technique and has been also suggested for computer vision applications (e.g., [BB95]). Recently, specifically for the domain of driving scenarios, games engines have been adapted [RHK17, DRC+17].

Although game engines provide a good starting point to simulate environments, they usually only offer a closed rendering set-up with many trade-offs balancing between real-time constraints and a subjectively good visual appearance for human observers. Specifically the lighting computation in the rendering pipelines is intransparent. Therefore, it does not produce a physically correct imagery; instead only a fixed rendering quality (as a function of lighting computation and tone mapping), resulting in output of images having a low dynamic range (LDR) (typically 8-bit per RGB color channel).

Recently, physical-based rendering techniques have been applied to the generation of data for training and validation, like Synscapes [WU18]. For our chapter we use a dataset in high dynamic range (HDR) created with the physical-based `Blender Cycles` renderer.[1] We implemented a customized tone mapping to 8-bit per color channel and sensor simulation, as described in the next section.

[1] Provided by the KI Absicherung project [KI 20].

While there is great interest in understanding the domain distance in the area of domain adaptation via generative strategies, i.e., GANs, there has been little research regarding sensor artifact influence on training and validation with synthetic images. Other works [dCCN+16, NdCCP18] add different kinds of sensor noise to their training set and report a degradation of performance, compared to a model trained with no noise in the training set, due to training of a harder, i.e., noisier, visual task. Adding noise in training is a common technique for image augmentation and can be seen as a regularization technique [Bis95] to prevent overfitting.

Our task of modeling sensor artifacts for synthetic images extracted from camera images is not aimed at improving the generalization through random noise, but to tune the parameters of our sensor model to closely replicate the real-world images and improve generalization on the target data.

First results of modeling camera effects to improve synthetic data learning on the perceptive task of bounding box detection have been proposed by [CSVJR18, LLFW20]. Lin et al. [LLFW20] additionally state that generalization is an asymmetric measure which should be considered when comparing with symmetric dataset distance measures from literature. Furthermore, Carlson et al. [CSVJR19] learned sensor artifact parameters from a real-world dataset and applied the learned parameters of their noise sources as image augmentation during training with synthetic data on the task of bounding box detection. However, contrasting our approach, they apply their optimization as style loss on a latent feature vector extracted from a `VGG-16` network trained on ImageNet and evaluate the performance on the task of 2D object detection.

3 Methods

Given a synthetic (CGI) dataset of urban street scenes, our goal is to decrease the domain gap to a real-world dataset for semantic segmentation by realistic sensor artifact simulation. Therefore, we systematically analyze the image sensor artifacts of the real-world dataset and use this extracted parametrization for our sensor artifact simulation. To compare our synthetic dataset with the real-world dataset we contrive a novel per-image performance-based metric to measure the generalization distance between the datasets. We utilize a `DeeplabV3+` [CZP+18] semantic segmentation model with a `ResNet101` [HZRS16] backbone to train and evaluate on the different datasets throughout this paper. To show the valuable properties of our measure we compare it with the established domain distance, i.e., Fréchet inception distance (FID). Lastly, we use our measure as optimization criteria for adapting the parameters of our sensor artifact simulation with the extracted parameters as starting point and show that we can further decrease the domain distance from synthetic images to real-world images.

3.1 Sensor Simulation

We implemented a simple sensor model with the principle blocks depicted in Fig. 2: The module expects images in linear RGB space. Rendering engines like `Blender Cycles`[2] can provide these images as results in `OpenEXR` format.[3]

We simulate a simple model by applying *chromatic aberration*, *blur*, and *sensor noise*, as additive Gaussian noise (zero mean, variance is a free parameter), followed by a simple exposure control (linear tone mapping), finished by non-linear *gamma correction*.

First, we apply blur by a simple box filter with filter size $F \times F$ and a chromatic aberration (CA). The CA is approximated using radial distortions (k1, second order), e.g., [CV14], as defined in OpenCV. The CA is implemented as a per channel (red, green, blue) variation of the k1 radial distortion, i.e., we introduce an incremental parameter ca that affects the radial distortions: $\mathrm{k1(blue)} = -ca$; $\mathrm{k1(green)} = 0$; $\mathrm{k1(red)} = +ca$. As the next step, we apply Gaussian noise to the input image.

Applying a linear function, the pixel values are then mapped and rounded to the target output byte range [0, ..., 255].

The two parameters of the linear mapping are determined by a histogram evaluation of the input RGB values of the respective image, imitating an auto exposure of a real camera. In our experiments we have set it to saturate 2% (initially) of the brightest pixel values, as these are usually values of very high brightness, induced by sky or even the sun. Values below the minimum or above the set maximum are mapped to 0 or 255, respectively.

In the last step we apply gamma correction to achieve the final processed synthetic image:

$$\mathbf{x} = (\tilde{\mathbf{x}})^{\gamma} \tag{1}$$

The parameter γ is an approximation of the sensor non-linear mapping function. For media applications this is usually $\gamma = 2.2$ for the sRGB color space [RD14]. However, for industrial cameras, this is not yet standardized and some vendors do

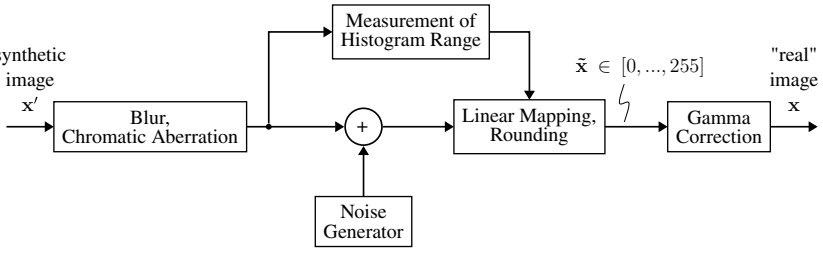


Fig. 2 Sensor artifact simulation

[2] https://www.blender.org/.

[3] https://www.openexr.com/.

Fig. 3 Left **a**: Synthetic images without lens artifacts. Right **b**: Applied sensor lens artifacts, including exposure control

not reveal it.[4] We therefore estimate the parameter as an approximation. Figure 3 depicts the difference of an image with and without simulated sensor artifacts.

3.2 Dataset Divergence Measure

Our proposed distance quantifies per image performance between models trained on different datasets but evaluated on the same dataset. Considering the task of semantic segmentation we chose the mIoU as our base metric. We then modify the mIoU to be calculated per image instead of the confusion matrix on the whole evaluated dataset. Next, we introduce the Wasserstein-1 or earth mover's distance (EMD) metric as our divergence measure between the per-image mIoU distribution of two classifiers trained on distinct datasets, i.e., synthetic and real-world datasets, but evaluated on the same real-world dataset the second classifier has been trained with.

The mIoU is defined as follows:

$$mIoU = \frac{1}{S} \sum_{s \in \mathcal{S}} \frac{TP_s}{TP_s + FP_s + FN_s} \times 100\%, \tag{2}$$

[4] The providers of the Cityscapes dataset don't document the exact mapping.

with TP_s, FN_s, and FP_s being the amount of true-positives, false-negatives, and false-positives of the sth class over all images of the evaluated dataset.

Here, $\mathcal{S} = \{0, 1, ..., S-1\}$, with $S = 11$, as we use the 11 classes defined in Table 1. These classes are the maximal overlap of common classes in the real and synthetic datasets considered for cross-evaluation and comparison of our measure with the Fréchet inception distance (FID), as can be seen later in Sect. 4.3, Tables 3 and 4.

A distribution over the per-image IoU takes the following form:

$$IoU_n = \frac{1}{S} \sum_{s \in \mathcal{S}} \frac{TP_{s,n}}{TP_{s,n} + FP_{s,n} + FN_{s,n}} \times 100\%, \tag{3}$$

where n denotes the nth image in the validation dataset. Here, IoU_n is measured in %. We want to compare the distributions of per-image IoU values from two different models; therefore, we apply the Wasserstein distance. The Wasserstein distance as an optimal mass transport metric from [KPT+17] is defined for density distributions p and q where inf denotes the infinium, i.e., lowest transportation cost, $\Gamma(p, q)$ denotes the set containing all joint distributions π, i.e., transportation maps, for (X,Y) which have the marginals p and q as follows:

$$W_r(p, q) = \left(\inf_{\pi \in \Gamma(p,q)} \int_{\mathbb{R} \times \mathbb{R}} |X - Y|^r \mathrm{d}\pi \right)^{1/r}. \tag{4}$$

This distance formulation is equivalent to the following [RTC17]:

$$W_r(p, q) = \left(\int_{-\infty}^{\infty} |P(t) - Q(t)|^r \right)^{1/r} \mathrm{d}t. \tag{5}$$

Here P and Q denote the respective cumulative distribution functions (CDFs) of p and q.

In our application we calculate the empirical distributions of p and q, which simplifies in this case to the function of the order statistics:

$$W_r(\hat{p}, \hat{q}) = \left(\sum_{i=1}^{n} |\hat{p}_i - \hat{q}_i|^r \right)^{1/r}, \tag{6}$$

where $\hat{p}$ and $\hat{q}$ are the empirical distributions of the marginals p and q sorted in ascending order. With $r = 1$ and equal weight distributions we get the earth mover's distance (EMD) which, in other words, measures the area between the respective CDFs with L_1 as ground distance.

We assume a sample size of at least 100 to be enough for the EMD calculation to be valid, as fewer samples might not guarantee a sufficient sampling of the domains. In our experiments we use sample sizes ≥ 500.

Table 1 Due to differences in label definition of real-world datasets, the class mapping for training and evaluation is decreased to 11 classes that are common in all considered datasets: A2D2 [GKM+20], Cityscapes [COR+16], Berkeley Deep Drive (BDD100K) [YCW+20], Mapillary Vistas (MV) [NOBK17], India Driving Dataset (IDD) [VSN+18], GTAV [RVRK16], our synthetic dataset [KI 20], and Synscapes [WU18]

s	0	1	2	3	4	5	6	7	8	9	10
Label	`road`	`sidewalk`	`building`	`pole`	`traffic light`	`traffic sign`	`vegetation`	`sky`	`human`	`car`	`truck`

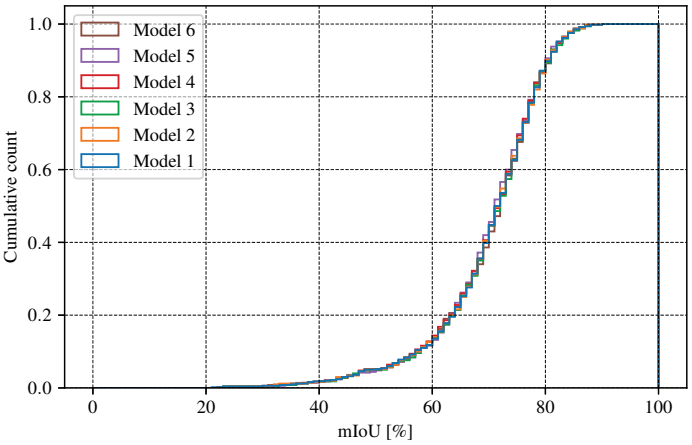


Fig. 4 CDF of ensembles of `DeeplabV3+` models trained on Cityscapes and evaluated on its validation set. Applying the 2-sample Kolmogorov-Smirnov test to each possible pair of the ensemble, we get a minimum $p-\text{value} > 0.95$

The FID is a special case of the Wasserstein-2 distance derived from (6) with $p = 2$ and $\hat{p}$ and $\hat{q}$ being normally distributed, leading to the following definition:

$$\text{FID} = ||\boldsymbol{\mu} - \boldsymbol{\mu}_w||^2 + \text{tr}(\Sigma + \Sigma_w - 2(\Sigma\Sigma_w)^{1/2}), \tag{7}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\mu}_w$ are the means, and Σ and Σ_w are the covariance matrices of the multivariate Gaussian-distributed feature vectors of synthetic and real-world datasets, respectively.

Compared to distance metrics such as the FID which by definition is symmetric, our measure is a divergence, i.e., the distance from dataset A to dataset B can be different to the distance from dataset B to dataset A. Being a divergence also reflects the characteristic of a classifier having different generalization distance when trained on dataset A and evaluated on dataset B or the other way around.

Because the ground measure of the signatures, i.e., the IoU per image, is bounded to $0 \leq IoU_n \leq 100$, the EMD measure is then bounded to $0 \leq EMD \leq 100^r$ with r being the Wasserstein norm. For $r = 1$, the measure is bound with $0 \leq EMD \leq 100$.

To verify whether the per-image IoU of a dataset is a good proxy of a dataset's domain distribution, we need to verify that the distribution stays (nearly) constant when training from different starting conditions. Therefore, we trained six models of the `DeeplabV3+` network with the same hyperparameters but different random initialization on the Cityscapes dataset and evaluated them on the validation set calculating the mIoU per image. The resulting distributions of each model in the ensemble are converted into a CDF as is shown in Fig. 4. To have a stronger empirical evidence of the per-image mIoU performance distribution being constant for a dataset, we apply the two-sample Kolmogorov-Smirnov test on each pair of distri-

bution in the ensemble. The resulting p-values are at least > 0.95, hence supporting our hypothesis.

3.3 Datasets

For our sensor parameter optimization experiments we consider two datasets. First, the real-world Cityscapes dataset, which consists of 2,975 annotated images for training and 500 annotated images for validation. All images were captured in urban street scenes in German cities. Second, the synthetic dataset provided by the KI-A project [KI 20]. This dataset consists of 21,802 annotated training images and 5,164 validation images. The KI-A synthetic dataset comprises urban street scenes, similar to Cityscapes, and suburban to rural street scenes which are characterized by less traffic and less dense house placements, therefore more vegetation and terrain objects.

4 Results and Discussion

4.1 Sensor Parameter Extraction

As a baseline for our sensor simulation, we analyzed images from the Cityscapes training data and measured the parameters. Sensor noise was extracted from about 10 images with uniformly colored areas ranging from dark to light colors. Chromatic aberration was extracted from 10 images with traffic signs on the outmost edges of the image, as can be seen in Fig. 5. The extracted values have been averaged over the count of images. The starting parameters of our optimization approach are then as follows: saturation $= 2.0\%$, noise $\sim \mathcal{N}(0,\ 3)$, $\gamma = 0.8$, $F = 4$, and $ca = 0.08$.

4.2 Sensor Artifact Optimization Experiment

Utilizing the EMD as dataset divergence measure and the extracted sensor parameters from camera images of Cityscapes, we apply an optimization strategy to iteratively decrease the gap between the Cityscapes and the synthetic dataset [KI 20]. For optimization, we chose to use the trust region reflective (trf) method [SLA+15] as implemented in `SciPy` [VGO+20]. The trf is a least-squares minimization method to find the local minimum of a cost function given certain input variables. The cost function is the EMD from synthetic model and real-world model predictions on the same real-world validation dataset. The variables as input to the cost function are the parameters of the sensor artifact simulation. The trf method has the capability

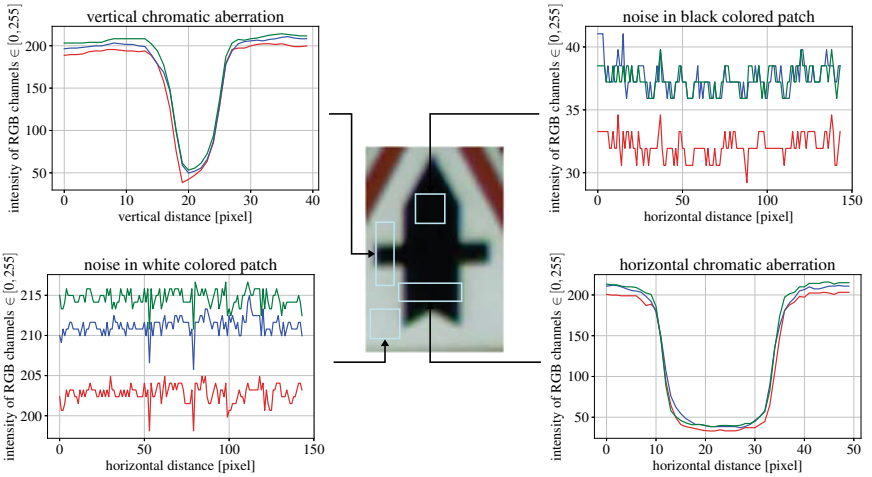


Fig. 5 Exemplary manual extraction of sensor parameters from an extracted patch of a Cityscapes image on a traffic sign in the top right corner. Diagrams clockwise beginning top left: vertical chromatic aberration, noise level on black area, horizontal chromatic aberration, noise level on a plain white area

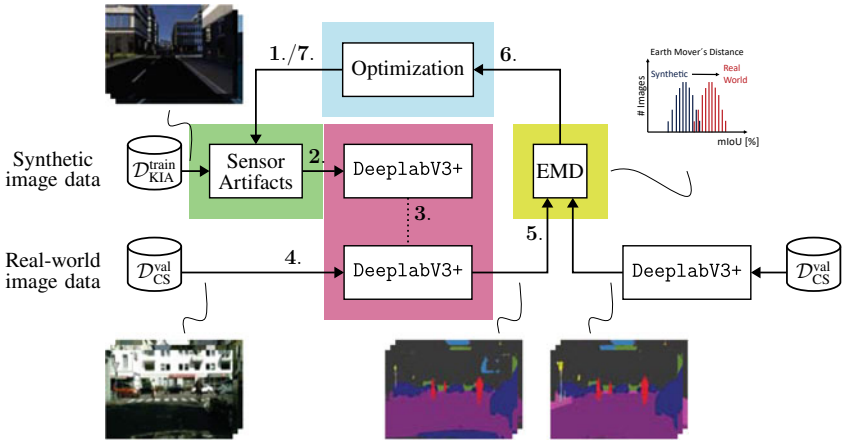


Fig. 6 Optimization of sensor artifacts to decrease divergence between real and synthetic datasets

of bounding the variables to meaningful ranges. The stop criterion is met when the increase of parameter step size or decrease of the cost function is below 10^{-6}.

The overall description of our optimization method is depicted in Fig. 6. *Step 1*: Initial parameters from the optimization method are applied in the sensor artifact simulation to the synthetic images. *Step 2*: The `DeeplabV3+` model with `ResNet101` backbone is pre-trained on 15 epochs on the original unmodified synthetic dataset and finetuned for one epoch on the synthetic dataset with applied sensor artifacts and

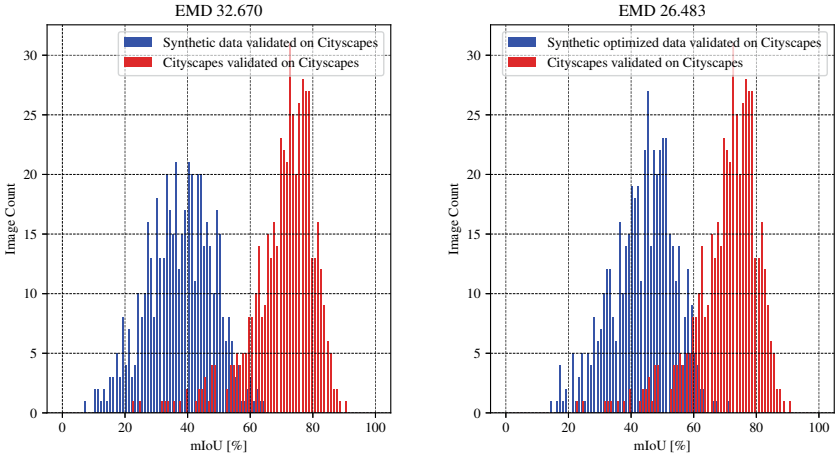


Fig. 7 EMD domain divergence calculation of synthetic and optimized synthetic data to real-world images. **a** Comparison of synthetic data with Cityscapes data, **b** synthetic sensor artifact optimized dataset compared to the target dataset Cityscapes

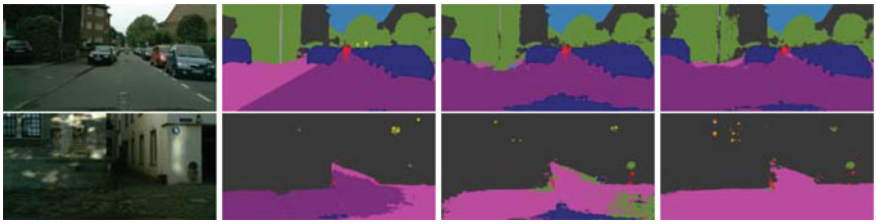
a learning rate of 0.1. *Step 3*: The model parameters are frozen and set to evaluate. *Step 4*: The model predicts on the validation set of the Cityscapes dataset. *Step 5*: The remaining domain divergence is measured by evaluation of the mIoU per image and calculation of the EMD to the evaluations of a model trained on Cityscapes. *Step 6*: The resulting EMD is fed as cost to the optimization method. *Step 7*: New parameters are set for the sensor artifact simulation, or the optimization ends if the stop criteria are met.

After iterating the parameter optimization with the trf method, we compare our optimized trained model with the unmodified synthetic dataset by their per-image mIoU distributions on the Cityscapes dataset. Figure 7 depicts the distributions resulting from this evaluation. The `DeeplabV3+` model trained with the optimized sensor artifact simulation applied on the synthetic dataset outperforms the baseline and achieves an EMD score of 26.48, while decreasing the domain gap by 6.19. The resulting parameters are saturation $= 2.11\%$, noise $\sim \mathcal{N}(0,\ 3.0000005)$, $\gamma = 0.800001$, $F = 4$ and $ca = 0.008000005$. The parameters changed only slightly from the starting point, indicating the extracted parameters as good first choice.

An exemplary visual inspection of the results in Fig. 8 helps to understand the distribution shift and therefore the decreased EMD. While the best prediction performance image (top row) increased only slightly from the synthetic trained model (c) to the sensor artifact optimized model (d), the worst prediction case (bottom row) shows improved segmentation performance for the sensor-artifact-optimized model (d), in this case even better than the Cityscapes trained model (b).

Table 2 Performance results as per-class mIoU, overall mIoU, and EMD domain divergence evaluated on Cityscapes with models trained on Cityscapes, our synthetic only, synthetic with random parameterized lens artifacts, and synthetic extracted parameterized lens artifacts datasets. For the latter two, there are models evaluated on Cityscapes with and without optimization of the parameters for the sensor lens artifact simulation. The model trained with optimized extracted parameters achieves the highest performance on the Cityscapes dataset

Model trained on	Optimized	road	sidewalk	building	pole	traffic light	traffic sign	vegetation	sky	human	car	truck	mIoU ←	EMD →
Cityscapes	no	97.50	81.21	92.22	54.54	57.28	70.26	92.33	93.98	82.55	93.67	81.62	81.56	–
w/o artifacts	no	60.46	23.51	71.99	10.70	26.00	13.47	75.72	74.70	51.27	34.46	1.74	40.37	32.67
w/ random artifacts	no	77.86	20.56	60.66	8.21	17.83	7.36	72.18	68.21	50.53	74.01	4.68	41.58	30.03
w/ extracted artifacts	no	80.65	27.17	66.54	10.43	21.64	11.87	68.82	67.99	59.22	70.06	**7.39**	44.71	29.84
w/ random artifacts	yes	**82.89**	**34.69**	71.11	11.29	16.66	9.12	69.81	**76.77**	**61.48**	**79.50**	5.30	45.76	26.58
w/ extracted artifacts	yes	79.62	30.30	**75.74**	**16.30**	**28.31**	**13.86**	**78.75**	70.88	59.74	64.06	6.42	**47.63**	**26.48**



(a) Original **(b)** CS Pred **(c)** Synth Pred **(d)** Synth Opt Pred

Fig. 8 Top row: Best performance predictions. Bottom row: Worst performance predictions. **a** Original, **b** Cityscapes-trained model prediction, **c** synthetically trained model prediction, and **d** sensor artifact optimized trained model prediction. While top performance increased only slightly, the optimization lead to more robust predictions in worst case, i.e., harder examples

We compare the overall mIoU performance on the Cityscapes datasets between models trained with the initial unmodified synthetic dataset, the synthetic dataset with random initialized lens artifact parameters, and the synthetic dataset with extracted parameters from Cityscapes with the baseline of a model trained on the Cityscapes dataset. Results are listed in Table 2 (rows 1–4). Additionally, for the random and the extracted parameters, we evaluate the performance with initial and optimized parameters, where the parameters have been optimized by our EMD minimization (rows 5 and 6). While the model without any sensor simulation achieves the lowest overall performance (row 2), the model with random parameter initialization achieves a slightly higher performance (row 3) and is surpassed by the model with the Cityscapes extracted parameters (row 4). Next, we take the models trained with optimized parameters into account (rows 5 and 6). Both models outperform all non-optimized experiment settings in terms of overall mIoU, with the model using optimized extracted parameters from Cityscapes showing the best overall mIoU (row 6). Concretely, the model trained with optimized random starting parameters achieves higher performance on classes road, sidewalk, human, and even significantly on the car class but still falls behind on five of the remaining classes and the overall performance on the Cityscapes dataset (row 5). Further, the random parameter optimized model took over 22 iterations to converge to its local minimum, whereas the optimization of extracted starting parameters only took six iterations until reaching a local minimum, making it more than three times faster to converge. Furthermore, it is shown that all models with applied sensor lens artifacts outperform the model trained without additional lens artifacts.

4.3 EMD Cross-evaluation

To get a deeper understanding of the implications of our EMD score, we evaluate our EMD results on a range of real-world and synthetic datasets for semantic segmentation. Including real-world datasets A2D2 [GKM+20], Cityscapes (CS) [COR+16],

Table 3 Cross-domain divergence results of models trained on different real-world and synthetic datasets and evaluated on various validation or test sets of an average size of 1000 images. The domain divergence is measured with our proposed EMD measure; boldface values indicate the lowest divergence comparing our synthetic (Synth) and synthetic-optimized (SynthOpt) datasets, whereas underlined values indicate the lowest divergence values over all the datasets. The model trained with optimized lens artifacts applied to the synthetic images exhibits a smaller domain divergence than the model trained without lens artifacts

EMD ↓	A2D2	BDD100K	CS	GTAV	IDD	MV	SYNS	Synth	SynthOpt
A2D2	–	18.70	23.46	37.84	20.03	10.72	46.78	34.95	**29.32**
BDD100K	6.36	–	9.45	22.14	7.26	1.42	36.33	26.43	**21.54**
CS	10.90	12.09	–	36.42	13.01	4.08	20.62	32.66	**26.48**
GTAV	33.28	28.37	29.72	–	30.55	23.30	37.53	36.08	**32.94**
IDD	24.37	19.83	24.71	34.71	–	12.81	46.23	41.64	**36.95**
MV	10.63	10.36	14.34	28.35	9.20	–	35.97	30.35	**27.03**
SYNS	25.45	31.46	23.64	45.16	25.12	23.45	–	43.76	**43.56**

Berkeley Deep Drive (BDD100K) [YCW+20], Mapillary Vistas (MV) [NOBK17], India Driving Dataset (IDD) [VSN+18], as well as synthetic GTAV [RVRK16], our synthetic (Synth and SynthOpt) [KI 20], and Synscapes (SYNS) [WU18] datasets. In Table 3 the results of cross-domain analysis measured with the EMD score are depicted. The columns denote that a `DeeplabV3+` model has been trained on the corresponding dataset, i.e., the source dataset, whereas the rows denote the datasets it was evaluated on, i.e., the target datasets. Our optimized synthetic dataset achieves lower EMD scores, shown in boldface, than the synthetic baseline. While the domain divergence decrease is high on real datasets, the divergence decreased only marginally for the other synthetic datasets. Inspecting the EMD result on all datasets, the lowest divergence values are indicated by underline; the MV dataset shows to be closest to all the other evaluated datasets.

To set our measure in relation to established domain distance measures, we calculated the FID from each of our considered datasets to one another. The results are shown in Table 4. The FID, defined in (7), is the Wasserstein-2 distance of feature vectors from the `InceptionV3` [SVI+16] network sampled on the two datasets to be compared with each other.

Again, boldface values indicate the lowest FID values between the synthetic (Synth) and synthetic-optimized (SynthOpt) datasets, whereas underlined values indicate the lowest values of all datasets. Here, only 4 out of the 7 datasets are closer, measured by the FID, to the synthetic-optimized dataset than to the original dataset. Furthermore, the FID sees the CS and the SYNS dataset closer to one another than the EMD divergence measure, while the MV dataset shows the lowest FID among the other evaluated datasets.

FID and EMD somewhat agree, if we evaluate the distance as minimum per-row in both tables, that the Mapillary Vistas dataset is in most cases the dataset that is closest to all other datasets.

Table 4 Cross-domain distance results measured with the Fréchet inception distance (FID). Lowest FID between synthetic (Synth) and synthetic optimized (SynthOpt) datasets are in boldface, whereas the lowest FID values over all datasets are underlined

FID ↓	A2D2	BDD100K	CS	GTAV	IDD	MV	SYNS	Synth	SynthOpt
A2D2	–	60.16	98.46	78.16	58.75	41.84	109.35	**116.54**	121.55
BDD100K	60.16	–	59.90	62.42	52.15	29.66	74.871	115.51	**109.08**
CS	98.46	59.90	–	85.81	68.92	59.69	43.87	119.97	**112.42**
GTAV	78.16	62.42	85.81	–	74.08	51.00	89.62	92.513	**92.24**
IDD	58.75	52.15	68.92	74.08	–	37.36	64.09	**118.06**	125.30
MV	41.84	29.66	59.69	51.00	37.36	–	70.24	**74.46**	78.66
SYNS	109.35	74.87	43.87	89.62	64.09	70.24	–	113.77	**108.3**

Now, calculating the minimum per-column in both tables, the benefit of our asymmetric EMD comes to the light. The minimum per-column values of the FID are unchanged due to the diagonal symmetry of the cross-evaluation matrix stemming from the inherent symmetry of the measure. However, the EMD regards the BDD100K as the closest dataset. An intuitive explanation for the different minimum observations of the EMD is as follows: Training with many images exhibiting different geospatial and sensor properties of the Mapillary Vistas dataset covers a very broad domain and results in good generalization capability and therefore evaluation performance. Training with any of the other datasets cannot generalize well to the vast domain of Mapillary Vistas but to the rather constrained domain of BDD100K, which consists of lower resolution images with heavy compression artifacts, where even a model that has been trained on BDD100K does not generalize well on.

The asymmetric nature of our EMD allows for a more thorough and complex analysis of dataset discrepancies, when applied to the tasks of visual understanding, e.g., semantic segmentation, which otherwise cannot be captured by inherently symmetric distance metrics such as FID. Contrasting to [LLFW20], we could with our evaluation method not identify a consistency between FID and the generalization divergence, i.e., our EMD measure.

5 Conclusions

In this chapter, we could demonstrate that by utilizing the performance metric per image as a proxy distribution for a dataset and the earth mover's distance (EMD) as a divergence measure between distributions, one can decrease visual differences of a synthetic dataset through optimization and increase the viability of CGI for training and validation purposes of perceptive AI. To reinforce our argument for per-image performance measures as proxy distributions, we showed that training an ensemble of a fixed model with different random starting conditions but with the same

hyperparameters leads to the same per-image performance distributions when these ensemble models are evaluated on the validation set of the training dataset. When utilizing synthetic imagery for validation, the domain gap, due to visual differences between real and computer-generated images, is hindering the applicability of these datasets. As a step toward decreasing the visual differences, we apply the proposed divergence measure as a cost function to an optimization which varies the parameters of the sensor artifact simulation, while trying to re-create the sensor artifacts that the real-world dataset exhibits. As starting point of the sensor artifact parameters, we extracted empirically the values from chosen images of the real-world dataset. The optimization improved the visual difference between the real-world and the optimized synthetic dataset measurably by the EMD and we could show that even when starting with random initialized parameters we can decrease the EMD and increase the mIoU on the target datasets. When measuring the divergence after parameter optimization to other real-world and synthetic datasets, we could show that the EMD decreases for all considered datasets but when measured by the FID only four of the datasets are closer. As the EMD is derived from the mIoU per image, it is an indicator of performance on the target dataset, whereas the FID fails to relate with performance. Effective minimization of the visual difference between synthetic and real-world datasets with the EMD domain divergence measure is one step further toward fully utilizing CGI for validation of perceptive AI functions.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)". The authors would like to thank the consortium for the successful cooperation.

References

[BB95] W. Burger, M.J. Barth, Virtual reality for enhanced computer vision, in J. Rix, S. Haas, J. Teixeira (eds.), *Virtual Prototyping: Virtual Environments and the Product Design Process* (Springer, 1995), pp. 247–257

[Bis95] M. Christopher Bishop, Training with noise is equivalent to Tikhonov regularization. Neural Comput. **7**(1), 108–116 (1995)

[BSAG21] M. Binkowski, D.J. Sutherland, M. Arbel, A. Gretton, *Demystifying MMD GANs*, Jan. 2021, pp. 1–36. arxiv:1801.01401

[CLP13] G. Csurka, D. Larlus, F. Perronnin, What is a good evaluation measure for semantic segmentation? in *Proceedings of the British Machine Vision Conference (BMVC)*, Bristol, UK, Sept. 2013, pp. 1–11

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The Cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 3213–3223

[CSVJR18] A. Carlson, K.A. Skinner, R. Vasudevan, M. Johnson-Roberson, Modeling camera effects to improve visual learning from synthetic data, in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, Munich, Germany, Aug. 2018, pp. 505–520

[CSVJR19] A. Carlson, K.A. Skinner, R. Vasudevan, M. Johnson-Roberson, *Sensor Transfer: Learning Optimal Sensor Effect Image Augmentation for Sim-to-Real Domain Adaptation*, Jan. 2019, pp. 1–8. arxiv:1809.06256

[CV14] V. Chari, A. Veeraraghavan, L. Distortion, Radial distortion, in *Computer Vision: A Reference Guide*. ed. by K. Ikeuchi (Springer, Boston, MA, 2014), pp. 443–445

[CZP+18] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-Decoder with atrous separable convolution for semantic image segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, Sept. 2018, pp. 833–851

[dCCN+16] G.B.P. da Costa, W.A. Contato, T.S. Nazaré, J.E.S. do Batista Neto, M. Ponti, *An Empirical Study on the Effects of Different Types of Noise in Image Classification Tasks*, Sept. 2016, pp. 1–6. arxiv:1609.02781

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, F.-F. Li, ImageNet: a large-scale hierarchical image database, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, USA, June 2009, pp. 248–255

[DRC+17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: an open urban driving simulator, in *Proceedings of the Conference on Robot Learning CORL*, Mountain View, CA, USA, Nov. 2017, pp. 1–16

[FMWR18] E. Fernandez-Moral, R. Martins, D. Wolf, P. Rives, A new metric for evaluating semantic segmentation: leveraging global and contour accuracy, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, June 2018, pp. 1051–1056

[GKM+20] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A.S. Chung, L. Hauswald, V.H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, P. Schuberth, *A2D2: Audi Autonomous Driving Dataset*, April 2020, pp. 1–10. arxiv:2004.06320

[GL15] Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in *Proceedings of the International Conference on Machine Learning (ICML)*, Lille, France, July 2015, pp. 1180–1189

[HRU+17] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, GANs trained by a two time-scale update rule converge to a local Nash equilibrium, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 6626–6637

[HTP+18] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, T. Darrell, CyCADA: cycle-consistent adversarial domain adaptation, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden, July 2018, pp. 1989–1998

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 770–778

[KI 20] KI Absicherung Consortium. KI Absicherung: Safe AI for Automated Driving (2020). Assessed 18 Nov. 2021

[KPT+17] S. Kolouri, S.R. Park, M. Thorpe, D. Slepcev, G.K. Rohde, Optimal mass transport: signal processing and machine-learning applications. IEEE Signal Process. Mag. **34**(4), 43–59 (2017)

[LCWJ15] M. Long, Y. Cao, J. Wang, M.I. Jordan, Learning transferable features with deep adaptation networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, July 2015, pp. 97–105

[LLFW20] Z. Liu, T. Lian, J. Farrell, B. Wandell, Neural network generalization: the impact of camera parameters. IEEE Access **8**, 10443–10454 (2020)

[NdCCP18] T.S. Nazaré, G.B.P. da Costa, W.A. Contato, M. Ponti, Deep convolutional neural networks and noisy images, in *Proceedings of the Iberoamerican Congress on Pattern Recognition (CIARP)*, Madrid, Spain, Nov. 2018, pp. 416–424

[NOBK17] G. Neuhold, T. Ollmann, S. Rota Bulò, P. Kontschieder, The mapillary vistas dataset for semantic understanding of street scenes, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017, pp. 4990–4999

[PTKY09] S.J. Pan, I.W. Tsang, J.T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, in *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, Pasadena, CA, USA, July 2009, pp. 1187–1192

[RD14] R. Ramanath, M.S. Drew, Color Spaces, in *Computer Vision: A Reference Guide*. ed. by K. Ikeuchi (Springer, Boston, MA, 2014), pp. 123–132

[RHK17] S.R. Richter, Z. Hayder, V. Koltun, Playing for benchmarks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017, pp. 2232–2241

[RSM+16] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, A.M. Lopez, The SYNTHIA dataset: a large collection of synthetic images for semantic segmentation of urban scenes, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 3234–3243

[RTC17] A. Ramdas, N. García Trillos, M. Cuturi, On Wasserstein two sample testing and related families of nonparametric tests. Entropy **19**(2), 47 (2017)

[RTG+19] H. Rezatofighi, N. Tsoi, J.Y. Gwak, A. Sadeghian, I. Reid, S. Savarese, Generalized intersection over union: a metric and a loss for bounding box regression, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, pp. 658–666

[RV19] S.V. Ravuri, O. Vinyals, Seeing is not necessarily believing: limitations of BigGANs for data augmentation, in: *Proceedings of the International Conference on Learning Representations (ICLR) Workshops*, New Orleans, LA, USA, June 2019, pp. 1–5

[RVRK16] S.R. Richter, V. Vineet, S. Roth, V. Koltun, Playing for data: ground truth from computer games, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Amsterdam, The Netherlands, Oct. 2016, pp. 102–118

[SAS+18] F.S. Saleh, M.S. Aliakbarian, M. Salzmann, L. Petersson, J.M. Alvarez, Effective use of synthetic data for urban scene semantic segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, Sept. 2018, pp. 84–100

[SGH20] Q.S. Sha, O. Grau, K. Hagn, DNN analysis through synthetic data variation, in *Proceedings of the ACM Computer Science in Cars Symposium (CSCS)*, virtual conference, Dec. 2020, pp. 1–10

[SGZ+16] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, Xi Chen, *Improved Techniques for Training GANs*, June 2016, pp. 1–10. arxiv:1606.03498

[SLA+15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in *Proceedings of the International Conference on Machine Learning (ICML)*, Lille, France, July 2015, pp. 1889–1897

[Stu14] P. Sturm, Pinhole Camera Model, in *Computer Vision: A Reference Guide*. ed. by K. Ikeuchi (Springer, Boston, MA, 2014), pp. 610–613

[SVI+16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 2818–2826

[THS+18] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, M. Chandraker, Learning to adapt structured output space for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 7472–7481

[THSD17] E. Tzeng, J. Hoffman, K. Saenko, T. Darrell, Adversarial discriminative domain adaptation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, July 2017, pp. 2962–2971

[VGO+20] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson,

K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, l. Polat, Y. Feng, E.W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributers. Fundamental algorithms for scientific computing in python. SciPy 1.0. Nat. Methods **17**, 261–272 (2020)

[VSN+18] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, C.V. Jawahar, *IDD: A Dataset for Exploring Problems of Autonomous Navigation in Unconstrained Environments*, Nov. 2018, pp. 1–9. arxiv:1811.10200

[WU18] M. Wrenninge, J. Unger, *Synscapes: A Photorealistic Synthetic Dataset for Street Scene Parsing*, Oct. 2018, pp. 1–13. arxiv:1810.08705

[YCW+20] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, T. Darrell, BDD100K: a diverse driving dataset for heterogeneous multitask learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference, June 2020, pp. 2636–2645

Improved DNN Robustness by Multi-task Training with an Auxiliary Self-Supervised Task

Marvin Klingner and Tim Fingscheidt

Abstract While deep neural networks for environment perception tasks in autonomous driving systems often achieve impressive performance on clean and well-prepared images, their robustness under real conditions, i.e., on images being perturbed with noise patterns or adversarial attacks, is often subject to a significantly decreased performance. In this chapter, we address this problem for the task of semantic segmentation by proposing multi-task training with the additional task of depth estimation with the goal to improve the DNN robustness. This method has a very wide potential applicability as the additional depth estimation task can be trained in a self-supervised fashion, relying only on unlabeled image sequences during training. The final trained segmentation DNN is, however, still applicable on a single-image basis during inference without additional computational overhead compared to the single-task model. Additionally, our evaluation introduces a measure which allows for a meaningful comparison between different noise and attack types. We show the effectiveness of our approach on the Cityscapes and KITTI datasets, where our method improves the DNN performance w.r.t. the single-task baseline in terms of robustness against multiple noise and adversarial attack types, which is supplemented by an improved absolute prediction performance of the resulting DNN.

1 Introduction

Motivation: For a safe operation of highly automated driving systems, a reliable perception of the environment is crucial. Various perception tasks such as semantic segmentation [LSD15], [CPK+18], depth estimation [EPF14], [ZBSL17], or optical flow estimation [LLKX19] are often implemented by deep neural networks (DNNs). The output of these DNNs is then used to build a model of the environment, which

M. Klingner (✉) · T. Fingscheidt
Institute for Communications Technology (IfN), Technische Universität Braunschweig, Schleinitzstr. 22, 38106 Braunschweig, Germany
e-mail: m.klingner@tu-bs.de

T. Fingscheidt
e-mail: t.fingscheidt@tu-bs.de

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_5

is subsequently used for decision making in high-level planning systems. As many decisions are thereby executed upon the DNN predictions, these predictions need to be reliable under all kinds of environmental changes. This is, however, in contrast to typical DNN training, which is carried out on annotated datasets [COR+16], [NOBK17] covering only a small portion of possible environment variations as more diverse datasets are often not available. During deployment, the DNN performance can therefore drop significantly due to domain shifts not covered in the training dataset. These include, for example, noises induced by different optical sensors, brightness and weather changes, deployment in a different country, or even directed adversarial attacks [GSS15]. Therefore, for a safe deployment of DNNs in autonomous driving vehicles, their performance also needs to be *robust* w.r.t. these environment changes. In this chapter we aim at an improved robustness for the task of semantic segmentation.

Robustness of DNNs: Improving the robustness of DNNs is a highly relevant research topic for applications such as autonomous driving, where the perception system has to deal with varying environmental conditions and potentially even with adversarial attacks. As adversarial attacks usually rely on the computation of gradients on the input [GSS15], [MMS+18], it has been proposed to apply a non-differentiable pre-processing [GRCvdM18], [LLL+19] such that the impact of the perturbation on the DNN performance is reduced. However, the gradients of these pre-processings can be approximated [ACW18] so that these strategies usually only make an attack harder to calculate. Our approach therefore focuses on a more robust training of the DNN where often adversarial examples or image augmentations [GSS15], [MMS+18] are utilized already during training such that the DNN will be more robust to those during inference. While these approaches often induce a decreased performance on clean images, we aim at developing a method improving performance as well as robustness.

Multi-task learning: The training of several tasks in a single multi-task DNN, i.e., multi-task learning, is known to improve the absolute prediction performance of the single tasks as well as the efficiency of their computation due to the shared network parts [EF15], [KTMFs20]. In this chapter, as shown in Fig. 1, we use a multi-task

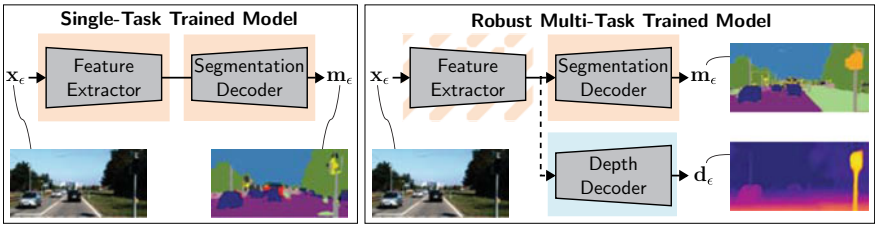


Fig. 1 Robustness improvement through multi-task learning during training. When the input $\mathbf{x}_\epsilon$ is subject to perturbations of strength ϵ, the output (segmentation $\mathbf{m}_\epsilon$, depth $\mathbf{d}_\epsilon$) of a multi-task trained model (right-hand side) is still well predicted, while the output quality of the single-task trained model (left-hand side) is strongly impaired

network for semantic segmentation and depth estimation, for which these properties have also been shown in several works [KGC18], [KTMFs20]. Particularly, we follow [KBFs20] in focusing on the additional robustness-increasing properties of such a multi-task training. Moreover, it is important to note that the depth estimation can be trained in a self-supervised fashion, which only requires short unlabeled video sequences and thereby usually does not impose large additional requirements in terms of data. While it was still necessary to manually find a good weighting between the single tasks in [KBFs20], we apply the GradNorm task weighting strategy [CBLR18], which automatically sets and adapts the task weighting according to the task-specific training progress. Also, we show the method's applicability across a wider range of datasets.

Comparability of perturbations: Up to now, a wide variety of different possible adversarial attack and noise types has been proposed [GW08], [GSS15], [CW17], [MMS+18]. However, each of these image perturbations is characterized by its own set of parameters, e.g., the standard deviation of the Gaussian noise distribution or the maximum noise value for many adversarial attacks. This makes it hard to compare perturbation strengths in a single mutual evaluation. To better compare these effects and to be able to draw conclusions between different noise and attack types, we employ a measure based on the signal-to-noise ratio, which enables a fair comparison between different perturbation types in terms of strength.

Contributions: To sum up, our contribution with this chapter is threefold. First, we propose a multi-task learning strategy with the task of self-supervised monocular depth estimation to improve a DNN's robustness. Second, we provide a detailed analysis of the positive effects of this method on DNN performance and robustness to multiple input perturbations. Third, we employ a general measure for perturbation strength, thereby making different noise and attack perturbation types comparable.

2 Related Works

In this section, we give an overview of related multi-task learning approaches for depth estimation and semantic segmentation. Afterward, we discuss methods to improve the robustness of DNNs focusing on approaches for semantic segmentation.

Multi-task learning The performance of basic perception tasks such as semantic segmentation [LSD15] and depth estimation [EF15] has increased significantly by employing fully convolutional neural networks. Furthermore, these tasks can be learned jointly in a multi-task learning setup by employing a shared encoder, which was shown to be of mutual benefit for both tasks through the more extensively learned scene understanding [EF15], [KGC18]. This could be further facilitated by combining the loss functions to enforce cross-task consistency during optimization [KGC18], [XOWS18], [ZCX+19]. For depth estimation, the research

focus shifted from supervised to self-supervised training techniques [ZBSL17], [GMAB17], [GMAFB19], due to the more general applicability on unlabeled videos, which are more readily available than labeled datasets. Accordingly, such techniques have also been employed in multi-task setups with semantic segmentation [CLLW19], [GHL+20], [NVA19]. Usually, the focus of these works is to improve the absolute prediction performance of the single involved tasks. Thereby, it was proposed to exclude the influence of dynamic objects such as cars or pedestrians [KTMFs20], [RKKY+21b], [RKKY+21a] to employ pixel-adaptive convolution for improved semantic guidance [GHL+20], [RKKY+21b], or to enforce cross-task (edge) consistency between both tasks' outputs [CLLW19], [ZBL20], [YZS+18], [MLR+19]. The approaches have also been extended from simple pinhole camera models to more general fisheye camera models [RKKY+21b], [RKKY+21a]. For semantic segmentation, the depth estimation can also be beneficial in unsupervised domain adaptation approaches [KTMFs20], [LRLG19], [VJB+19].

In this chapter, we build upon these advances by employing a multi-task learning setup of self-supervised depth estimation and semantic segmentation as introduced by Klingner et al. [KTMFs20]. However, in contrast to [KTMFs20], we put the focus rather on the robustness instead of the absolute prediction performance of the resulting semantic segmentation DNN.

Robustness of (semantic segmentation) DNNs While DNNs can achieve an impressive performance on clean images, their performance is usually not robust w.r.t. additive noise [Bis95], [HK92]. For safety-critical application this is a particularly high risk, if this noise is calculated in a way that it is nearly not recognizable if added to the image, but still heavily impairs the performance, as shown by the works on adversarial examples by Szegedy et al. [SZS+14]. Consequently, subsequent works developed various kinds of targeted and non-targeted perturbations, calculated in an image-specific fashion and optimized to fool the DNN. These adversarial examples range from simple non-iterative methods such as the fast gradient sign method (FGSM) [GSS15] to more complex iteratively calculated methods such as the momentum iterative fast gradient sign method (MI-FGSM) [DLP+18], the Carlini and Wagner attack (CW) [CW17], or the projected gradient descent (PGD) method [MMS+18]. While these image-specific attacks may not be a relevant danger in real applications due to the high computational effort per image, the existence of image-agnostic adversarial perturbations (UAPs) has also been shown, e.g., by prior-driven uncertainty estimation (PD-UA) [LJL+19], universal adversarial perturbation (UAP) [MDFFF17], or fast feature fool (FFF) [MGB17]. Although most of these works focus on the rather simple task of image classification, the applicability of these attacks to semantic segmentation is well known [AMT18], [MGR19], [BLK+21]. Furthermore, the attacks can be designed in a fashion such that the semantic segmentation outputs are completely wrong but still appear realistic [ASG+19], [MCKBF17].

The vulnerability of DNNs to different kinds of adversarial attacks has encouraged research in defense methods, which improve the robustness against these perturbations. They can be roughly divided into three categories. First, for *gradient masking*, the idea is to prevent a potential attacker from calculating the gradients

[CBG+17], [MMS+18], [PMG+17]. However, Athalye et al. [ACW18] showed that the perturbations can be calculated from a different model. Also, they can be calculated in an image-agnostic (universal) fashion [MCKBF17], [MDFFF17], see also the Chapter "Improving Transferability of Generated Universal Adversarial Perturbations for Image Classification and Segmentation" [HAMFs22], such that gradient masking methods cannot prevent the network's vulnerability to adversarial attacks under all conditions. Second, through *input transformations*, the perturbations can be removed from the input image [GRCvdM18], [KBV+21], [LLL+19], e.g., by JPEG image compression [ATT18] or by incorporation of randomness [RSFM19], [XWZ+18]. While this improves the performance on attacked images, these transformations usually impair the performance on clean images, introducing an unfavorable trade-off. Third, *redundancy* among two or three networks serving the same task can be exploited, if networks are enforced to reveal some independence [BHSFs19], [BKV+20]. Fourth, *robust training* methods can be employed such that the DNN is less sensitive to fail because of attacks from the start. Common approaches include, e.g., adversarial training [GSS15], [MMS+18], robustness-oriented loss functions [BHSFs19], [BKV+20], [CLC+19], or self-supervision from auxiliary tasks [CLC+20], [HMKS19], [ZYC+20] during (pre-)training. Our chapter also focuses on robustness through self-supervision, where we introduce multi-task learning with the auxiliary task of depth estimation as a method to improve a semantic segmentation DNN's robustness, which can be seen as an extension of [KBFs20]. Thereby, we achieve an improved performance, while also being more robust and even introducing a second useful task for scene perception. Compared to [KBFs20], we improve the multi-task learning strategy to reduce the need for manual task weighting and provide experiments across a wider range of datasets.

3 Multi-task Training Approach

Describing our multi-task learning approach, we start by defining our primary semantic segmentation task, followed by the auxiliary task of depth estimation. Finally, we describe how both tasks are trained in a multi-task learning setup as shown in Fig. 2.

Primary semantic segmentation: Semantic segmentation is defined as the pixel-wise classification of an input image $\mathbf{x}_t = (\mathbf{x}_{t,i}) \in \mathcal{G}^{H \times W \times C}$ at time t, with height H, width W, number of channels $C = 3$, and $\mathcal{G} = \{0, 1, ..., 255\}$ (cf. top branch of Fig. 2). Accordingly, for each pixel $\mathbf{x}_{t,i} \in \mathcal{G}^3$ with pixel index $i \in \mathcal{I} = \{1, ..., H \cdot W\}$ an output $\mathbf{y}_{t,i} = (y_{t,i,s}) \in \mathbb{I}^{|\mathcal{S}|}, \mathbb{I} = [0, 1]$ is predicted, which can be interpreted as the posterior probabilities that the pixel at index i belongs to a class $s \in \mathcal{S}$ from the set of classes $\mathcal{S} = \{1, 2, ..., |\mathcal{S}|\}$, with the number of classes $|\mathcal{S}|$. The final predicted semantic class $m_{t,i} \in \mathcal{S}$ is determined by applying the argmax operation $m_{t,i} = \text{argmax}_{s \in \mathcal{S}} y_{t,i,s}$. Thereby, the network output $\mathbf{y}_t \in \mathbb{I}^{H \times W \times |\mathcal{S}|}$ is converted to a segmentation mask $\mathbf{m}_t = (m_{t,i}) \in \mathcal{S}^{H \times W}$.

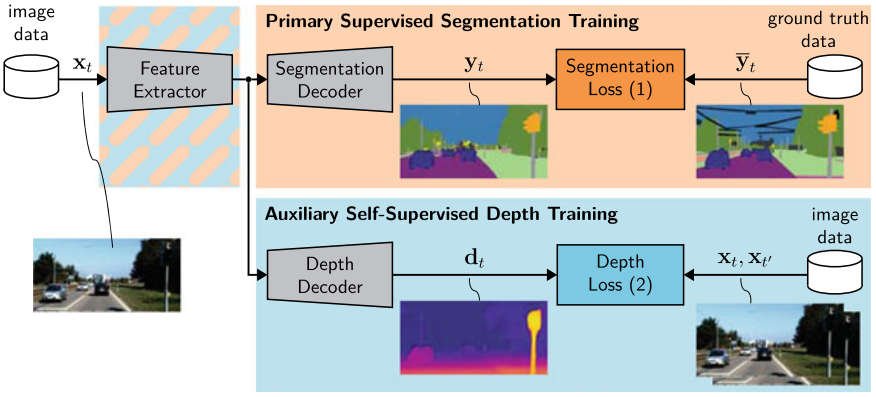


Fig. 2 Multi-task training setup across domains for the joint learning of depth estimation and semantic segmentation. While the semantic segmentation is trained on labeled image pairs, the depth estimation is trained on unlabeled image sequences

As shown in Fig. 2, training of a semantic segmentation network requires ground truth labels $\overline{\mathbf{y}}_t \in \{0, 1\}^{H \times W \times |\mathcal{S}|}$, which are derived from the ground truth segmentation mask $\overline{\mathbf{m}}_t \in \mathcal{S}^{H \times W}$ represented by a one-hot encoding. These are utilized to train the network by a cross-entropy loss function as

$$J_t^{\text{seg}} = -\frac{1}{H \cdot W} \sum_{i \in \mathcal{I}} \sum_{s \in \mathcal{S}} w_s \overline{y}_{t,i,s} \cdot \log\left(y_{t,i,s}\right), \tag{1}$$

where w_s are class-wise weights obtained as outlined in [PCKC16], and $\overline{y}_{t,i,s} \in \{0, 1\}$ are the single elements from the one-hot encoded ground truth tensor $\overline{\mathbf{y}}_t = (\overline{y}_{t,i,s})$.

Auxiliary depth estimation: Aiming at a more robust feature representation, we employ the auxiliary depth estimation task, which can be trained on unlabeled image sequences, as shown in Fig. 2, bottom part. During training, the depth estimation DNN predicts a depth map $\mathbf{d}_t = (d_{t,i}) \in \mathbb{D}^{H \times W}$, representing the distance of each pixel from the camera plane, where $\mathbb{D} = [d_{\min}, d_{\max}]$ represents the space of considered depth values constrained between the lower bound $d_{\min}$ and the upper bound $d_{\max}$. We optimize the depth estimation DNN in a self-supervised fashion by considering two loss terms: First, we make use of the image reconstruction term J_t^{ph} (photometric loss), which essentially uses the depth estimation in conjunction with the relative pose between two images, to optimize the camera reprojection models between two consecutive frames of a video. Second, we apply a smoothness loss term J_t^{sm}, allowing high gradients in the depth estimate's values only in image regions with high color gradients, such that the total loss can be written as

$$J_t^{\text{depth}} = J_t^{\text{ph}} + \beta J_t^{\text{sm}}, \tag{2}$$

where $\beta = 10^{-3}$ is adopted from previous works [CPMA19], [GMAFB19], [KTMFs20].

To optimize the network, we rely solely on sequential pairs of images $\mathbf{x}_t, \mathbf{x}_{t'}$ with $t' \in \mathcal{T}' = \{t-1, t+1\}$, which are taken from a video. These pairs are passed to an additional pose network, which predicts the relative poses $\mathbf{T}_{t\to t'} \in SE(3)$ between the image pairs, where $SE(3)$ is the special Euclidean group representing the set of all possible rotations and translations [Sze10]. The network predicts this transformation in an axis-angle representation such that only the six degrees of freedom are predicted, which are canonically converted to a 4×4 matrix for further processing. By letting the depth network predict the depth $\mathbf{d}_t$, both outputs, i.e., the depth $\mathbf{d}_t$ and the poses $\mathbf{T}_{t\to t'}$, can be used to project the image frame $\mathbf{x}_{t'}$ at time t' onto the pixel coordinates of the image frame $\mathbf{x}_t$, which results in two projected images $\mathbf{x}_{t'\to t}$, $t' \in \mathcal{T}'$ (for a detailed description, we refer to [KTMFs20]). Conclusively, the reprojection model can be optimized by minimizing the pixel-wise distance between the projected images $\mathbf{x}_{t'\to t} = (\mathbf{x}_{t'\to t,i})$ and the actual images $\mathbf{x}_t = (\mathbf{x}_{t,i})$ as

$$J_t^{\text{ph}} = \frac{1}{|\mathcal{I}|} \sum_{i\in\mathcal{I}} \min_{t'\in\mathcal{T}'} \left(\frac{\gamma}{2} \left(1 - \text{SSIM}_i\left(\mathbf{x}_t, \mathbf{x}_{t'\to t}\right)\right) + (1-\gamma)\, \frac{1}{C} \left\| \mathbf{x}_{t,i} - \mathbf{x}_{t'\to t,i} \right\|_1 \right). \tag{3}$$

This so-called minimum reprojection loss or photometric loss [GMAFB19] utilizes a mixture of the structural similarity (SSIM) difference term $\text{SSIM}_i(\cdot) \in \mathbb{I}$, with $\mathbb{I} = [0, 1]$, and is computed on 3×3 patches of the input, and an L_1 difference term $\|\cdot\|_1$ computed over all $C = 3$ gray value channels. For optimal absolute prediction performance, the terms are weighted by a factor $\gamma = 0.85$, chosen as in previous approaches [CPMA19], [GMAFB19], [KTMFs20], [YS18]. The depth and pose networks are then implicitly optimized, as their outputs are the parameters of the projection model, used to obtain the projected image $\mathbf{x}_{t'\to t}$, whose distance to the actual image $\mathbf{x}_t$ is minimized by (3).

As the photometric loss J_t^{ph} does not enforce a relationship in the depth map between depth values of neighboring pixels, we use a second smoothness loss term J_t^{sm} [GMAB17], allowing non-smoothness in the depth map $\mathbf{d}_t$ only in image locations with strong color gradients. This loss is computed on the mean-normalized inverse depth $\check{\boldsymbol{\rho}}_t \in \mathbb{R}^{H\times W}$, whose elements can be obtained from the depth map as $\check{\rho}_{t,i} = \frac{\rho_{t,i}}{\frac{1}{HW}\sum_{j\in\mathcal{I}} \rho_{t,j}}$ with $\rho_{t,i} = \frac{1}{d_{t,i}}$. Thereby, the loss can be formulated as

$$J_t^{\text{sm}} = \frac{1}{|\mathcal{I}|} \sum_{i\in\mathcal{I}} \left(|\partial_h \check{\rho}_{t,i}| \exp\left(-\frac{1}{C} \left\| \partial_h \mathbf{x}_{t,i} \right\|_1 \right) + |\partial_w \check{\rho}_{t,i}| \exp\left(-\frac{1}{C} \left\| \partial_w \mathbf{x}_{t,i} \right\|_1 \right) \right), \tag{4}$$

with one-dimensional difference quotients $\partial_h \check{\rho}_{t,i} = \partial_h \check{\rho}_{t,(h,w)} = \check{\rho}_{t,(h,w)} - \check{\rho}_{t,(h-1,w)}$ and $\partial_w \check{\rho}_{t,i} = \partial_w \check{\rho}_{t,(h,w)} = \check{\rho}_{t,(h,w)} - \check{\rho}_{t,(h,w-1)}$ defined with respect to the height and width dimensions of the image, respectively. The indices h and w represent the exact pixel position in two dimensions, where $h \in \{2, ..., H\}$ and $w \in \{2, ..., W\}$, where we exclude $h = 1$ and $w = 1$ to ensure the existence of a preceding pixel in height and width dimensions.

Multi-task learning: To train our model directly in a one-stage fashion without a pre-trained semantic segmentation network, we choose a multi-task setup with a shared encoder and two task-specific decoder heads as shown in Fig. 2. The decoders for semantic segmentation and depth estimation are optimized for their respective tasks according to the losses defined in (1) and (2), respectively, while the encoder is optimized for both tasks. As in [GL15], [KTMFs20], we let the task-specific gradients $\mathbf{g}_t^{\text{depth}}$ and $\mathbf{g}_t^{\text{seg}}$ propagate unscaled in the respective decoders, while their influence when reaching the encoder layers during backpropagation are scaled as

$$\mathbf{g}_t^{\text{total}} = \lambda^{\text{depth}} \mathbf{g}_t^{\text{depth}} + \lambda^{\text{seg}} \mathbf{g}_t^{\text{seg}}, \tag{5}$$

where the scalar weight λ^{seg} and λ^{depth} determine the weighting between the two tasks. In the other encoder layers, the backpropagation is then executed as usual. By scaling the gradients instead of the losses, the two decoders can learn optimal task-specific weights, while their influence on the shared encoder can be scaled to the optimal ratio using (5). Thereby, the encoder does not only learn optimal features for the semantic segmentation task, but can also take profit from the additional data accessible by the depth estimation, which can be trained on arbitrary videos.

In this chapter, we compare two kinds of task weightings: First, we apply the gradient weighting (GW) from [KBFs20], where we set $\lambda^{\text{seg}} = \lambda$ and $\lambda^{\text{depth}} = 1 - \lambda$ to scale the influence of both tasks. Here, the hyperparameter λ needs to be tuned to find the optimal weighting of both tasks. However, the results from [KBFs20] show that for a badly chosen hyperparameter λ, performance decreases drastically, which is why we apply the GradNorm (GN) multi-task learning technique [CBLR18], where the scale factors λ^{seg} and λ^{depth} are reformulated as learnable parameters $\lambda^{\text{seg}}(\tau)$ and $\lambda^{\text{depth}}(\tau)$ which are adapted at each learning step τ. For simplicity, we henceforth abbreviate the tasks with an index k with $k \in \mathcal{K} = \{\text{depth}, \text{seg}\}$. Thereby, the loss function used for optimization of the scale factors, i.e., the task weights, is calculated as follows:

$$J_t^{\text{GN}}(\tau) = \sum_{k \in \mathcal{K}} \left| G_t^{(k)}(\tau) - \left(\frac{1}{|\mathcal{K}|} \sum_{\kappa \in \mathcal{K}} G_t^{(\kappa)}(\tau) \right) \cdot \left(r_t^{(k)}(\tau) \right)^{\alpha} \right|. \tag{6}$$

It depends on the magnitude $G_t^{(k)}(\tau) = \left\| \lambda^{(k)}(\tau) \mathbf{g}_t^{(k)}(\tau) \right\|_2$ of the task-specific gradients $\mathbf{g}_t^{(k)}(\tau)$ in the last shared layer and the task-specific training rates $r_t^{(k)}(\tau) = \tilde{J}_t^{(k)}(\tau) \cdot \left(\frac{1}{|\mathcal{K}|} \sum_{\kappa \in \mathcal{K}} \tilde{J}_t^{(\kappa)}(\tau) \right)^{-1}$ with $\tilde{J}_t^{(k)}(\tau) = J_t^{(k)}(\tau) \cdot \left(J_t^{(k)}(0) \right)^{-1}$, depending on the value of the task-specific losses $J_t^{(k)}(\tau)$ taken from (1) and (2) at each step τ in comparison to their values $J_t^{(k)}(0)$ at step $\tau = 0$. These training rates can be interpreted as the convergence progress of the single tasks. Note that through the loss for the scaling factors in (6) a similar and stable convergence speed of both tasks is encouraged, which is essential for a successful multi-task training. The network is optimized with alternating updates of the scaling factors $\lambda^{(k)}(\tau)$ by (6) and the network weights by (1) and (2). Although a more stable convergence is generally

expected, one can still optimize GradNorm (GN) with the hyperparameter α in (6), controlling the restoring force back to balanced training rates. Also, after each step, the task weights are renormalized such that $\lambda^{\text{seg}}(\tau) + \lambda^{\text{depth}}(\tau) = 1$.

4 Evaluation Setup

In this section, we present the input perturbations used to evaluate the model robustness, define the perturbation strength, and detail our chosen evaluation metrics. Finally, we provide an overview of our chosen databases and the implementation details of our method. The evaluation setup is depicted in Fig. 3.

Image perturbations: After the semantic segmentation network has been trained (either by single-task or multi-task training), we evaluate its robustness during inference by adding a perturbation $\mathbf{r}_\epsilon$ of strength ϵ to each input image $\mathbf{x}$, yielding a perturbed input image $\mathbf{x}_\epsilon = \mathbf{x} + \mathbf{r}_\epsilon$. For simplicity, we omit the subscript t as the evaluation is carried out on single images. We conduct experiments using two noise types and two adversarial attacks, for which we aim at imposing a perturbation strength ϵ to make different perturbation types comparable. We measure this comparable strength by the signal-to-noise ratio (SNR) on the input image $\mathbf{x}^{\text{adv}}$, which is defined as

$$\text{SNR} = \frac{\text{E}\left(\|\mathbf{x}\|_2^2\right)}{\text{E}\left(\|\mathbf{r}_\epsilon\|_2^2\right)}. \tag{7}$$

Here, $\text{E}\left(\|\mathbf{x}\|_2^2\right)$ is the expectation value of the sum of the image's squared gray values, and $\text{E}\left(\|\mathbf{r}_\epsilon\|_2^2\right)$ is the expectation value of the sum of the squared noise pixels. As $\text{E}\left(\|\mathbf{x}\|_2^2\right)$ is always equal for different perturbation types, we define the perturbation's strength in dependency of $\text{E}\left(\|\mathbf{r}_\epsilon\|_2^2\right)$ as

$$\epsilon = \sqrt{\frac{1}{HWC}\,\text{E}\left(\|\mathbf{r}_\epsilon\|_2^2\right)}. \tag{8}$$

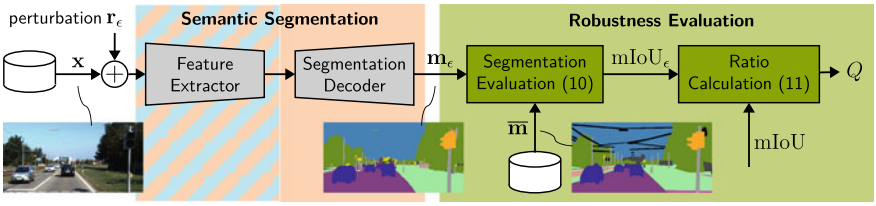


Fig. 3 Evaluation setup using additive perturbations to evaluate the robustness of a segmentation DNN. As perturbations, we use various noise and attack types to simulate deployment conditions

We consider Gaussian noise, salt and pepper noise, the fast gradient sign method (FGSM) attack [GSS15], and the projected gradient descent (PGD) attack [KGB17]. For Gaussian noise, the perturbation strength can be identified as the standard deviation of the underlying Gaussian distribution. For salt and pepper noise, on the other hand, some pixels are randomly set to 0 or 255. Therefore, first the input is perturbed, then the perturbation is obtained by $\mathbf{r}_\epsilon = \mathbf{x}_\epsilon - \mathbf{x}$, and finally the perturbation strength is computed by (8).

For the FGSM adversarial attack, the perturbation is calculated according to

$$\mathbf{x}_\epsilon = \mathbf{x} + \epsilon \cdot \mathbf{sign}\left(\nabla_{\mathbf{x}} J^{\mathrm{ce}}\left(\overline{\mathbf{y}}, \mathbf{y}\left(\mathbf{x}\right)\right)\right), \tag{9}$$

where $\nabla_{\mathbf{x}}$ represents the derivative of the loss function with respect to the unperturbed image $\mathbf{x}$, and $\mathbf{sign}(\cdot)$ represents the signum function applied to each element of its vectorial argument. As all elements $r_{\epsilon,j}$ of the perturbation $\mathbf{r}_\epsilon$ can only take on values $r_{\epsilon,j} = \pm\epsilon$, $j \in \{1, ..., H \cdot W \cdot C\}$, the perturbation variance is equal to ϵ^2 when applying (8). We also consider the PGD adversarial attack, due to its more advanced attack design, which is optimized over several optimization steps. This attack can be interpreted as an iterative version of (9) and allows investigations of the network robustness w.r.t. stronger adversarial attacks.

Evaluation metrics: To evaluate DNN performance on a perturbed input image $\mathbf{x}_\epsilon$, we pass this image to the DNN and generate the output $\mathbf{m}_\epsilon$, see Fig. 3. The output quality for a perturbation with $\epsilon > 0$ is typically worse than the output generated from clean images $\mathbf{x}_{\epsilon=0}$. The absolute segmentation performance can then be obtained by calculating the mean intersection-over-union metric [EVGW+15] as

$$\mathrm{mIoU}_\epsilon = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{\mathrm{TP}_{\epsilon,s}}{\mathrm{TP}_{\epsilon,s} + \mathrm{FP}_{\epsilon,s} + \mathrm{FN}_{\epsilon,s}}, \tag{10}$$

where the number of true positives $\mathrm{TP}_{\epsilon,s}$, false positives $\mathrm{FP}_{\epsilon,s}$, and false negatives $\mathrm{FN}_{\epsilon,s}$ for each class s are calculated over the entire evaluation subset before the application of (10).

As different models may have a different absolute prediction performance $\mathrm{mIoU}_{\epsilon=0}$ on clean images, a better performance on perturbed images can either mean that the model is better in general or more robust. Therefore, we rather compare the mIoU ratio

$$Q = \frac{\mathrm{mIoU}_\epsilon}{\mathrm{mIoU}_{\epsilon=0}} \tag{11}$$

obtained from the performance mIoU_ϵ on perturbed images with a perturbation of strength ϵ, normalized by the performance on clean images.

Databases: The additional training with the auxiliary task of depth estimation in a self-supervised fashion is applied jointly with the training of the primary semantic segmentation task. To accomplish this, for training, we always rely on an unlabeled image sequence dataset (bottom part of Table 1(a)) to train the depth estimation and

Table 1 (**a**) Overview on the employed datasets with their respective subsets; (**b**) the employed image perturbations used to evaluate the robustness of the DNN (right-hand side)

(**a**) Databases; number of images

Dataset	Symbol	Training	Validation	Test
Cityscapes [COR+16]	$\mathcal{X}^{\text{CS}}$	2,975	500	1525
KITTI [GLSU13, MG15]	$\mathcal{X}^{\text{KIT}}$	28,937	200	200
Cityscapes (seq.) [COR+16]	$\mathcal{X}^{\text{CS,seq}}$	83,300	–	–

(**b**) Perturbations

Perturbation	Type			
Gaussian noise	Random noise			
Salt and pepper noise	Random noise			
FGSM [GSS15]	Adversarial attack			
PGD [KGB17]	Adversarial attack			

one labeled image dataset (top part of Table 1(a)) to train the semantic segmentation. Both datasets can be from different domains, as [KTMFs20] showed that a successful training for both tasks is also possible then. For segmentation training, we use Cityscapes [COR+16] ($\mathcal{X}^{\text{CS}}_{\text{train}}$), while for depth estimation training we either use Cityscapes [COR+16] ($\mathcal{X}^{\text{CS,seq}}_{\text{train}}$) or KITTI [GLSU13] ($\mathcal{X}^{\text{KIT}}_{\text{train}}$), with the KITTI training split defined by [GMAB17]. Note that each image of the Cityscapes segmentation dataset contains 19 preceding and 10 succeeding unlabeled image frames, which we use for the depth estimation training. The number of training images for the depth estimation training splits deviates slightly from the original definitions due to the need of a preceding and a succeeding frame. For evaluation, we use the validation set from Cityscapes ($\mathcal{X}^{\text{CS}}_{\text{val}}$), as the test sets are not publicly available. We also evaluate on the training set from the KITTI 2015 Stereo dataset [MG15] ($\mathcal{X}^{\text{KIT}}_{\text{val}}$), which is disjoint from the training split as outlined in [GMAB17].

Implementation details: All models as well as training and evaluation protocols are implemented in `PyTorch` [PGM+19] and executed on an `NVIDIA Tesla P100` graphics card. Same as [KTMFs20], we choose an encoder-decoder multi-task network architecture based on a `ResNet18` feature extractor [HZRS16] pre-trained on ImageNet [RDS+15] and two task-specific decoder heads. These heads have an identical architecture except for the last layer: For depth estimation we employ a sigmoid output $\boldsymbol{\sigma} \in \mathbb{I}^{H \times W}$, which is converted to depth values in a pixel-wise fashion as $\frac{1}{a\sigma_i + b}$, where a and b define the depth to the considered range [0.1, 100]. The segmentation output logits are comprised of $S = |\mathcal{S}|$ feature maps which are converted to class probabilities via a softmax function. The pose network, which is required to train the depth estimation in a self-supervised fashion on videos, utilizes the same network architecture as in [GMAFB19].

We resize all images used for depth estimation training to a resolution of 640 × 192, while the images used for segmentation training are resized to 512 × 1024 and cropped to the same resolution. We train the network for 40 epochs using the Adam optimizer [KB15] with a learning rate of 10^{-4}, which is reduced to 10^{-5} for the last 10 epochs. To ensure fairness, we use batch sizes of 12 and 6 for the single-task model and the two tasks of the multi-task model, respectively. Moreover, gradient scaling (5) is applied at all connections between the encoder and the decoder. For further training details the interested reader is referred to [KTMFs20].

5 Experimental Results and Discussion

In this section, we will first analyze the multi-task learning method w.r.t. the achieved absolute performance, where we will put the focus on how the performance can be improved without extensive hyperparameter tuning. Afterwards, the resulting models will be analyzed w.r.t. robustness against common image corruptions such as random noise or adversarial attacks.

Absolute performance: While the main focus of this chapter is the improved robustness of semantic segmentation DNNs, this robustness improvement should not come at the price of a significantly decreased absolute performance. In [KBFs20] it was proposed to scale the gradients using a manually tunable hyperparameter (GW). However, the results from Table 2 show that the absolute performance can decrease significantly, e.g., for GW ($\lambda = 0.9$) on the KITTI dataset (right column in both tables), compared to the single-task baseline. As there is no general way to know which task weighting is optimal, we propose to use the GradNorm (GN) technique [CBLR18] instead of manual gradient weighting (GW). For this technique we observe that for all considered GN hyperparameters α and on both dataset settings, the GradNorm technique improves the absolute performance over the single-task baseline. Interestingly, for this task weighting strategy, the task weights change over the course of the learning process. In particular, side experiments showed that the final task weights at the end of the training process do yield a decreased performance, if they are used constantly throughout the whole training process. This shows the importance of adapting them in dependence of the task-specific training progresses. Still, optimal performance is sometimes rather reached with manual gradient weighting (e.g., Table 2a), however, for robustness purposes an optimal absolute performance is not as important as a stable convergence of the multi-task training. We therefore use the GradNorm technique (GN) instead of the manual gradient weighting for all following experiments w.r.t. DNN robustness.

Robustness to input noise: As a simple next experiment, we compared the robustness w.r.t. Gaussian input noise between the baseline trained in a single-task fashion and models trained in a multi-task fashion with the GradNorm method. The results in Table 3a are obtained for a setting, where the segmentation and depth tasks are trained

Table 2 Absolute segmentation performance measured by the mIoU [%] metric for models where the segmentation is trained on Cityscapes ($\mathcal{X}^{\text{CS}}_{\text{train}}$) and the auxiliary depth estimation is either trained on KITTI ($\mathcal{X}^{\text{KIT}}_{\text{train}}$, top) or Cityscapes ($\mathcal{X}^{\text{CS,seq}}_{\text{train}}$, bottom). We report numbers on the Cityscapes ($\mathcal{X}^{\text{CS}}_{\text{val}}$) and KITTI ($\mathcal{X}^{\text{KIT}}_{\text{val}}$) validation sets for manual gradient weighting (GW) and the GradNorm (GN) multi-task training method. Best numbers are in boldface. Second best numbers are underlined

(**a**) Segmentation: $\mathcal{X}^{\text{CS}}_{\text{train}}$, depth: $\mathcal{X}^{\text{KIT}}_{\text{train}}$

Method	mIoU on $\mathcal{X}^{\text{CS}}_{\text{val}}$	mIoU on $\mathcal{X}^{\text{KIT}}_{\text{val}}$
Baseline	63.5	43.0
GW ($\lambda = 0.1$)	<u>67.4</u>	**49.6**
GW ($\lambda = 0.2$)	**68.9**	47.7
GW ($\lambda = 0.5$)	<u>67.4</u>	34.7
GW ($\lambda = 0.9$)	67.7	29.8
GN ($\alpha = 0.2$)	66.8	44.0
GN ($\alpha = 0.5$)	67.0	47.0
GN ($\alpha = 1.0$)	66.4	<u>48.5</u>
GN ($\alpha = 2.0$)	65.7	45.6

(**b**) Segmentation: $\mathcal{X}^{\text{CS}}_{\text{train}}$, depth: $\mathcal{X}^{\text{CS,seq}}_{\text{train}}$

Method	mIoU on $\mathcal{X}^{\text{CS}}_{\text{val}}$	mIoU on $\mathcal{X}^{\text{KIT}}_{\text{val}}$
Baseline	63.5	43.0
GW ($\lambda = 0.1$)	65.8	<u>47.2</u>
GW ($\lambda = 0.2$)	66.6	46.0
GW ($\lambda = 0.5$)	65.1	44.9
GW ($\lambda = 0.9$)	66.1	40.5
GN ($\alpha = 0.2$)	66.1	46.1
GN ($\alpha = 0.5$)	**67.9**	**48.3**
GN ($\alpha = 1.0$)	<u>67.1</u>	45.5
GN ($\alpha = 2.0$)	66.5	**48.3**

on Cityscapes and KITTI, respectively. We clearly observe that all multi-task models (GN variants) exhibit a higher robustness measured by the mIoU ratio Q (11), e.g., $Q = 27.5\%$ to $Q = 33.3\%$ for a perturbation strength of $\epsilon = 16$, compared to the single task baseline ($Q = 18.4\%$). The model variant GN ($\alpha = 0.5$) is furthermore shown in Fig. 4. When looking at the curves for the Cityscapes and KITTI validation sets and for both Gaussian and salt and pepper noise, we observe that the multi-task model is consistently either on par or better w.r.t. robustness, measured by the mIoU ratio Q.

A slightly different behavior is observed when the auxiliary depth estimation task is trained on the same dataset as the semantic segmentation task, as shown in Table 3b. Here, for small perturbation strengths ($\epsilon \leq 4$), the robustness is still improved, however, for larger perturbation strengths, the robustness is either similar or even decreased. This can be interpreted as an indication that the additional data

Table 3 Robustness to Gaussian input noise measured by the mIoU ratio Q [%] (11) for models where the segmentation is trained on Cityscapes ($\mathcal{X}_{\text{train}}^{\text{CS}}$) and the auxiliary depth estimation is either trained on KITTI ($\mathcal{X}_{\text{train}}^{\text{KIT}}$, top) or Cityscapes ($\mathcal{X}_{\text{train}}^{\text{CS,seq}}$, bottom). Best numbers are in boldface

(**a**) Segmentation: $\mathcal{X}_{\text{train}}^{\text{CS}}$, depth: $\mathcal{X}_{\text{train}}^{\text{KIT}}$

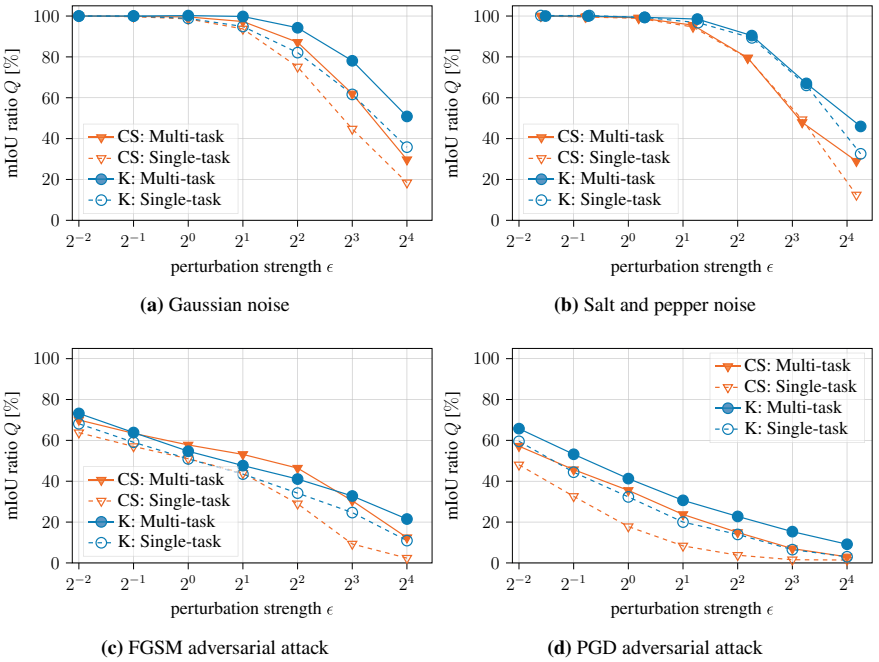
ϵ	0.25	0.5	1	2	4	8	16
Baseline	**100.0**	99.7	98.6	93.7	75.1	44.7	18.4
GN ($\alpha = 0.2$)	**100.0**	99.9	99.0	96.1	85.5	60.8	28.1
GN ($\alpha = 0.5$)	**100.0**	99.9	**99.6**	**97.3**	87.2	61.9	29.7
GN ($\alpha = 1.0$)	**100.0**	**100.0**	99.5	97.1	**88.1**	**65.4**	**33.3**
GN ($\alpha = 2.0$)	**100.0**	99.8	99.2	96.2	86.2	60.7	27.5

(**b**) Segmentation: $\mathcal{X}_{\text{train}}^{\text{CS}}$, depth: $\mathcal{X}_{\text{train}}^{\text{CS,seq}}$

ϵ	0.25	0.5	1	2	4	8	16
Baseline	**100.0**	99.7	98.6	93.7	75.1	44.7	**18.4**
GN ($\alpha = 0.2$)	**100.0**	99.8	**99.4**	**96.8**	82.0	**45.4**	13.0
GN ($\alpha = 0.5$)	99.9	99.7	99.1	95.3	79.4	41.8	10.2
GN ($\alpha = 1.0$)	**100.0**	**99.9**	**99.4**	96.3	**82.4**	42.5	15.6
GN ($\alpha = 2.0$)	99.8	99.7	99.1	96.2	82.0	44.5	8.0

from another domain is mainly responsible for the robustness improvement rather than the additional task itself. However, the self-supervised training technique of the auxiliary task is the precondition for being able to make use of additional unlabeled data.

Robustness to adversarial attacks: In addition to simple noise conditions, we also investigate adversarial attacks, where the noise pattern is optimized to fool the network. In Table 4 we show results on robustness w.r.t. the FGSM adversarial attack. We again observe that when the auxiliary task is trained in a different domain (left-hand side), the robustness is significantly improved regardless of the perturbation strength. In contrast to simple noise perturbations, the FGSM attack can degrade performance even for very small and visually hard-to-perceive perturbation strengths ($\epsilon \leq 1$), for which robustness is still improved by our method. Moreover, we again observe that the robustness improvement is not as good, when the auxiliary depth task is trained in the same domain (right-hand side), as here the robustness improvement is not as high. For instance, for $\epsilon = 8$ the robustness improves from $Q = 9.6\%$ to $Q = 33.4\%$ for the best multi-task model, when the depth is trained out-of-domain, while for in-



(a) Gaussian noise

(b) Salt and pepper noise

(c) FGSM adversarial attack

(d) PGD adversarial attack

Fig. 4 Robustness to several input perturbation types for models where the segmentation is trained on Cityscapes ($\mathcal{X}^{\text{CS}}_{\text{train}}$) and the auxiliary depth estimation on KITTI ($\mathcal{X}^{\text{KIT}}_{\text{train}}$). We report mIoU ratios Q [%] (11) on the Cityscapes ($\mathcal{X}^{\text{CS}}_{\text{val}}$) and KITTI ($\mathcal{X}^{\text{KIT}}_{\text{val}}$) validation sets for the GradNorm ($\alpha = 0.5$) multi-task training method

domain training the improvement is only from $Q = 9.6\%$ to $Q = 21.8\%$. Still, for the FGSM attack all multi-task models improve upon the baseline in terms of robustness, showing the general applicability of the GradNorm multi-task learning technique for robustness improvement.

To also investigate this effect on a wider range of datasets and perturbations, we show robustness results on the Cityscapes and KITTI validation sets and for the FGSM and PGD adversarial attack in Fig. 4, bottom. For all considered cases, the robustness of the multi-task model GN ($\alpha = 0.5$) improves upon the single-task baseline. We also show qualitative results in Fig. 5 for Gaussian noise ($\epsilon = 8.0$) and the FGSM attack ($\epsilon = 4.0$) for the GN model ($\alpha = 0.5$), where we also qualitatively observe a significant improvement over the single-task baseline.

Table 4 Robustness to the FGSM adversarial attack measured by the mIoU ratio Q [%] (11) for models where the segmentation is trained on Cityscapes ($\mathcal{X}_{\text{train}}^{\text{CS}}$) and the auxiliary depth estimation is either trained on KITTI ($\mathcal{X}_{\text{train}}^{\text{KIT}}$, top) or Cityscapes ($\mathcal{X}_{\text{train}}^{\text{CS,seq}}$, bottom). Best numbers are in boldface

(**a**) Segmentation: $\mathcal{X}_{\text{train}}^{\text{CS}}$, depth: $\mathcal{X}_{\text{train}}^{\text{KIT}}$

ϵ	0.25	0.5	1	2	4	8	16
Baseline	63.8	57.0	51.0	43.9	29.6	9.6	2.4
GN ($\alpha = 0.2$)	**73.0**	**67.2**	**62.3**	**58.7**	**51.2**	32.6	14.4
GN ($\alpha = 0.5$)	70.0	63.4	57.8	53.1	46.4	30.4	12.4
GN ($\alpha = 1.0$)	70.3	63.9	59.3	55.8	49.2	**33.4**	13.3
GN ($\alpha = 2.0$)	72.1	66.1	61.5	57.5	44.5	31.1	**14.6**

(**b**) Segmentation: $\mathcal{X}_{\text{train}}^{\text{CS}}$, depth: $\mathcal{X}_{\text{train}}^{\text{CS,seq}}$

ϵ	0.25	0.5	1	2	4	8	16
Baseline	63.8	57.0	51.0	43.9	29.6	9.6	2.4
GN ($\alpha = 0.2$)	71.4	65.7	61.0	**57.5**	**46.9**	**21.8**	**7.7**
GN ($\alpha = 0.5$)	**74.3**	66.0	61.1	56.7	44.5	17.7	5.6
GN ($\alpha = 1.0$)	72.6	**66.3**	**61.5**	57.2	40.8	16.4	**7.7**
GN ($\alpha = 2.0$)	72.2	66.0	**61.5**	57.1	44.5	16.2	2.9

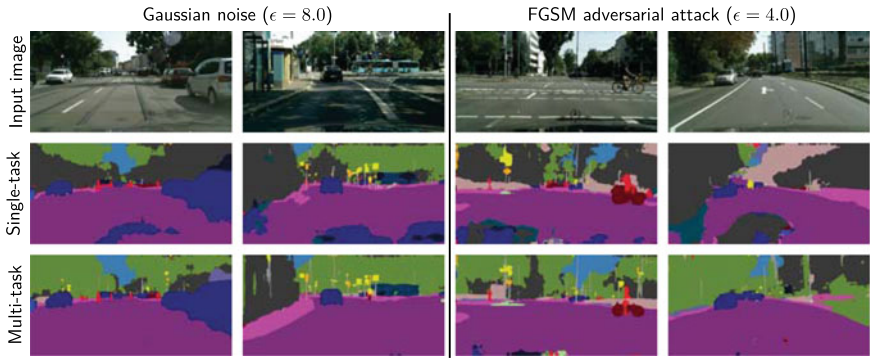


Fig. 5 Qualitative result comparison for models where the segmentation is trained on Cityscapes ($\mathcal{X}_{\text{train}}^{\text{CS}}$) and the auxiliary depth estimation on KITTI ($\mathcal{X}_{\text{train}}^{\text{KIT}}$). We show results for two different perturbations and compare between the single-task baseline and the GradNorm ($\alpha = 0.5$) multi-task training method

6 Conclusions

In this chapter, we show how the robustness of a semantic segmentation DNN can be improved by multi-task training with the auxiliary task of depth estimation, which can be trained in a self-supervised fashion on arbitrary videos. We show this robustness improvement across two noise perturbations and two adversarial attacks, where we ensure comparability of different perturbations in terms of strength. By making use of the GradNorm task weighting strategy, we are able to remove the necessity for manual tuning of hyperparameters, thereby achieving a stable and easy-to-apply robustness improvement. Also, we show that in-domain training with the additional task of depth estimation already improves robustness to some degree, while out-of-domain training on additional unlabeled data enabled by the self-supervised training improves robustness even further. Moreover, our method is easy-to-apply, induces no computational overhead during inference, and even improves absolute performance, which can be of interest for applications such as autonomous driving, virtual reality, or medical imaging as long as additional video data is available. Our method thereby demonstrates various further advantages of multi-task training for semantic segmentation, which could be potentially generalizable to various further computer vision tasks.

References

[ACW18] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden, July 2018, pp. 274–283

[AMT18] A. Arnab, O. Miksik, P.H.S. Torr, On the robustness of semantic segmentation models to adversarial attacks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 888–897

[ASG+19] F. Assion, P. Schlicht, F. Greßner, W. Günther, F. Hüger, N.M. Schmidt, U. Rasheed, The attack generator: a systematic approach towards constructing adversarial attacks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Long Beach, CA, USA, June 2019, pp. 1370–1379

[ATT18] A.E. Aydemir, A. Temizel, T.T. Temizel, *The Effects of JPEG and JPEG2000 Compression on Attacks Using Adversarial Examples*, Mar. 2018, pp. 1–4. arxiv:1803.10418

[BHSFs19] A. Bär, F. Hüger, P. Schlicht, T. Fingscheidt, On the robustness of redundant teacher-student frameworks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Long Beach, CA, USA, June 2019, pp. 1380–1388

[Bis95] C.M. Bishop, Training with noise is equivalent to Tikhonov regularization. Neural Comput. **7**(1), 108–116 (1995)

[BKV+20] A. Bär, M. Klingner, S. Varghese, F. Hüger, P. Schlicht, T. Fingscheidt, Robust semantic segmentation by redundant networks with a layer-specific loss contribution and majority vote, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1348–1358, virtual conference, June 2020

[BLK+21] A. Bär, J. Löhdefink, N. Kapoor, S.J. Varghese, F. Hüger, P. Schlicht, T. Fingscheidt, The vulnerability of semantic segmentation networks to adversarial attacks in autonomous driving: enhancing extensive environment sensing. IEEE Signal Process. Mag. **38**(1), 42–52 (2021)

[CBG+17] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier, Parseval networks: improving robustness to adversarial examples, in *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, NSW, Australia, Aug. 2017, pp. 854–863

[CBLR18] Z. Chen, V. Badrinarayanan, C.-Y. Lee, A. Rabinovich, GradNorm: gradient normalization for adaptive loss balancing in deep multitask networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden, July 2018, pp. 794–803

[CLC+19] H.-Y. Chen, J.-H. Liang, S.-C. Chang, J.-Y. Pan, Y.-T. Chen, W. Wei, D.-C. Juan, Improving adversarial robustness via guided complement entropy, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea, Oct. 2019, pp. 4881–4889

[CLC+20] T. Chen, S. Liu, S. Chang, Y. Cheng, L. Amini, Z. Wang, Adversarial robustness: from self-supervised pre-training to fine-tuning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference, June 2020, pp. 699–708

[CLLW19] P.-Y. Chen, A.H. Liu, Y.-C. Liu, Y.-C.F. Wang, Towards scene understanding: unsupervised monocular depth estimation with semantic-aware representation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, pp. 2624–2632

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 3213–3223

[CPK+18] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **40**(4), 834–848 (2018)

[CPMA19] V. Casser, S. Pirk, R. Mahjourian, A. Angelova, Depth videos, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA, Jan. 2019, pp. 8001–8008

[CW17] N. Carlini, D.A. Wagner, Towards evaluating the robustness of neural networks, in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 39–57

[DLP+18] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, Boosting adversarial attacks with momentum, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 9185–9193

[EF15] D. Eigen, R. Fergus, Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, Dec. 2015, pp. 2650–2658

[EPF14] D. Eigen, C. Puhrsch, R. Fergus, Depth map prediction from a single image using a multi-scale deep network, in *Proceedings of the Conference on Neural Information*

Processing Systems (NIPS/NeurIPS), Montréal, QC, Canada, Dec. 2014, pp. 2366–2374

[EVGW+15] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge: a retrospective. Int. J. Comput. Vis. (IJCV) **111**(1), 98–136 (2015)

[GHL+20] V. Guizilini, R. Hou, J. Li, R. Ambrus, A. Gaidon, Semantically-Guided representation learning for self-supervised monocular depth, in *Proceedings of the International Conference on Learning Representations (ICLR)*, virtual conference, Apr. 2020, pp. 1–14

[GL15] Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in *Proceedings of the International Conference on Machine Learning (ICML)*, Lille, France, July 2015, pp. 1180–1189

[GLSU13] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: the KITTI dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (2013)

[GMAB17] C. Godard, O.M. Aodha, G.J. Brostow, Unsupervised monocular depth estimation with left-right consistency, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, July 2017, pp. 270–279

[GMAFB19] C. Godard, O.M. Aodha, M. Firman, G.J. Brostow, Digging into self-supervised monocular depth estimation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea, Oct. 2019, pp. 3828–3838

[GRCvdM18] C. Guo, M. Rana, M. Cissé, L. van der Maaten, Countering adversarial images using input transformations, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, Apr. 2018, pp. 1–12

[GSS15] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, pp. 1–11 (2015)

[GW08] R.C. Gonzales, R.E. Woods, *Digital Image Processing* (Prentice Hall, 2008)

[HAMFs22] A.S. Hashemi, B. Andreas, S. Mozaffari, T. Fingscheidt, Improving transferability of generated universal adversarial perturbations for image classification and segmentation, in *Deep Neural Networks and Data for Automated Driving—Robustness, Uncertainty Quantification, and Insights Towards Safety*, ed. by T. Fingscheidt, H. Gottschalk, S. Houben, (Springer, 2022), pp. 195–222

[HK92] L. Holström, P. Koistinen, Using additive noise in backpropagation-training. IEEE Trans. Neural Netw. (TNN) **3**(1), 24–38 (1992). (January)

[HMKS19] D. Hendrycks, M. Mazeika, S. Kadavath, D. Song, Using self-supervised learning can improve model robustness and uncertainty, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada, Dec. 2019, pp. 15637–15648

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 770–778

[KB15] D.P. Kingma, J. Ba, ADAM: a method for stochastic optimization, in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15

[KBFs20] M. Klingner, A. Bär, T. Fingscheidt, Improved noise and attack robustness for semantic segmentation by using multi-task training with self-supervised depth estimation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, virtual conference, June 2020, pp. 1299–1309

[KBV+21] N. Kapoor, A. Bär, S. Varghese, J.D. Schneider, F. Hüger, P. Schlicht, T. Fingscheidt, From a Fourier-domain perspective on adversarial examples to a Wiener filter defense for semantic segmentation, in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, virtual conference, July 2021, pp. 1–8

[KGB17] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, in *Proceedings of the International Conference on Learning Representations (ICLR) Workshops*, Toulon, France, Apr. 2017, pp. 1–14

[KGC18] A. Kendall, Y. Gal, R. Cipolla, Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 7482–7491

[KTMFs20] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, T. Fingscheidt, Self-Supervised monocular depth estimation: solving the dynamic object problem by semantic guidance, in *Proceedings of the European Conference on Computer Vision (ECCV)*, virtual conference, Aug. 2020, pp. 582–600

[LJL+19] H. Liu, R. Ji, J. Li, B. Zhang, Y. Gao, Y. Wu, F. Huang, Universal adversarial perturbation via prior driven uncertainty approximation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, Korea, Oct. 2019, pp. 2941–2949

[LLKX19] P. Liu, M. Lyu, I. King, J. Xu, SelFlow: self-supervised learning of optical flow, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, pp. 4571–4580

[LLL+19] Z. Liu, Q. Liu, T. Liu, N. Xu, X. Lin, Y. Wang, W. Wen, Feature distillation: DNN-Oriented JPEG compression against adversarial examples, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, pp. 860–868

[LRLG19] K.-H. Lee, G. Ros, J. Li, A. Gaidon, SPIGAN: privileged adversarial learning from simulation, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, Apr. 2019, pp. 1–14

[LSD15] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015, pp. 3431–3440

[MCKBF17] J.H. Metzen, M.C. Kumar, T. Brox, V. Fischer, Universal adversarial perturbations against semantic image segmentation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017, pp. 2774–2783

[MDFFF17] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, Universal adversarial perturbations, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, July 2017, pp. 1765–1773

[MG15] M. Menze, A. Geiger, Object scene flow for autonomous vehicles, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015, pp. 3061–3070

[MGB17] K.R. Mopuri, U. Garg, R. Venkatesh Babu, Fast feature fool: a data independent approach to universal adversarial perturbations, in *Proceedings of the British Machine Vision Conference (BMVC)*, London, UK, Sept. 2017, pp. 1–12

[MGR19] K.R. Mopuri, A. Ganeshan, V.B. Radhakrishnan, Generalizable data-free objective for crafting universal adversarial perturbations. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **41**(10), 2452–2465 (2019)

[MLR+19] Y. Meng, Y. Lu, A. Raj, S. Sunarjo, R. Guo, T. Javidi, G. Bansal, D. Bharadia, SIGNet: semantic instance aided unsupervised 3D geometry perception, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, pp. 9810–9820

[MMS+18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, Apr. 2018, pp. 1–10

[NOBK17] G. Neuhold, T. Ollmann, S.R. Bulò, P. Kontschieder, The mapillary vistas dataset for semantic understanding of street scenes, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017, pp. 4990–4999

[NVA19] J. Novosel, P. Viswanath, B. Arsenali, Boosting semantic segmentation with multi-task self-supervised learning for autonomous driving applications, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops*, Vancouver, BC, Canada, Dec. 2019, pp. 1–11

[PCKC16] A. Paszke, A. Chaurasia, S. Kim, E. Culurciello, *ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation*, pp. 1–10, June 2016. arxiv:1606.02147

[PGM+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison et al., PyTorch: an imperative style, high-performance deep learning library, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Vancouver, BC, Canada, Dec. 2019, pp. 8024–8035

[PMG+17] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z.B. Celik, A. Swami, Practical black-box attacks against machine learning, in *Proceedings of the ACM ASIA Conference on Computer and Communications Security (ASIACSS)*, Abu Dhabi, UAE, Apr. 2017, pp. 506–519

[RDS+15] O. Russakovsky, J. Deng, S. Hao, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**(3), 211–252 (2015)

[RKKY+21a] V.R. Kumar, M. Klingner, S. Yogamani, M. Bach, S. Milz, T. Fingscheidt, P. Mäder, SVDistNet: self-supervised near-field distance estimation on surround view fisheye cameras. IEEE Trans. Intell. Transp. Syst. (TITS) 1–10, June 2021. early access

[RKKY+21b] V.R. Kumar, M. Klingner, S. Yogamani, S. Milz, T. Fingscheidt, P. Mäder, SynDistNet: self-supervised monocular fisheye camera distance estimation synergized with semantic segmentation for autonomous driving, in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, virtual conference, Jan. 2021, pp. 61–71

[RSFM19] E. Raff, J. Sylvester, S. Forsyth, M. McLean, Barrage of random transforms for adversarially robust defense, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, pp. 6528–6537

[Sze10] R. Szeliski. *Computer Vision: Algorithms and Applications* (Springer, 2010)

[SZS+14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, AB, Canada, Dec. 2014, pp. 1–10

[VJB+19] T.-H. Vu, H. Jain, M. Bucher, M. Cord, P. Perez, ADVENT: adversarial entropy minimization for domain adaptation in semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, pp. 2517–2526

[XOWS18] D. Xu, W. Ouyang, X. Wang, N. Sebe, PAD-Net: multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 675–684

[XWZ+18] C. Xie, J. Wang, Z. Zhang, Z. Ren, A. Yuille, Mitigating adversarial effects through randomization, in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–15, Vancouver, BC, Canada, Apr. 2018

[YS18] Z. Yin, J. Shi, GeoNet: unsupervised learning of dense depth, optical flow and camera pose, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 1983–1992

[YZS+18] G. Yang, H. Zhao, J. Shi, Z. Deng, J. Jia, SegStereo: exploiting semantic information for disparity estimation, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, Sept. 2018, pp. 636–651

[ZBL20] S. Zhu, G. Brazil, X. Liu, The edge of depth: explicit constraints between segmentation and depth, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference, June 2020, pp. 13116–13125

[ZBSL17] T. Zhou, M. Brown, N. Snavely, D.G. Lowe, Unsupervised learning of depth and Ego-Motion from video, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, July 2017, pp. 1851–1860

[ZCX+19] Z. Zhang, Z. Cui, C. Xu, Y. Yan, N. Sebe, J. Yang, Pattern-Affinitive propagation across depth, surface normal and semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, pp. 4106–4115

[ZYC+20] L. Zhang, M. Yu, T. Chen, Z. Shi, C. Bao, K. Ma, Auxiliary training: towards accurate and robust models, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference, June 2020, pp. 372–381

Improving Transferability of Generated Universal Adversarial Perturbations for Image Classification and Segmentation

Atiye Sadat Hashemi, Andreas Bär, Saeed Mozaffari, and Tim Fingscheidt

Abstract Although deep neural networks (DNNs) are high-performance methods for various complex tasks, e.g., environment perception in automated vehicles (AVs), they are vulnerable to adversarial perturbations. Recent works have proven the existence of universal adversarial perturbations (UAPs), which, when added to most images, destroy the output of the respective perception function. Existing attack methods often show a low success rate when attacking target models which are different from the one that the attack was optimized on. To address such weak transferability, we propose a novel learning criterion by combining a low-level feature loss, addressing the similarity of feature representations in the first layer of various model architectures, with a cross-entropy loss. Experimental results on ImageNet and Cityscapes datasets show that our method effectively generates universal adversarial perturbations achieving state-of-the-art fooling rates across different models, tasks, and datasets. Due to their effectiveness, we propose the use of such novel generated UAPs in robustness evaluation of DNN-based environment perception functions for AVs.

A. S. Hashemi · A. Bär (✉) · T. Fingscheidt
Institute for Communications Technology (IfN), Technische Universität Braunschweig, Schleinitzstr. 22, 38106 Braunschweig, Germany
e-mail: andreas.baer@tu-bs.de

A. S. Hashemi
e-mail: atiye.hashemi1991@gmail.com; atiye.hashemi@semnan.ac.ir

T. Fingscheidt
e-mail: t.fingscheidt@tu-bs.de

A. S. Hashemi · S. Mozaffari
Semnan University, Campus 1, 35131-19111 Semnan, Iran
e-mail: mozaffari@semnan.ac.ir; saeed_mozaffari@yahoo.com

S. Mozaffari
University of Windsor, 401 Sunset Ave, Windsor, ON N9B 3P4, Canada

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_6

1 Introduction

Reaching desired safety standards quickly is of utmost importance for automated vehicles (AVs), in particular, for their environment perception module. Recently, growing advancements in deep neural networks (DNNs) gave the means to researchers for solving real-world problems, specifically improving the state of the art in environment perception [GTCM20, NVM+19]. These networks can help AVs in understanding the environment, such as identifying traffic signs and detecting surrounding objects, by incorporating many sensors (e.g., camera, LiDAR, and RaDAR) to build an overall representation of the environment [WLH+20, FJGN20], or even to provide an end-to-end control for the vehicle [KBJ+20], see Fig. 1.

A large body of studies has addressed adversarial attacks [SZS+14, MDFF+18] and robustness enhancement [GRM+19, SOZ+18] of DNN architectures. In AVs, it was shown that adversarial signs could fool a commercial classification system in real-world driving conditions [MKMW19]. In addition to the vulnerability of image classifiers, Arnab et al. [AMT18] extensively analyzed the behavior of semantic segmentation architectures for AVs and illustrated their susceptibility to adversarial attacks. To reach the high level of automation, defined by the SAE standard J3016 [SAE18], car manufacturers have to consider various threats and perturbations targeting the AV systems.

In vision-related tasks, adversarial perturbations can be divided into two types: image-dependent (per-instance) adversarial perturbations and universal adversarial perturbations (UAPs). In image-dependent adversarial perturbations, a specific optimization has to be performed for each image individually to generate an adversarial example. In contrast, UAPs are more general perturbations in the sense that an additive single perturbation triggers all the images in a dataset to become adversarial examples. Therefore, they are much more efficient in terms of computation cost and time when compared to image-dependent adversarial attacks [CAB+20]. However, while showing a high success rate on the models they are optimized on, they lack transferability to other models.

In this chapter, we aim to specifically address the vulnerability of image classification and semantic segmentation systems as cornerstones of visual perception in automated vehicles. In particular, we aim at improving the transferability of task-specific universal adversarial perturbations. Our major contributions are as follows:

First, we present a comprehensive similarity analysis of features from several layers of different DNN classifiers.

Second, based on these findings, we propose a novel fooling loss function for generating universal adversarial perturbations. In particular, we combine the fast feature fool loss [MGR19], however, focusing on the first layer only, with the cross-entropy loss, to train an attack generator with the help of a source model for generating targeted and non-targeted UAPs for any other target model.

Third, we show that our UAPs not only exhibit remarkable transferability across multiple networks trained on the same dataset, but also they can be generated using a

reduced subset of the training dataset, while still having a satisfactory generalization power over unseen data.

Fourth, using our method, we are able to surpass state-of-the-art performance in both white-box and black-box settings.

Finally, by extensive evaluation of the generated UAPs on various image classification and semantic segmentation models, we demonstrate that our approach is generalizable across multiple vision tasks.

The remainder of this chapter is organized as follows. In Sect. 2, we present some background and related works. The proposed method, along with mathematical notations, is introduced in Sect. 3. Experimental results on image classification and semantic segmentation tasks are then presented in Sects. 4 and 5, respectively. Finally, in Sect. 6 we conclude the chapter.

2 Related Works

2.1 Environment Perception for Automated Vehicles

Figure 1 gives an overview of the major components comprised within an automated vehicle: Environment perception, motion planning, and motion control [GR20]. The environment perception module collects data using several sensors to obtain an overall representation of the surroundings. This information is then processed through the motion planning module to calculate a reasonable trajectory, which is executed via the motion control module at the end. In this chapter, we concentrate on the environment perception of automated vehicles.

The environment perception of automated vehicles comprises several sensors, including radio detection and ranging (RaDAR), light detection and ranging (LiDAR),

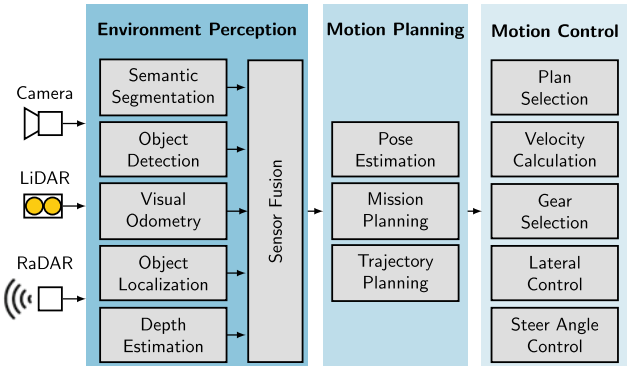


Fig. 1 Environment perception and subsequent functions in an automated vehicle (AV), acc. to [GR20]

and camera [RT19]. RaDAR sensors supplement camera vision in times of low visibility, e.g., night driving, and are able to improve the detection accuracy [PTWA17]. LiDAR sensors are commonly used to make high-resolution maps, and they are capable of detecting obstacles [WXZ20]. The recent rise of deep neural networks puts a high interest to camera-based environment perception solutions, e.g., object detection and classification [ZWJW20], as well as semantic segmentation [RABA18].

In this chapter, we focus on camera-based environment perception with deep neural networks for image classification as well as for semantic segmentation in the context of AVs.

2.2 Adversarial Attacks

It is well known that deep neural networks are vulnerable to adversarial attacks [SZS+14]. While adversarial attacks for image classification have been widely studied, the influence of these attacks in other applications, such as semantic segmentation, has rarely been investigated [AM18, BHSFs19, BZIK20, BKV+20, KBFs20, ZBIK20a, BLK+21]. In this section, we review existing attack methods in both classification and semantic segmentation perception tasks.

There are several possible ways of categorizing an adversarial attack, e.g., targeted/non-targeted attack, and white-box/black-box attack [HM19]. In a targeted attack, the adversary generates an adversarial example to change the system's output to a maliciously chosen target label. In a non-targeted attack, on the other hand, the attacker wants to direct the result to a label that is different from the ground truth, no matter what it is. It should be noted that in semantic segmentation, targeted attacks can be divided into static and dynamic target segmentation [MCKBF17]. Static attacks are similar to the targeted attacks in image classification. Here, the aim is to enforce the model to always output the same target semantic segmentation mask. The dynamic attack follows the objective of replacing all pixels classified with a beforehand chosen target class by its spatial nearest neighbor class. Another way of attack categorization is based on the attackers' knowledge of the parameters and the architecture of the target model. While a white-box scenario refers to the case when an attacker has full knowledge about the underlying model, a black-box scenario implies that no information about the respective model is available. We address both targeted and non-targeted attacks as well as white-box and black-box scenarios in this chapter.

In the following, we will investigate two types of image-dependent and image-agnostic adversarial perturbations in classification and semantic segmentation models. We will also review the problem of transferability in adversarial attacks.

Image-dependent adversarial perturbations: There are various iterative, non-iterative, and generative methods for crafting image-dependent adversarial examples. Goodfellow et al. [GSS15] introduced the fast gradient sign method (FGSM), one of the first adversarial attacks. FGSM aims at computing the gradients of the

source model S with respect to the (image) input $\mathbf{x}$ and a loss J to create an adversarial example. Iterative FGSM (I-FGSM) [KGB17] iteratively applies FGSM with a small step size, while momentum iterative FGSM [DLP+18] utilizes a momentum-based optimization algorithm for stronger adversarial attacks. Another iterative attack is projected gradient descent (PGD) [MMS+18], where the main difference to I-FGSM is random restarts in the optimization process. Kurakin et al. [KGB17] proposed the least-likely class method (LLCM), in which the target is set to the least-likely class predicted by the network. Xiao et al. [XZL+18] introduced the spatial transform attack (STM) for generating adversarial examples. In STM, instead of changing the pixel values in a direct manner, spatial filters are employed to substitute pixel values maliciously. Also, some works [HM19] have employed a generative adversarial network (GAN) [GPAM+14] for generating adversarial examples.

A basic analysis on the behavior of semantic segmentation DNNs against adversarial examples was performed by Arnab et al. [AMT18]. They applied three commonly known adversarial attacks for image classification tasks, i.e., FGSM [GSS15], Iterative-FGSM [KGB17], and LLCM [KGB17], to semantic segmentation models and illustrated the vulnerability of this task. There are also some works that concentrate on sophisticated adversarial attacks that lead to more reasonable outcomes in the context of semantic segmentation [XDL+18, PKGB18].

Universal adversarial perturbations: Universal adversarial perturbations (UAPs) were firstly introduced by Moosavi-Dezfooli et al. as image-agnostic perturbations [MDFFF17]. Similar to image-dependent adversarial attacks, there are some iterative, non-iterative, and generative techniques for creating UAPs. An iterative algorithm to generate UAPs for an image classifier was presented in [MDFF+18]. The authors provided an analytical analysis of the decision boundary in DNNs based on geometry and proved the existence of small universal adversarial perturbations. Some researchers focused on generative models that can be trained for generating UAPs [HD18, RMOGVB18]. Mopuri et al. presented a network for adversary generation (NAG) [RMOGVB18], which builds upon GANs. NAG utilizes fooling and diversity loss functions to model the distribution of UAPs for a DNN image classifier. Moreover, Poursaeed et al. [PKGB18] introduced the generative adversarial perturbation (GAP) algorithm for transforming noise drawn from a uniform distribution to adversarial perturbations to conduct adversarial attacks in classification and semantic segmentation tasks. Metzen et al. [MCKBF17] proposed an iterative algorithm for semantic segmentation, which led to more realistically looking false segmentation masks.

Contrary to the previous *data-dependent* methods, Mopuri et al. [MGR19] introduced fast feature fool (FFF), a *data-independent* algorithm for producing non-targeted UAPs. FFF aims at injecting maximal adversarial energy into each layer of the source model S. This is done by the following loss function:

$$J^{\mathrm{FFF}}(\mathbf{r}) = \sum_{\ell=1}^{L} J_{\ell}^{\mathrm{FFF}}(\mathbf{r}), \quad J_{\ell}^{\mathrm{FFF}}(\mathbf{r}) = -\log(\|\mathbf{A}_{\ell}(\mathbf{r})\|_2), \tag{1}$$

where $\mathbf{A}_\ell(\mathbf{r})$ is the mean of all feature maps of the ℓ-th layer (after the activation function in layer ℓ), when *only* the UAP $\mathbf{r}$ is fed into the model. Note that usually $\mathbf{x}^{\text{adv}} = \mathbf{x} + \mathbf{r}$, with clean image $\mathbf{x}$, is fed into the model. This algorithm starts with a random $\mathbf{r}$ which is then iteratively optimized. For mitigating the absence of data in producing UAPs, Mopuri et al. [MUR18] proposed class impressions (CIs), a form of reconstructed images that are obtained via simple optimization from the source model. After finding multiple CIs in the input space for each target class, they trained a generator to create UAPs. Recently, Zhang et al. [ZBIK20b] proposed a targeted UAP algorithm using random source images (TUAP-RSI) from a proxy dataset instead of the original training dataset.

In this chapter, we follow Poursaeed et al. [PKGB18] by proposing an efficient generative approach that focuses on propagating adversarial energy in a source model to generate UAPs for the task of image classification and semantic segmentation.

Transferability in black-box attacks: The ability of an adversarial example to be effective against a different, potentially unknown, target model is known as transferability. Researchers have evaluated the transferability of adversarial examples on image classifiers [MGR19, MDFFF17, PXL+20, LBX+20] and semantic segmentation networks [PKGB18, AMT18].

Regarding the philosophy behind transferability, Goodfellow et al. [GSS15] demonstrated that estimating the size of adversarial subspaces is relevant to the transferability issue. Another potential perspective lies in the similarity of decision boundaries. Learning substitute models, approximating the decision boundaries of target models, is a famous approach to attack an unknown model [PMJ+16]. Wu et al. [WWX+20] considered DNNs with skip connections and found that using more gradients from the skip connections, rather than the residual modules, allows the attacker to craft more transferable adversarial examples. Wei et al. [WLCC18] proposed to manipulate feature maps, extracted by a separate feature network, to create more transferable image-dependent perturbations using a GAN. Li et al. [LBZ+20] introduced a virtual model known as a ghost network to apply feature-level perturbations to an existing model to produce a large set of diverse models. They showed that ghost networks, together with a coupled ensemble strategy, improve the transferability of existing techniques. Wu et al. [WZTE18] empirically investigated the dependence of adversarial transferability on model-specific attributes, including model capacity, architecture, and test accuracy. They demonstrated that fooling rates heavily depend on the similarity of the source model and target model architectures.

In this chapter, we increase the tranferability of generated UAPs by including a loss term inspired by Mopuri et al. [MGR19] focusing on the adversarial energy in early layers.

3 Method

3.1 Mathematical Notations

We first introduce notations for image classification and semantic segmentation in a natural domain without attacks and then extend this to the adversarial domain.

Natural Domain: Consider a *classifier* S trained on a training set $\mathcal{X}^{\text{train}}$ having M different classes. This network assigns a label $m = S(\mathbf{x}) \in \mathcal{M} = \{1, ..., M\}$ to each input image $\mathbf{x}$ from training set $\mathcal{X}^{\text{train}}$ or test set $\mathcal{X}^{\text{test}}$. We assume that image $\mathbf{x} \in \mathbb{I}^{H \times W \times C}$ is a clean image, meaning that it contains no adversarial perturbations, with height H, width W, $C = 3$ color maps, and $\mathbb{I} = [0, 1]$. Each image $\mathbf{x}$ is tagged with a ground truth label $\overline{m} \in \mathcal{M}$.

In *semantic segmentation*, given an input image $\mathbf{x}$, we assign each pixel with a class label. In this case, the semantic segmentation network S outputs a label map $\mathbf{m} = S(\mathbf{x}) \in \mathcal{M}^I$ for each input image $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_I)$, with $I = H \times W$. Similar to before, each pixel $\mathbf{x}_i \in \mathbb{I}^C$ of an image $\mathbf{x}$ is tagged with a ground truth label $\overline{m}_i \in \mathcal{M}$, resulting in the ground truth label map $\overline{\mathbf{m}}$ for the entire image $\mathbf{x}$.

Adversarial domain: Let $\mathbf{x}^{\text{adv}}$ be an adversarial example that belongs to the adversarial space of the network; for example, for the network S this space is defined as $\mathcal{X}_S^{\text{adv}}$. In order to have a quasi-imperceptible perturbation $\mathbf{r}$ when added to clean images to obtain adversarial examples, i.e., $\mathbf{x}^{\text{adv}} = \mathbf{x} + \mathbf{r}$, it is bounded by $\|\mathbf{r}\|_p \leq \epsilon$, with ϵ being the supremum of a respective p-norm $\|\cdot\|_p$.

In case of *classification* networks, for each $\mathbf{x}^{\text{adv}} \in \mathcal{X}_S^{\text{adv}}$, the purpose of non-targeted attacks is to obtain $S(\mathbf{x}^{\text{adv}}) \neq \overline{m}$. In targeted attacks, the attacker tries to enforce $S(\mathbf{x}^{\text{adv}}) = \mathring{m} \neq \overline{m}$, where $\mathring{m}$ denotes the target class the attacker aims at.

In case of *semantic segmentation* networks, for each $\mathbf{x}^{\text{adv}} \in \mathcal{X}_S^{\text{adv}}$, in non-targeted attacks we aim at $S(\mathbf{x}^{\text{adv}}) = \mathbf{m} = (m_i)$ with $m_i \neq \overline{m}_i$ for all $i \in \mathcal{I} = \{1, ..., I\}$. On the other hand, in static targeted attacks, the goal is $S(\mathbf{x}^{\text{adv}}) = \mathring{\mathbf{m}} \neq \overline{\mathbf{m}}$, where $\mathring{\mathbf{m}}$ denotes the target mask of the attacker.

Also, let T be the target model under attack, which is a deep neural network (DNN) with given (frozen) parameters, pretrained on some datasets to perform a specific environment perception task. We define $\mathbf{z}$ as a random variable sampled from a distribution, which is fed to a UAP generator $\mathbf{G}$ to produce a perturbation $\mathbf{r} = \mathbf{G}(\mathbf{z})$. Also, J stands for a loss function, which is differentiable with respect to the network parameters and the input.

3.2 Method Introduction and Initial Analysis

In order to improve the robustness of DNNs for environment perception, knowledge of sophisticated image-agnostic adversarial attacks is needed, capable of working in both white-box and black-box fashions.

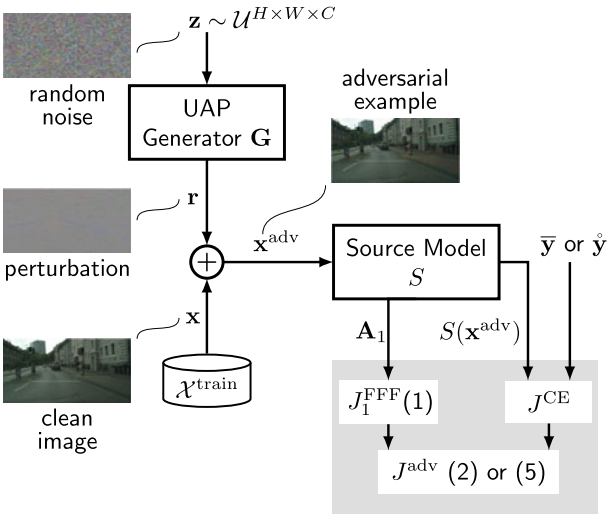


Fig. 2 Proposed training approach for a UAP generator for non-targeted attacks (label $\overline{\mathbf{y}}$) and targeted attacks ($\mathring{\mathbf{y}}$)

In this section, we present our proposed approach to generate UAPs. It builds upon the adversarial perturbation generator proposed by Poursaeed et al. [PKGB18]. Unlike [PKGB18], we focus on a fooling loss function for generating effective UAPs in both white-box and black-box scenarios. We employ a pretrained DNN as the source model S, which is exposed to UAPs during training the UAP generator. Our goal is to find a UAP $\mathbf{r}$ by an appropriate loss function, which is able to not only deceive the source model S on a training or test *dataset* $\mathcal{X}^{\text{train}}$ or $\mathcal{X}^{\text{test}}$, respectively, but also to effectively deceive a target *model* T, for which $T \neq S$ holds.

Figure 2 gives an overview of our UAP generator training methodology. Let $\mathbf{G}(\mathbf{z}) = \mathbf{r}$ be the UAP generator function mapping an unstructured, random multi-dimensional input $\mathbf{z} \sim \mathcal{U}^{H \times W \times C}$ sampled from a prior uniform distribution $\mathcal{U} = \mathbb{I}$, onto a perturbation $\mathbf{r} \in \mathbb{I}^{H \times W \times C}$. To obtain a p-norm scaled $\mathbf{r}$, a preliminary obtained perturbation $\mathbf{r}'$ is bounded by multiplying the generator network raw output $\mathbf{r}' = \mathbf{G}'(\mathbf{z})$ with $\min(1, \frac{\epsilon}{\|\mathbf{G}'(\mathbf{z})\|_p})$. Next, the resulting adversarial perturbation $\mathbf{r}$ is added to an image $\mathbf{x} \in \mathcal{X}^{\text{train}}$ before being clipped to a valid range of RGB image pixel values, resulting in an adversarial example $\mathbf{x}^{\text{adv}}$. Finally, the generated adversarial example $\mathbf{x}^{\text{adv}}$ is fed to a pretrained source model S to compute the adversarial loss functions based on targeted or non-targeted attack types.

To increase the model transferability of the generated UAPs, we seek similarities between different pretrained DNNs to take advantage of this property. For this, we selected some state-of-the-art classifiers such as `VGG-16`, `VGG-19` [SZ15], `ResNet-18`, and `ResNet-152` [HZRS16], all pretrained on ImageNet [RDS+15], to investigate their extracted feature maps in different levels. Then, we first measure the similarity of the mean feature maps of a layer between all networks over the entire

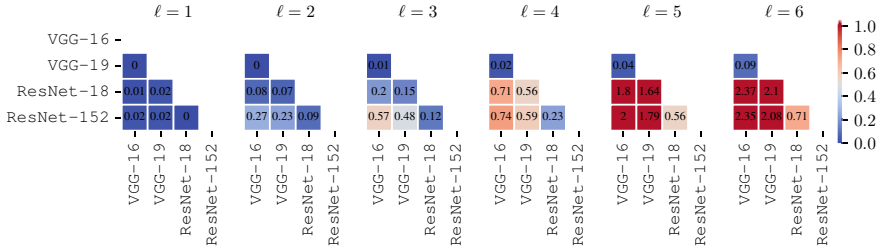


Fig. 3 Mean squared error (MSE) $\|\mathbf{A}_\ell^{\text{net1}}(\mathbf{x}) - \mathbf{A}_\ell^{\text{net2}}(\mathbf{x})\|_2^2$ between the mean of feature representations $\mathbf{A}_\ell(\mathbf{x})$ in layers $\ell \in \{1, 2, 3, 4, 5, 6\}$ of different DNN classifiers pretrained on the ImageNet dataset. The results are reported for images $\mathbf{x}$ from the ImageNet validation dataset. All networks show a considerable similarity in terms of MSE in the first layer, whereas similarity in later layers is only seen among `VGG-16` and `VGG-19` [SZ15]

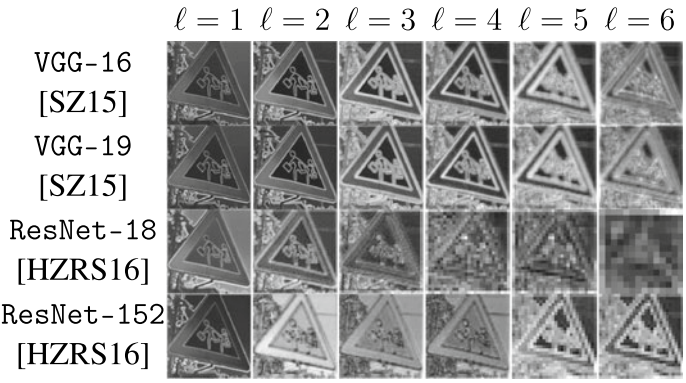


Fig. 4 The layer-wise mean of feature representations $\mathbf{A}_\ell(\mathbf{x})$ within different pretrained classifiers, computed for the first six layers for a random input image $\mathbf{x}$ taken from the ImageNet validation dataset. High similarity is observed in the first layers, in the later layers only among the `VGG-16` and the `VGG-19` network

ImageNet [RDS+15] validation set, using the well-known and universally applicable mean squared error (MSE).[1] Figure 3 displays the resulting heat maps. In addition, Fig. 4 shows the mean of feature representations $\mathbf{A}_\ell(\mathbf{x})$ for these four pretrained classifiers computed for layer $\ell = 1$ up to $\ell = 6$ (after each activation function) for a selected input image $\mathbf{x}$. Both figures, Figs. 3 and 4, show that the respective networks share a qualitatively and quantitatively high similarity in the first layer compared to all subsequent layers. Only for close relatives, such as `VGG-16` and `VGG-19`, this similarity is found in later layers as well. We thus hypothesize that by applying the fast feature fool loss J_1^{FFF} (1) *only to the first layer* of the source model during train-

[1] We also evaluated the similarity of feature maps in different layers by the structural similarity index (SSIM) [WBSS04] and peak signal to noise ratio (PSNR) [HZ10]. The results are very similar to Fig. 3.

ing, we not only inject high adversarial energy into the first layer but also increase the transferability of the generated UAPs.

In the following, we formulate our fooling loss and consider the non-targeted and targeted attack cases (see Fig. 2).

3.3 Non-targeted Perturbations

In the non-targeted case, we want to fool the source model S so that its prediction $S(\mathbf{x}^{\text{adv}})$ differs from the ground truth one-hot representation $\overline{\mathbf{y}}$. In the simplest and most sensible possible way, researchers define the negative cross-entropy as the fooling loss for non-targeted attacks, while Poursaeed et al. [PKGB18] proposed the logarithm of this loss function.

For *image classification*, we define the generator fooling loss for our non-targeted attacks as

$$J^{\text{adv,nontargeted}} = -\alpha \cdot J^{\text{CE}}(S(\mathbf{x}^{\text{adv}}), \overline{\mathbf{y}}) + (1-\alpha) \cdot J_1^{\text{FFF}}(\mathbf{x}^{\text{adv}}), \tag{2}$$

where J^{CE} denotes the cross-entropy loss, $\overline{\mathbf{y}} = (\overline{y}_\mu) \in \{0, 1\}^M$ is the one-hot encoding of the ground truth label $\overline{m}$ for image $\mathbf{x}$, and μ being the class index, such that $\overline{m} = \arg\max_{\mu \in \mathcal{M}} \overline{y}_\mu$. Also, let $J_1^{\text{FFF}}(\mathbf{x}^{\text{adv}})$ be the fast feature fool loss of layer $\ell = 1$ (see (1)), when $\mathbf{x}^{\text{adv}}$ is fed to the network S. Further, the cross-entropy loss is defined as

$$J^{\text{CE}}(\mathbf{y}, \overline{\mathbf{y}}) = -\sum_{\mu \in \mathcal{M}} \overline{y}_\mu \log\left(y_\mu\right) = -\log\left(y_{\overline{m}}\right), \tag{3}$$

where $\mathbf{y} = S(\mathbf{x}^{\text{adv}}) = (y_\mu) \in \mathbb{I}^M$ is the network output vector of S with the predictions for each class μ. For optimization, we utilize Adam [KB15] in standard configuration, following [PKGB18].

For *semantic segmentation*, in (2) $\mathbf{y} = S(\mathbf{x}^{\text{adv}}) \in \mathbb{I}^{H \times W \times M}$ and $\overline{\mathbf{y}} \in \{0, 1\}^{H \times W \times M}$ holds, and the cross-entropy loss in (3) is changed to

$$\begin{aligned} J^{\text{CE}}(\mathbf{y}, \overline{\mathbf{y}}) &= -\sum_{i \in \mathcal{I}} \sum_{\mu \in \mathcal{M}} \overline{y}_{i,\mu} \log\left(y_{i,\mu}\right) \\ &= -\sum_{i \in \mathcal{I}} \log\left(y_{i,\overline{m}_i}\right), \end{aligned} \tag{4}$$

with $y_{i,\mu}$, $\overline{y}_{i,\mu}$ being the prediction and ground truth for class μ at pixel i, respectively, and $y_{i,\overline{m}_i}$ being the prediction at pixel i for the ground truth class $\overline{m}_i$ of pixel i.

3.4 Targeted Perturbations

Different from the non-targeted case, the goal of a targeted one is to let the DNN output $S(\mathbf{x}^{\text{adv}})$ take on values $\mathring{\mathbf{y}}$ defined by the attacker, which usually differ from the ground truth (if source target labels align with the ground truth of a particular image, the UAP will output that ground truth label). Hence, the attacker aims to decrease the cross-entropy loss with respect to a target until the source model S predicts the selected target class with high confidence. As before, we add the fast feature fool loss in the first layer to boost the transferability of the targeted generated UAP.

For *image classification*, our generator fooling loss for targeted attacks is

$$J^{\text{adv,targeted}} = \alpha \cdot J^{\text{CE}}(S(\mathbf{x}^{\text{adv}}), \mathring{\mathbf{y}}) + (1-\alpha) \cdot J_1^{\text{FFF}}(\mathbf{x}^{\text{adv}}), \tag{5}$$

where $\mathring{\mathbf{y}} \in \{0,1\}^M$ is the one-hot encoding of the target label $\mathring{m} \neq \overline{m}$. Note that the sign of the cross entropy is flipped compared to the non-targeted case (2). Then, similar to the non-targeted attack, the Adam optimizer is utilized.

If we consider *semantic segmentation*, the ground truth $\mathring{\mathbf{y}}$ in (5) becomes a one-hot encoded semantic segmentation mask $\mathring{\mathbf{y}} \in \{0,1\}^{H \times W \times M}$.

4 Experiments on Image Classification

In this section, we will first describe the dataset, network architectures, and evaluation metrics, which are used to measure the performance of generated UAPs on *image classifiers*. Afterward, we will analyze the effectiveness of the proposed fooling method on state-of-the-art image classifiers and will compare it to other state-of-the-art attack methods.

4.1 Experimental Setup

Dataset: We use the ImageNet dataset [RDS+15], which is a large collection of human-annotated images. For all our experiments, a universal adversarial perturbation is computed for a set of 10,000 images taken from the ImageNet training set $\mathcal{X}^{\text{train}}$ (i.e., 10 images per class) and the results are reported on the ImageNet validation set $\mathcal{X}^{\text{val}}$ (50,000 images).

Network architectures: There are several design options regarding the architecture choices for generator $\mathbf{G}$ and source model S. For our generator, we follow [ZPIE17] and [PKGB18] and choose the `ResNet` generator from [JAFF16], which consists of some convolution layers for downsampling, followed by some residual blocks, before performing upsampling using transposed convolutions. As topology for the source model S, we utilize the same *set* of pretrained image classifiers as for the target

Table 1 Fooling rates (%) of our proposed non-targeted UAPs in white-box attacks, for different values of α and various target classifiers pretrained on ImageNet. Results are reported on a second training set of 10,000 images. The adversarial perturbation is bounded by $L_\infty(\mathbf{r}) \leq \epsilon = 10$. The highest fooling rates (%) are printed in boldface

α	Source Model S = Target Model T				Avg
	VGG-16	VGG-19	ResNet-18	ResNet-152	
0	8.52	8.29	7.24	4.04	7.02
0.6	90.49	93.48	88.93	84.41	89.32
0.7	**95.20**	**93.79**	**89.16**	87.05	**91.30**
0.8	90.03	93.24	89.07	**89.91**	90.56
0.9	95.13	92.14	88.34	89.37	91.24
1	92.87	71.88	88.88	85.34	84.74

Table 2 Fooling rates (%) of our proposed non-targeted UAPs in white-box attacks, for various target classifiers pretrained on ImageNet. Results are reported on the ImageNet validation set. We report results for two L_p norms, namely $L_2(\mathbf{r}) \leq \epsilon = 2000$ and $L_\infty(\mathbf{r}) \leq \epsilon = 10$

p	ϵ	α	Source Model S = Target Model T			
			VGG-16	VGG-19	ResNet-18	ResNet-152
2	2000	0.7	96.57	94.99	91.85	88.73
∞	10	0.7	95.70	94.00	90.46	90.40

model T, i.e., VGG-16, VGG-19 [SZ15], ResNet-18, ResNet-152 [HZRS16], and also GoogleNet [SLJ+15].

Evaluation metrics: We use the fooling rate as our metric to assess the performance of our crafted UAPs on DNN image classifiers [MDFFF17, PKGB18, MUR18, MGR19]. In the case of non-targeted attacks, it is the percentage of input images for which $T(\mathbf{x}^{\text{adv}}) \neq T(\mathbf{x})$ holds. For targeted attacks, we calculate the top-1 target accuracy, which can be understood as the percentage of adversarial examples, that is classified "correctly" as the target class as desired by the attacker.

4.2 Non-Targeted Universal Perturbations

According to Fig. 2, we train our model with the non-targeted fooling loss (2). For tuning the hyperparameter α, the weight of our novel adversarial loss components, we utilized a second training set of 10,000 random images (again 10 images per class) taken from the ImageNet training set which is disjoint from both the training and the validation dataset. Table 1 shows that the best α for non-targeted attacks, on average over all model topologies, is $\alpha = 0.7$.

For white-box attacks, where $S = T$ holds, results on the ImageNet validation set for two different norms are given in Table 2. The maximum permissible L_p norm of

Table 3 Fooling rates (%) of various non-targeted state-of-the-art methods on various target classifiers trained on ImageNet (white-box attacks). The results of other state-of-the-art methods are reported from the respective paper. $^{+}$For comparison reasons, the average of our method leaves out the `ResNet-18` model results. Highest fooling rates are printed in boldface

p	ϵ	Method	$S = T$				Avg^{+}
			`VGG-16`	`VGG-19`	`ResNet-18`	`ResNet-152`	
∞	*10	FFF	47.10	43.62	-	29.78	40.16
		CIs	71.59	72.84	–	60.72	68.38
		UAP	78.30	77.80	–	84.00	80.03
		GAP	83.70	80.10	–	82.70	82.16
		NAG	77.57	83.78	–	87.24	82.86
		TUAP-RSI	94.30	**94.98**	–	90.08	93.12
		Ours	**95.70**	94.00	**90.46**	**90.40**	**93.36**
2	2000	UAP	90.30	84.50	–	88.50	87.76
		GAP	93.90	94.90	–	79.50	89.43
		Ours	**96.57**	**94.99**	**91.85**	**88.73**	**93.43**

(a) The UAP $\mathbf{r}$ (left) and the respective adversarial examples $\mathbf{x}^{\text{adv}} = \mathbf{x} + \mathbf{r}$ (right), with $L_2(\mathbf{r}) \leq 2000$.

(b) Original images $\mathbf{x}$.

(c) The UAP $\mathbf{r}$ (left) and the respective adversarial examples $\mathbf{x}^{\text{adv}} = \mathbf{x} + \mathbf{r}$ (right), with $L_\infty(\mathbf{r}) \leq 10$.

Fig. 5 Examples of our non-targeted UAPs and adversarial examples. In **a** the universal adversarial perturbation is given on the left and eight different adversarial examples are shown on the right, where the L_2 norm of the adversarial perturbation is bounded by $\epsilon = 2000$, i.e., $L_2(\mathbf{r}) \leq 2000$. In **b** the respective original images are shown, whereas in **c** the L_∞ norm of the adversarial perturbation is bounded by $\epsilon = 10$, i.e., $L_\infty(\mathbf{r}) \leq 10$, $\alpha = 0.7$. In these experiments, the source model S is `ResNet-18` [HZRS16]. The pixel values in UAPs are scaled for better visibility

the perturbations for $p = 2$ and $p = \infty$ is set to be $\epsilon = 2000$ and $\epsilon = 10$, respectively, following [MDFFF17]. As Moosavi-Dezfooli et al. [MDFFF17] pointed out, these values are selected to acquire a perturbation whose norm is remarkably smaller than the average image norms in the ImageNet dataset to obtain quasi-imperceptible adversarial examples. The results in Table 2 show that the proposed method is successful in the white-box setting. For the L_∞ norm, all reported fooling rate numbers

Table 4 Transferability of our proposed non-targeted UAPs (white-box and black-box attacks). Results are reported in the form of fooling rates (%) for various combinations of source models S and target models T, pretrained on ImageNet. The generator is trained to fool the source model (rows), and it is tested on the target model (columns). The adversarial perturbation is bounded by $L_\infty(\mathbf{r}) \leq \epsilon = 10$, $\alpha = 0.7$. *The average is computed *excluding* the easier white-box attacks (main diagonal)

		Target model T				Avg*
		VGG-16	VGG-19	ResNet-18	ResNet-152	
Source model S	VGG-16	**95.70**	86.67	49.98	36.34	57.66
	VGG-19	84.77	**94.00**	47.24	36.46	56.15
	ResNet-18	76.49	72.18	**90.46**	50.46	66.37
	ResNet-152	86.19	82.36	76.04	**90.40**	**81.53**

are above 90%. To illustrate that our adversarial examples are quasi-imperceptible to humans, we illustrate some adversarial examples with their respective scaled UAPs in Fig. 5.

In Table 3, we compare our proposed approach in generating non-targeted UAPs with state-of-the-art methods, i.e., fast feature fool (FFF, as originally proposed, on all layers) [MGR19], class impressions (CIs) [MUR18], universal adversarial perturbation (UAP) [MDFFF17], generative adversarial perturbation (GAP) [PKGB18], network for adversary generation (NAG) [RMOGVB18], and targeted UAP-random source image (TUAP-RSI) [ZBIK20b]. In these experiments, we again consider the white-box setting, i.e., $S = T$. Our proposed approach achieves a new state-of-the-art performance on *almost all* models on *both* L_p norms, being on average 4% absolute better in fooling rate with the L_2 norm, and at least 0.26% absolute better with the L_∞ norm.

In Table 4 we investigate the white-box ($S = T$) and black-box ($S \neq T$) capability of our proposed method through various combinations of source and target models. Note that the rightmost column represents an average over the fooling rates in the black-box settings and thus indicates the transferability of our proposed method. Overall, in the white-box and black-box settings, we achieve fooling rates of over 90% and 55%, respectively. We also compare the transferability of our produced UAPs with the same state-of-the-art methods as before. The results for these experiments are shown in Table 5, where VGG-16, ResNet-152, and GoogleNet are used as the source model in Table 5a–c, respectively. It turns out to be advisable to choose a deep network as the source model (ResNet-152); since then our performance on the unseen VGG-16 and VGG-19 target models is about 12% absolute better than earlier state of the art (L_∞ norm).

For investigating the generalization power of UAPs w.r.t. unseen data, we evaluate the influence of the size of the training dataset $\mathcal{X}^{\text{train}}$ on the quality of UAPs in conducting white-box and black-box attacks. Figure 6 shows the fooling rates obtained for VGG-16 as the target model T, on the ImageNet validation set for different sizes

Table 5 Transferability of our proposed non-targeted UAPs compared to other methods, i.e., FFF [MGR19], CIs [MUR18], UAP [MDFFF17], GAP [PKGB18], and NAG [RMOGVB18], using different source models S and target models T. The UAP is bounded by $L_\infty(\mathbf{r}) \leq \epsilon = 10$. Values of our method are taken from Table 4 (except `GoogleNet`). *Note that the results are reported from the respective paper. Highest fooling rates are printed in boldface

(a) S: `VGG-16` [SZ15]		
T	Method	Fooling Rate (%)
`VGG-19`	FFF*	41.98
	CIs*	65.64
	UAP*	73.10
	GAP	79.14
	NAG*	73.25
	Ours	**86.67**
`ResNet-152`	FFF*	27.82
	CIs*	45.33
	UAP*	**63.40**
	GAP	30.32
	NAG*	54.38
	Ours	36.34
`ResNet-18`	**Ours**	**49.98**
(b) S: `ResNet-152` [HZRS16]		
T	Method	Fooling Rate (%)
`VGG-16`	FFF*	19.23
	CIs*	47.21
	UAP*	47.00
	GAP	70.45
	NAG*	52.17
	Ours	**86.19**
`VGG-19`	FFF*	17.15
	CIs*	48.78
	UAP*	45.50
	GAP	70.38
	NAG*	53.18
	Ours	**82.36**
`ResNet-18`	**Ours**	**76.04**
(c) S: `GoogleNet` [SLJ+15]		
T	Method	Fooling Rate (%)
`VGG-16`	FFF*	40.91
	CIs*	59.12
	UAP*	39.20
	GAP	71.14
	NAG*	56.40
	Ours	**75.35**
`ResNet152`	FFF*	25.31
	CIs*	47.81
	UAP*	45.50
	GAP	51.72
	NAG*	59.22
	Ours	**61.24**
`ResNet-18`	**Ours**	**67.30**

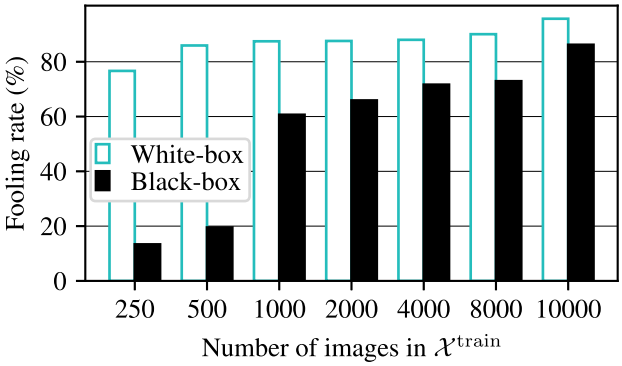


Fig. 6 Fooling rates (%) of our non-targeted UAPs on the ImageNet validation set for different sizes of $\mathcal{X}^{\text{train}}$ in both white-box and black-box settings. In the white-box setting, the source model S and the target model T are `VGG-16`, while in the black-box setting, the source model S is `ResNet-152` and the target model T is again `VGG-16`. Results are reported for $L_\infty(\mathbf{r}) \leq \epsilon = 10$ and $\alpha = 0.7$

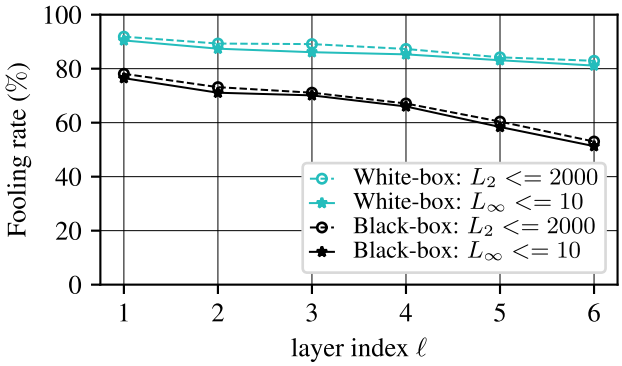


Fig. 7 Fooling rates (%) of our non-targeted UAPs on the ImageNet validation set, in both white-box and black-box attacks for L_∞ and L_2, for different layers ℓ in (1) applied in the loss function (2). In the white-box setting, the source model S and the target model T are `ResNet-18`. In the black-box setting, the source model S is again `ResNet-18` and the target model T is `VGG-16`. Results are reported for $\alpha = 0.7$

of $\mathcal{X}^{\text{train}}$. The results show that by using a dataset $\mathcal{X}^{\text{train}}$ containing only 1000 images, our approach leads to a fooling rate of more than 60% on the ImageNet validation dataset in both the white-box and black-box settings. Additionally, the number of images in $\mathcal{X}^{\text{train}}$ turns out to be more vital for the fooling rate of black-box attacks as compared to white-box attacks.

To examine the impact of the layer which is used in our loss function, we utilized different layers in (1), then applied them in the loss function (2) to train the source model for generating UAPs. In practice, we are interested in the impact the layer position has. Figure 7 shows the fooling rate in white-box and black-box settings for L_∞ and L_2, when different layers from $\ell = 1$ to $\ell = 6$ are applied in (1) and

(2). This figure shows that choosing deeper layers leads to a decreasing trend in the fooling rate. Also, this trend is stronger in black-box attacks, where a more than 20% drop in the attack success rate can be observed. This indicates that it is advisable to choose earlier layers, in particular the first layer, in generating UAPs to obtain both an optimal fooling rate and transferability.

4.3 Targeted Universal Perturbations

In this section, we apply the targeted fooling loss (5), again with $\alpha = 0.7$, for training the generator in Fig. 2. We assume the chosen α is appropriate for targeted attacks as well and thus dispense further investigations. If results are good, then this supports some robustness w.r.t. the choice of α. Figure 8 depicts two examples of our targeted UAPs, some original images and respective adversarial examples. In these experiments, the fooling rate (top-1 target accuracy) on the validation set for the target class $\mathring{m} = 919$ (`street sign`) and $\mathring{m} = 920$ (`traffic light, traffic signal, stoplight`) are 63.2 and 57.83%, respectively, which underlines the effectiveness of our approach.

For assessing the generalization power of our proposed method across different target classes and comparison with GAP [PKGB18], we followed Poursaeed et al. and used 10 randomly sampled classes. The resulting average top-1 target accuracy, when the adversarial perturbation is bounded by $L_\infty(\mathbf{r}) \leq \epsilon = 10$, is 66.57%, which is significantly higher than the one reported for GAP [PKGB18] with 52.0%.

(a) The UAP $\mathbf{r}$ for target label `traffic light, signal, stoplight` (left) and the respective adversarial examples $\mathbf{x}^{\text{adv}}$ (right).

(b) Original images $\mathbf{x}$.

(c) The UAP $\mathbf{r}$ for target label `street sign` (left) and the respective adversarial examples $\mathbf{x}^{\text{adv}}$ (right).

Fig. 8 Examples of our targeted UAPs and adversarial examples. In these experiments, the source model S is `VGG-16` [SZ15], with $L_\infty(\mathbf{r}) \leq \epsilon = 10$, $\alpha = 0.7$. The pixel values in UAPs are scaled for better visibility

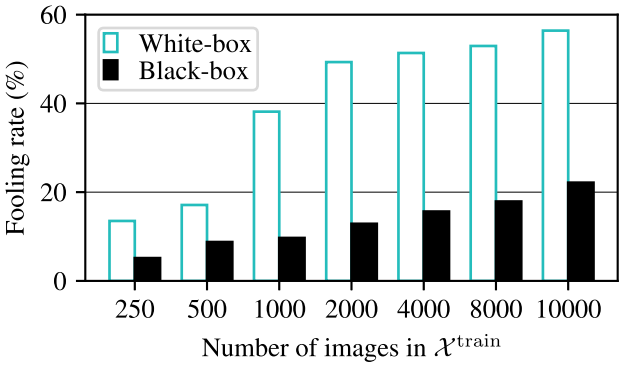


Fig. 9 Fooling rates (%) of our targeted UAPs on the ImageNet validation set for different sizes of $\mathcal{X}^{\text{train}}$ in both white-box and black-box attacks. In the white-box setting, the source model S and the target model T are `VGG-16`, while in the black-box setting, the source model S is again `VGG-16` and the target model T is `VGG-19`. Results are reported for $L_\infty(\mathbf{r}) \leq \epsilon = 10$, $\alpha = 0.7$, and target class $\mathring{m} = 847$ (`tank, army tank, armored combat vehicle, armoured combat vehicle`)

To demonstrate the generalization power of our targeted UAPs w.r.t. unseen data, we visualize the attack success rate obtained by the source model `VGG-16`, in white-box and black-box settings, on the Image-Net validation dataset for different sizes of $\mathcal{X}^{\text{train}}$ in Fig. 9. For instance, with $\mathcal{X}^{\text{train}}$ containing 10,000 images, we are able to fool the target model on over 20% of the images in the ImageNet validation set. It should be noted that training the generator $\mathbf{G}$ to produce a single UAP forcing the target model to output a specific target class $\mathring{m}$ is an extremely challenging task. However, we particularly observe that utilizing 10,000 training images again seems to be sufficient for a white-box attack.

5 Experiments on Semantic Segmentation

We continue our investigations by applying our method to the task of semantic segmentation to show its cross-task applicability. We start with the experimental setup, followed by an evaluation of non-targeted and targeted attacks.

5.1 Experimental Setup

Dataset: We conduct our experiments on the widely known Cityscapes dataset [COR+16]. It contains pixel-level annotations of 5,000 high-resolution images (2,975 training, 500 validation, and 1,525 test images) being captured in urban street scenes.

Table 6 The mean intersection-over-union (mIoU) (%) of non-targeted UAP methods on the semantic segmentation model FCN-8 pretrained on Cityscapes. Our method is compared with GAP [PKGB18]. In these experiments, both the source model S and the target model T are either the same, i.e., $S = T =$ FCN-8, or different, i.e., $S =$ ERFNet, $T =$ FCN-8. Parameters are set to $L_\infty(\mathbf{r}) \leq \epsilon$, $\alpha = 0.7$. Best results are printed in boldface

(a) $S = T =$ FCN-8				
Method	ϵ			
	2	5	10	20
GAP	18.1	12.8	4.0	2.1
Ours	**16.2**	**9.8**	**2.0**	**0.3**
(b) $S =$ ERFNet, $T =$ FCN-8				
Method	ϵ			
	2	5	10	20
GAP	27.3	16.4	7.0	4.3
Ours	**26.4**	**14.8**	**4.1**	**1.5**

The original images have a resolution of 2048 × 1024 pixels, that we downsample to 1024 × 512 for our experiments [MCKBF17, PKGB18].

Network architecture: Regarding the architecture of the source model S, we use FCN-8 [LSD15] for white-box attacks, and ERFNet [RABA18] for the black-box setting. FCN-8 consists of an encoder part which transforms an input image into a low-resolution semantic representation and a decoder part which recovers the high spatial resolution of the image by fusing different levels of feature representations together. ERFNet also consists of an encoder-decoder structure, but without any bypass connections between the encoder and the decoder. Additionally, residual units are used with factorized convolutions to obtain a more efficient computation.

In our experiments, we consider FCN-8 [LSD15] as our segmentation target model T, and use L_∞ norm, to be comparable with [MCKBF17, PKGB18].

Evaluation metrics: To assess the performance of a semantic segmentation network, we used the mean intersection-over-union (mIoU) [COR+16]. It is defined as

$$\text{mIoU} = \frac{1}{|\mathcal{M}|} \sum_{\mu \in \mathcal{M}} \frac{\text{TP}_\mu}{\text{TP}_\mu + \text{FP}_\mu + \text{FN}_\mu}, \tag{6}$$

with class $\mu \in \mathcal{M}$, class-specific true positives TP_μ, false positives FP_μ, and false negatives FN_μ. For assessing the impact of non-targeted adversarial attacks on semantic segmentation, we compute the mIoU on adversarial examples [AMT18].

To measure the attack success of our targeted attack, we compute the pixel accuracy (PA) between the prediction $\mathbf{m} = T(\mathbf{x}^{\text{adv}})$ and the target $\mathring{\mathbf{m}}$ [PKGB18]. In this chapter, we restrict our analysis to the static target segmentation scenario [MCKBF17] and use the same target mask as in [PKGB18, MCKBF17] (see also Fig. 11).

5.2 Non-targeted Universal Perturbations

For the non-targeted case, we train our model with the non-targeted adversarial loss function (2), where we use (4) as the cross-entropy loss. The maximum permissible L_p norm of the perturbations for $p = \infty$ is set to $\epsilon \in \{2, 5, 10, 20\}$. We report results for both our method and GAP [PKGB18] on the Cityscapes validation set in Table 6a. It can be observed that across different values for ϵ our method is superior to GAP in terms of decreasing the mIoU of the underlying target model.

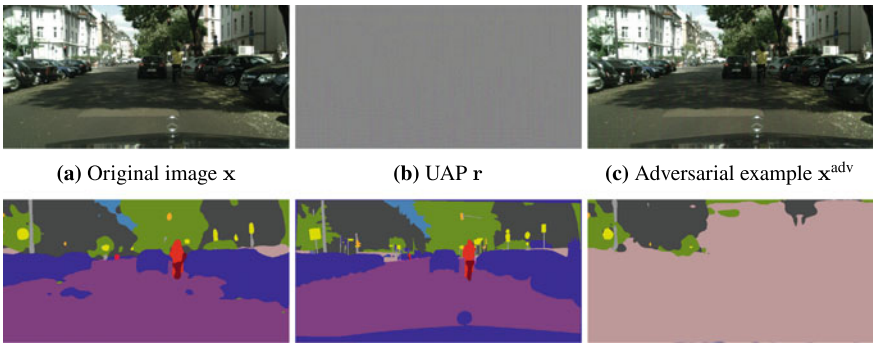
We visualize the effect of the generated non-targeted UAPs in Fig. 10 by illustrating the original image, the UAP, the resulting adversarial example, the prediction on the original image, the ground truth mask, and the resulting segmentation output. Here, the maximum permissible L_p norm of the perturbations $p = \infty$ is set to $\epsilon = 5$.

For investigating the transferability of our generated UAPs, we use a UAP optimized on `ERFNet` as the source model S and test it on the target model T being `FCN-8`. Table 6b reports black-box attack results for our method compared to the attack method GAP [PKGB18]. Our non-targeted UAPs decrease the mIoU of the `FCN-8` on the Cityscapes dataset more than GAP [PKGB18] does, in all different ranges of adversarial perturbations ($\epsilon \in \{2, 5, 10, 20\}$). These results illustrate the effectivity of the generated perturbation.

5.3 Targeted Universal Perturbations

In the targeted case, we aim at finding a UAP which forces the segmentation source model S and the segmentation target model T to predict a specific target mask $\mathring{\mathbf{m}}$. We now train our model with the targeted adversarial loss function (5), using $J^{\text{CE}}(\mathbf{y}, \mathring{\mathbf{y}})$ according to (4) for the cross-entropy loss. We apply the same target $\mathring{\mathbf{y}}$ as in [PKGB18], see $\mathring{\mathbf{m}}$ in Fig. 11e.

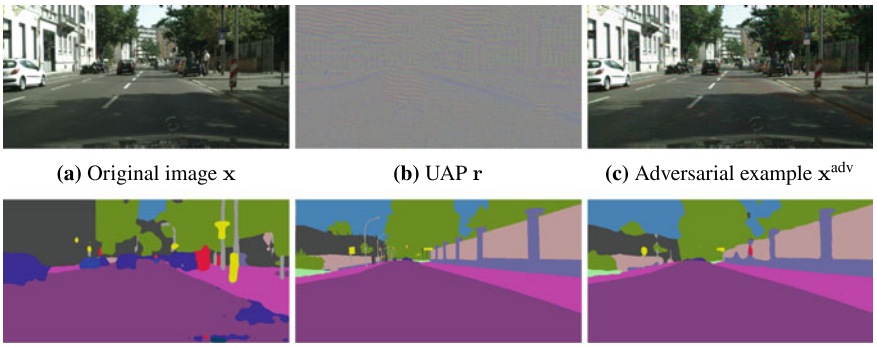
Figure 11 depicts an original image, our generated targeted UAP, the respective adversarial example, the prediction on the original image, the target segmentation mask, and the prediction on the adversarial example. The results show that the generated UAP resembles the target mask and is able to fool the `FCN-8` in a way that it now outputs the target segmentation mask.



(a) Original image $\mathbf{x}$ **(b)** UAP $\mathbf{r}$ **(c)** Adversarial example $\mathbf{x}^{\text{adv}}$

(d) Prediction for original image **(e)** Ground truth mask **(f)** Prediction for adver. example

Fig. 10 An example of our non-targeted UAPs optimized for the task of semantic segmentation on the Cityscapes dataset. Results are displayed on the Cityscapes validation set. In these experiments, both the source model S and the target model T are `FCN-8`, with $L_\infty(\mathbf{r}) \leq \epsilon = 5$, $\alpha = 0.7$. The pixel values in the UAP are scaled for better visibility



(a) Original image $\mathbf{x}$ **(b)** UAP $\mathbf{r}$ **(c)** Adversarial example $\mathbf{x}^{\text{adv}}$

(d) Prediction for original image **(e)** Target mask $\mathring{\mathbf{m}}$ **(f)** Prediction for adv. example

Fig. 11 An example of our targeted UAPs optimized for the task of semantic segmentation on the Cityscapes dataset. Results are displayed on the Cityscapes validation set. In these experiments, both the source model S and the target model T are `FCN-8`, with $L_\infty(\mathbf{r}) \leq \epsilon = 10$, $\alpha = 0.7$. The pixel values in the UAP are scaled for better visibility

We compare our targeted attack with two state-of-the-art methods in Table 7. While our method performs comparably well as state of the art for weak attacks, in medium to strong attacks we outperform both GAP [PKGB18] and UAP-Seg [MCKBF17].

Table 7 Pixel accuracy (%) of targeted UAP methods (white-box attack) on the semantic segmentation model FCN-8 pretrained on the Cityscapes training set. Results are reported on the Cityscapes validation set. Our method is compared to UAP-Seg [MCKBF17] and GAP [PKGB18]. In these experiments, both the source model S and the target model T are FCN-8, with $L_\infty(\mathbf{r}) \leq \epsilon, \alpha = 0.7$. Best results are printed in boldface

Method	ϵ			
	2	5	10	20
GAP	**61.2**	79.5	92.1	97.2
UAP-Seg	60.9	80.3	91.0	96.3
Ours	61.0	**81.8**	**93.1**	**97.4**

6 Conclusions

We presented a novel method to effectively generate targeted and non-targeted universal adversarial perturbations (UAPs) in both white-box and black-box settings. Our proposed method shows new state-of-the-art fooling rates for targeted as well as non-targeted UAPs on different classifiers. Additionally, our non-targeted UAPs show a significantly higher transferability across models when compared to other methods, given that we generated our UAPs on the deepest network in the investigation. This is achieved by incorporating an additional loss term during training, which aims at increasing the activation of the first layer. Finally, we extended our method to the task of semantic segmentation to prove its applicability also in more complex environment perception tasks. Due to its state-of-the-art effectiveness for object classification and semantic segmentation, we strongly recommend to employ the proposed types of attacks in validation of automated vehicles' environment perception.

References

[AM18] N. Akhtar, A. Mian, Threat of adversarial attacks on deep learning in computer vision: a survey. IEEE Access **6**, 14410–14430 (2018)

[AMT18] A. Arnab, O. Miksik, P.H.S. Torr, On the robustness of semantic segmentation models to adversarial attacks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 888–897

[BHSFs19] A. Bär, F. Hüger, P. Schlicht, T. Fingscheidt, On the robustness of redundant teacher-student frameworks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Long Beach, CA, USA, June 2019, pp. 1380–1388

[BKV+20] A. Bär, M. Klingner, S. Varghese, F. Hüger, P. Schlicht, T. Fingscheidt. Robust semantic segmentation by redundant networks with a layer-specific loss contribution and majority vote, in *Proceedings of the IEEE/CVF Conference on Computer*

Vision and Pattern Recognition (CVPR) Workshops, virtual conference, June 2020, pp. 1348–1358

[BLK+21] A. Bär, J. Löhdefink, N. Kapoor, S.J. Varghese, F. Hüger, P. Schlicht, T. Fingscheidt. The vulnerability of semantic segmentation networks to adversarial attacks in autonomous driving: enhancing extensive environment sensing. IEEE Signal Process. Mag. **38**(1), 42–52 (2021)

[BZIK20] P. Benz, C. Zhang, T. Imtiaz, I.-S. Kweon, Double targeted universal adversarial perturbations, in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, virtual conference, Nov. 2020, pp. 284–300

[CAB+20] A. Chaubey, N. Agrawal, K. Barnwal, K.K. Guliani, P. Mehta, Universal Adversarial Perturbations: A Survey, pp. 1–20, May 2020. arxiv:2005.08087

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 3213–3223

[DLP+18] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, Boosting adversarial attacks with momentum, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 9185–9193

[FJGN20] J. Fayyad, M.A. Jaradat, D. Gruyer, H. Najjaran, Deep learning sensor fusion for autonomous vehicle perception and localization: a review. Sensors **20**(15), 4220–4255 (2020)

[GPAM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, Dec. 2014, pp. 2672–2680

[GR20] N. Ghafoorianfar, M. Roopaei, Environmental perception in autonomous vehicles using edge level situational awareness, in *Proceedings of the IEEE Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, Jan. 2020, pp. 444–448

[GRM+19] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F.A. Wichmann, Brendel, W. ImageNet-Trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness, in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–22

[GSS15] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–11

[GTCM20] S. Grigorescu, B. Trasnea, T. Cocias, G. Macesanu, A survey of deep learning techniques for autonomous driving. J. Field Robot. **37**(3), 362–386 (2020)

[HD18] J. Hayes, G. Danezis, Learning universal adversarial perturbations with generative models, in *Proceedings of the IEEE Symposium on Security and Privacy (SP) Workshops*, San Francisco, CA, USA, May 2018, pp. 43–49

[HM19] A.S. Hashemi, S. Mozaffari, Secure deep neural networks using adversarial image generation and training with Noise-GAN. Comput. Secur. **86**, 372–387 (2019)

[HZ10] A. Hore, D. Ziou, Image quality metrics: PSNR versus SSIM, in *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR)*, Istanbul, Turkey, Aug. 2010, pp. 2366–2369

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 770–778

[JAFF16] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Amsterdam, The Netherlands, Oct. 2016, pp. 694–711

[KB15] D.P. Kingma, J. Ba, ADAM: a method for stochastic optimization, in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15

[KBFs20] M. Klingner, A. Bär, T. Fingscheidt, Improved noise and attack robustness for semantic segmentation by using multi-task training with self-supervised depth estimation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, virtual conference, June 2020, pp. 1299–1309

[KBJ+20] S. Kuutti, R. Bowden, Y. Jin, P. Barber, S. Fallah, A survey of deep learning applications to autonomous vehicle control. IEEE Trans. Intell. Transp. Syst. (TITS) **22**(2), 721–733 (2020)

[KGB17] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, in *Proceedings of the International Conference on Learning Representations (ICLR) Workshops*, Toulon, France, Apr. 2017, pp. 1–14

[LBX+20] Y. Li, S. Bai, C. Xie, Z. Liao, X. Shen, A. Yuille, Regional homogeneity: towards learning transferable universal adversarial perturbations against defenses, in *Proceedings of the European Conference on Computer Vision (ECCV)*, virtual conference, Aug. 2020, pp. 795–813

[LBZ+20] Y. Li, S. Bai, Y. Zhou, C. Xie, Z. Zhang, A.L. Yuille, Learning transferable adversarial examples via ghost networks, in *Proceedings of the AAAI Conference on Artificial Intelligence*, New York, NY, USA, Feb. 2020, pp. 11458–11465

[LSD15] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015, pp. 3431–3440

[MCKBF17] J.H. Metzen, M.C. Kumar, T. Brox, V. Fischer, Universal adversarial perturbations against semantic image segmentation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2774–2783, Venice, Italy, Oct. 2017

[MDFF+18] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, S. Soatto, Robustness of classifiers to universal perturbations: a geometric perspective, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, Apr. 2018, pp. 1–15

[MDFFF17] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, Universal adversarial perturbations, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, July 2017, pp. 1765–1773

[MGR19] K.R. Mopuri, A. Ganeshan, V.B. Radhakrishnan, Generalizable data-free objective for crafting universal adversarial perturbations. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **41**(10), 2452–2465 (2019)

[MKMW19] N. Morgulis, A. Kreines, S. Mendelowitz, Y. Weisglass, Fooling a Real Car With Adversarial Traffic Signs, June 2019, pp. 1–19. arxiv:1907.00374

[MMS+18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, Apr. 2018, pp. 1–10

[MUR18] K.R. Mopuri, P.K. Uppala, V.B. Radhakrishnan, Ask, Acquire, and Attack: Data-Free UAP generation using class impressions. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, Sept. 2018, pp. 19–34

[NVM+19] A.M. Nascimento, L.F. Vismari, C.B.S.T. Molina, P.S. Cugnasca, J.B. Camargo, J.R. de Almeida, R. Inam, E. Fersman, M.V. Marquezini, A.Y. Hata, A systematic literature review about the impact of artificial intelligence on autonomous vehicle safety. IEEE Trans. Intell. Transp. Syst. (TITS) **21**(12), 4928–4946 (2019)

[PKGB18] O. Poursaeed, I. Katsman, B. Gao, S. Belongie, Generative adversarial perturbations, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 4422–4431

[PMJ+16] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. Berkay Celik, A. Swami, The limitations of deep learning in adversarial settings, in *Proceedings of the IEEE European Symposium on Security and Privacy (ESP)*, Saarbrücken, Germany, Mar. 2016, pp. 372–387

[PTWA17] S.M. Patole, M. Torlak, D. Wang, M. Ali, Automotive radars: a review of signal processing techniques. IEEE Signal Process. Mag. **34**(2), 22–35 (2017)

[PXL+20] H. Phan, Y. Xie, S. Liao, J. Chen, B. Yuan, CAG: a real-time low-cost enhanced-robustness high-transferability content-aware adversarial attack generator, in *Proceedings of the AAAI Conference on Artificial Intelligence*, New York, NY, USA, Feb. 2020, pp. 5412–5419

[RABA18] E. Romera, J.M. Alvarez, L.M. Bergasa, R. Arroyo, ERFNet: efficient residual factorized convnet for real-time semantic segmentation. IEEE Trans. Intell. Transp. Syst. (TITS) **19**(1), 263–272 (2018)

[RDS+15] O. Russakovsky, J. Deng, S. Hao, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**(3), 211–252 (2015)

[RMOGVB18] K.R. Mopuri, U. Ojha, U. Garg, R. Venkatesh Babu, NAG: network for adversary generation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 742–751

[RT19] A. Rasouli, J.K. Tsotsos, Autonomous vehicles that interact with pedestrians: a survey of theory and practice. IEEE Trans. Intell. Transp. Syst. (TITS) **21**(3), 900–918 (2019)

[SAE18] SAE International, *SAE J3016: Surface Vehicle Recommended Practice – Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles* (SAE International, June 2018)

[SLJ+15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015, pp. 1–9

[SOZ+18] Z. Sun, M. Ozay, Y. Zhang, X. Liu, T. Okatani, Feature quantization for defending against distortion of images, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 7957–7966

[SZ15] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–14

[SZS+14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, AB, Canada, Dec. 2014, pp. 1–10

[WBSS04] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)

[WLCC18] X. Wei, S. Liang, N. Chen, X. Cao, Transferable Adversarial Attacks for Image and Video Object Detection, pp. 1–7, May 2018. arxiv:1811.12641

[WLH+20] Y. Wang, Z. Li, H. Hao, H. Yang, Y. Zheng, Research on visual perception technology of autonomous driving based on improved convolutional neural network. J. Phys.: Conf. Ser. **1550**(3), 21–27 (2020)

[WWX+20] D. Wu, Y. Wang, S.-T. Xia, J. Bailey, X. Ma, Skip Connections Matter: On the Transferability of Adversarial Examples Generated With ResNets, pp. 1–15, Feb. 2020. arxiv:2002.05990

[WXZ20] W. Jiangqing, X. Hao, J. Zhao, Automatic lane identification using the roadside LiDAR sensors. IEEE Intell. Transp. Syst. Mag. **12**(1), 25–34 (2020)

[WZTE18] L. Wu, Z. Zhu, C. Tai, E. Weinan, Understanding and Enhancing the Transferability of Adversarial Examples, pp. 1–15, Feb. 2018. arxiv:1802.09707

[XDL+18] C. Xiao, R. Deng, B. Li, F. Yu, M. Liu, D. Song, Characterizing adversarial examples based on spatial consistency information for semantic segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, Sept. 2018, pp. 217–234

[XZL+18] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, D. Song, Spatially Transformed Adversarial Examples, pp. 1–29, Jan. 2018. arxiv:1801.02612

[ZBIK20a] C. Zhang, P. Benz, T. Imtiaz, I.-S. Kweon, CD-UAP: class discriminative universal adversarial perturbation, in *Proceedings of the AAAI Conference on Artificial Intelligence*, New York, NY, USA, Feb. 2020, pp. 6754–6761

[ZBIK20b] C. Zhang, P. Benz, T. Imtiaz, I.-S. Kweon, Understanding adversarial examples from the mutual influence of images and perturbations, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference, June 2020, pp. 14521–14530

[ZPIE17] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017, pp. 2242–2251

[ZWJW20] K. Zhang, S.J. Wang, L. Ji, C. Wang, DNN based camera and LiDAR fusion framework for 3D object recognition. J. Phys.: Conf. Ser. **1518**(1), 12–44 (2020)

Invertible Neural Networks for Understanding Semantics of Invariances of CNN Representations

Robin Rombach, Patrick Esser, Andreas Blattmann, and Björn Ommer

Abstract To tackle increasingly complex tasks, it has become an essential ability of neural networks to learn abstract representations. These task-specific representations and, particularly, the invariances they capture turn neural networks into black-box models that lack interpretability. To open such a black box, it is, therefore, crucial to uncover the different semantic concepts a model has learned as well as those that it has learned to be invariant to. We present an approach based on invertible neural networks (INNs) that (i) recovers the task-specific, learned invariances by disentangling the remaining factor of variation in the data and that (ii) invertibly transforms these recovered invariances combined with the model representation into an equally expressive one with accessible semantic concepts. As a consequence, neural network representations become understandable by providing the means to (i) expose their semantic meaning, (ii) semantically modify a representation, and (iii) visualize individual learned semantic concepts and invariances. Our invertible approach significantly extends the abilities to understand black-box models by enabling post hoc interpretations of state-of-the-art networks without compromising their performance. Our implementation is available at https://compvis.github.io/invariances/.

1 Introduction

Key to the wide success of deep neural networks is end-to-end learning of powerful hidden representations that aim to *(i)* capture all task-relevant characteristics while *(ii)* being invariant to all other variabilities in the data [LeC12, AS18]. Deep learning

R. Rombach (✉) · P. Esser · A. Blattmann · B. Ommer
HCI, IWR, Heidelberg University, Berliner Str. 43, 69120 Heidelberg, Germany
e-mail: robin.rombach@iwr.uni-heidelberg.de

P. Esser
e-mail: patrick.esser@iwr.uni-heidelberg.de

A. Blattmann
e-mail: bjoern.ommer@iwr.uni-heidelberg.de

B. Ommer
e-mail: andreas.blattmann@iwr.uni-heidelberg.de

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_7

can yield abstract representations that are perfectly adapted feature encodings for the task at hand. However, their increasing abstraction capability and performance comes at the expense of a lack in interpretability [BBM+15]: although the network may solve a problem, it does not convey an understanding of its predictions or their causes, often leaving the impression of a black box [Mil19]. In particular, users are missing an explanation of semantic concepts that the model has learned to *represent* and of those it has learned to *ignore*, i.e., its invariances.

Providing such explanations and an understanding of network predictions and their causes is thus crucial for transparent AI. Not only is this relevant to discover limitations and promising directions for future improvements of the AI system itself, but also for compliance with legislation [GF17, Eur20], knowledge distillation from such a system [Lip18], and post hoc verification of the model [SWM17]. Consequently, research on interpretable deep models has recently gained a lot of attention, particularly methods that investigate latent representations to understand what the model has learned [SWM17, SZS+14, BZK+17, FV18, ERO20].

Challenges and aims: Assessing these latent representations is challenging due to two fundamental issues: *(i)* To achieve robustness and generalization despite noisy inputs and data variability, hidden layers exhibit a distributed coding of semantically meaningful concepts [FV18]. Attributing semantics to a single neuron via backpropagation [MLB+17] or synthesis [YCN+15] is thus impossible without altering the network [MSM18, ZKL+16], which typically degrades performance. *(ii)* End-to-end learning trains deep representations toward a goal task, making them invariant to features irrelevant for this goal. Understanding these characteristics that a representation has abstracted away is challenging, since we essentially need to portray features that have been discarded.

These challenges call for a method that can interpret existing network representations by recovering their invariances without modifying them. Given these recovered invariances, we seek an invertible mapping that translates a representation *and* the invariances onto understandable semantic concepts. The mapping disentangles the distributed encoding of the high-dimensional representation and its invariances by projecting them onto separate multi-dimensional factors that correspond to human-understandable semantic concepts. Both this translation and the recovering of invariances are implemented with invertible neural networks (INNs) [Red93, DSB17, KD18]. For the translation, this guarantees that the resulting understandable representation is equally expressive as the model representation combined with the recovered invariances (no information is lost). Its invertibility also warrants that feature modifications applied in the semantic domain correctly adjust the recovered representation.

Our contributions: Our contributions to a comprehensive understanding of deep representations are as follows: (i) We present an approach, which, by utilizing invertible neural networks, improves the understanding of representations produced by existing network architectures with no need for re-training or otherwise compromising their performance. (ii) Our generative approach is able to recover the invariances that result from the non-injective projection (of input onto a latent representation) which

deep networks typically learn. This model then provides a probabilistic visualization of the latent representation and its invariances. (iii) We bijectively translate an arbitrarily abstract representation and its invariances via a non-linear transformation into another representation of equal expressiveness, but with accessible semantic concepts. (iv) The invertibility also enables manipulation of the original latent representations in a semantically understandable manner, thus facilitating further diagnostics of a network.

2 Background

Two main approaches to interpretable AI can be identified, those which aim to incorporate interpretability directly into the design of models, and those which aim to provide interpretability to existing models [MSM18]. Approaches from the first category range from modifications of network architectures [ZKL+16], over regularization of models encouraging interpretability [LBMO19, PASC+20], toward combinations of both [ZNWZ18]. However, these approaches always involve a trade-off between model performance and model interpretability. Being of the latter category, our approach allows to interpret representations of existing models without compromising their performance.

To better understand what an existing model has learned, its representations must be studied [SWM17]. Syegedy et al. [SZS+14] show that both random directions and coordinate axes in the feature space of networks can represent semantic properties and conclude that they are not necessarily represented by individual neurons. Different works attempt to select groups of neurons which have a certain semantic meaning, such as based on scenes [ZKL+15], objects [SR15] and object parts [SRD14]. [BZK+17] studied the interpretability of neurons and found that a rotation of the representation space spanned by the neurons decreases its interpretability. While this suggests that the neurons provide a more interpretable basis compared to a random basis, [FV18] shows that the choice of basis is not the only challenge for interpretability of representations. Their findings demonstrate that learned representations are distributed, i.e., a single semantic concept is encoded by an activation pattern involving multiple neurons, and a single neuron is involved in the encoding of multiple different semantic concepts. Instead of selecting a set of neurons directly, [ERO20] learns an INN that transforms the original representation space to an interpretable space, where a single semantic concept is represented by a known group of neurons and a single neuron is involved in the encoding of just a single semantic concept. However, to interpret not only the representation itself but also its invariances, it is insufficient to transform only the representation itself. Our approach therefore transforms the latent representation space of an autoencoder, which has the capacity to represent its inputs faithfully, and subsequently translates a model representation and its invariances into this space for semantic interpretation and visualization.

A large body of works approach interpretability of existing networks based on visualizations. Selvaraju et al. [SCD+20] use gradients of network outputs

with respect to a convolutional layer to obtain coarse localization maps. Bach et al. [BBM+15] propose an approach to obtain pixel-wise relevance scores for a specific class of models which is generalized in [MLB+17]. To obtain richer visual interpretations, [ZF14, SVZ14, YCN+15, MV16] reconstruct images which maximally activate certain neurons. Nguyen et al. [NDY+16] use a generator network for this task, which was introduced in [DB16] for reconstructing images from their feature representation. Our key insight is that these existing approaches do not explicitly account for the invariances learned by a model. Invariances imply that feature inversion is a one-to-many mapping and thus they must be recovered to solve the task. Recently, [SGM+20] introduced a GAN-based approach that utilizes features of a pretrained classifier as a semantic pyramid for image generation. Nash et al. [NKW19] used samples from an autoregressive model of images conditioned on a feature representation to gain insights into the representation's invariances. In contrast, our approach recovers an explicit representation of the invariances, which can be recombined with modified feature representations, and thus makes the effect of modifications to representations, e.g., through adversarial attacks, visible.

Other works consider visual interpretations for specialized models. Santurkar et al. [SIT+19] showed that the quality of images which maximally activate certain neurons is significantly improved when activating neurons of an adversarially robust classifier. Bau et al. [BZS+19] explore the relationship between neurons and the images produced by a generative adversarial network. For the same class of models, [GAOI19] finds directions in their input space which represent semantic concepts corresponding to certain cognitive properties. Such semantic directions have previously also been found in classifier networks [UGP+17] but requires aligned data. All of these approaches require either special training of models, are limited to a very special class of models which already provide visualizations, or depend on special assumptions on model and data. In contrast, our approach can be applied to arbitrary models without re-training or modifying them, and provides both visualizations and semantic explanations, for both the model's representation and its learned invariances.

3 Method

Common tasks of computer vision can be phrased as a mapping from an input image $\boldsymbol{x}$ to some output $\boldsymbol{f}(\boldsymbol{x})$ such as a classification of the image, a regression (e.g., of object locations), a (semantic) segmentation map, or a re-synthesis that yields another image. Deep learning utilizes a hierarchy of intermediate network layers that gradually transform the input into increasingly more abstract representations. Let $\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x}) \in \mathbb{R}^{N_z}$ be the representation extracted by one such layer (without loss of generality we consider $\boldsymbol{z}$ to be an N_z-dim vector, flattening it if necessary) and $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\Psi}(\boldsymbol{z}) = \boldsymbol{\Psi}(\boldsymbol{\Phi}(\boldsymbol{x}))$ the mapping onto the output.

An essential characteristic of a deep feature encoding $\boldsymbol{z}$ is the increasing abstractness of higher feature encoding layers and the resulting reduction of information.

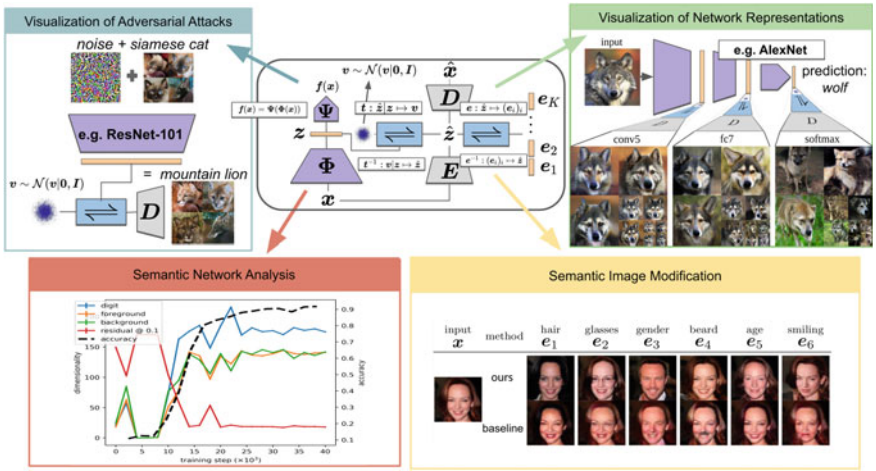


Fig. 1 Proposed architecture. We provide post hoc interpretation for a given deep network $\boldsymbol{f} = \boldsymbol{\Psi} \circ \boldsymbol{\Phi}$. For a deep representation $\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x})$ a conditional INN $\boldsymbol{t}$ recovers $\boldsymbol{\Phi}$'s invariances $\boldsymbol{v}$ from a representation $\hat{\boldsymbol{z}}$ which contains entangled information about *both* $\boldsymbol{z}$ and $\boldsymbol{v}$. The INN $\boldsymbol{e}$ then translates the representation $\hat{\boldsymbol{z}}$ into a factorized representation with accessible semantic concepts. This approach allows for various applications, including visualizations of network representations of natural (green box) and altered inputs (blue box), semantic network analysis (red box) and semantic image modifications (yellow box)

This reduction generally causes the feature encoding to become invariant to those properties of the input image, which do not provide salient information for the task at hand [CWG+18]. To explain a latent representation, we need to recover such invariances $\boldsymbol{v}$ and make $\boldsymbol{z}$ *and* $\boldsymbol{v}$ interpretable by learning a bijective mapping onto understandable semantic concepts, see Fig. 1. Section 3.1 describes our INN $\boldsymbol{t}$ to recover an encoding $\boldsymbol{v}$ of the invariances. Due to the generative nature of $\boldsymbol{t}$, our approach can correctly sample visualizations of the model representation and its invariances without leaving the underlying data distribution and introducing artifacts. With $\boldsymbol{v}$ then available, Sect. 3.2 presents an INN $\boldsymbol{e}$ that translates $\boldsymbol{t}$'s encoding of $\boldsymbol{z}$ and $\boldsymbol{v}$ without losing information onto disentangled semantic concepts. Moreover, the invertibility allows modifications in the semantic domain to correctly project back onto the original representation or into image space.

3.1 Recovering the Invariances of Deep Models

Learning an encoding to help recover invariances: Key to a deep representation is not only the information $\boldsymbol{z}$ captures, but also what is has learned to abstract away. To learn what $\boldsymbol{z}$ misses with respect to $\boldsymbol{x}$, we need an encoding $\hat{\boldsymbol{z}}$, which, in contrast to $\boldsymbol{z}$, includes the invariances exhibited by $\boldsymbol{z}$. Without making prior assumptions about

the deep model $\boldsymbol{f}$, autoencoders provide a generic way to obtain such an encoding $\hat{z}$, since they ensure that their input $\boldsymbol{x}$ can be recovered from their learned representation $\hat{z}$, which hence also comprises the invariances.

Therefore, we learn an autoencoder with an encoder $\boldsymbol{E}$ that provides the data representation $\hat{z} = \boldsymbol{E}(\boldsymbol{x})$ and a decoder $\boldsymbol{D}$ producing the data reconstruction $\hat{\boldsymbol{x}} = \boldsymbol{D}(\hat{z})$. Section 3.2 will utilize the decoding from $\hat{z}$ to $\hat{\boldsymbol{x}}$ to visualize both z and $\boldsymbol{v}$. The autoencoder is trained to reconstruct its inputs by minimizing a perceptual metric between input and reconstruction, $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|$, as in [DB16]. The details of the architecture and training procedure can be found in Sect. 3.3, autoencoder E, D. It is crucial that the autoencoder only needs to be trained once on the training data. Consequently, the same $\boldsymbol{E}$ can be used to interpret different representations z, e.g., different models or layers within a model, thus ensuring fair comparisons between them. Moreover, the complexity of the autoencoder can be adjusted based on the computational needs, allowing us to work with much lower dimensional encodings $\hat{z}$ compared to reconstructing the invariances directly from the images $\boldsymbol{x}$. This reduces the computational demands of our approach significantly.

Learning a conditional INN that recovers invariances: Due to the reconstruction task of the autoencoder, $\hat{z}$ not only contains the invariances $\boldsymbol{v}$, but also the representation z. Thus, we must disentangle [EHO19, LSOL20, KSLO19] $\boldsymbol{v}$ and z using a mapping $\boldsymbol{t}(\cdot|z) : \hat{z} \mapsto \boldsymbol{v} = \boldsymbol{t}(\hat{z}|z)$ which, depending on z, extracts $\boldsymbol{v}$ from $\hat{z}$.

Besides extracting the invariances from a given $\hat{z}$, $\boldsymbol{t}$ must also enable an inverse mapping from given model representations z to $\hat{z}$ to support a further mapping onto semantic concepts Sect. 3.2 and visualization based on $\boldsymbol{D}(\hat{z})$. There are many different $\boldsymbol{x}$ with $\boldsymbol{\Phi}(\boldsymbol{x}) = z$, namely, all those $\boldsymbol{x}$ which differ only in properties that $\boldsymbol{\Phi}$ is invariant to. Thus, there are also many different $\hat{z}$ that this mapping must recover. Consequently, the mapping from z to $\hat{z}$ is set-valued. However, to understand $\boldsymbol{f}$ we do not want to recover all possible $\hat{z}$, but only those which are likely under the training distribution of the autoencoder. In particular, this excludes unnatural images such as those obtained by DeepDream [MOT15] or adversarial attacks [SZS+14]. In conclusion, we need to sample $\hat{z} \sim p(\hat{z}|z)$.

To avoid a costly inversion process of $\boldsymbol{\Phi}$, $\boldsymbol{t}$ must be invertible (implemented as an INN) so that a change of variables

$$p(\hat{z}|z) = \frac{p(\boldsymbol{v}|z)}{|\det \nabla(\boldsymbol{t}^{-1})(\boldsymbol{v}|z)|} \quad , \text{where } \boldsymbol{v} = \boldsymbol{t}(\hat{z}|z), \tag{1}$$

yields $p(\hat{z}|z)$ by means of the distribution $p(\boldsymbol{v}|z)$ of invariances, given a model representation z. Here, the denominator denotes the absolute value of the determinant of Jacobian $\nabla(\boldsymbol{t}^{-1})$ of $\boldsymbol{v} \mapsto \boldsymbol{t}^{-1}(\boldsymbol{v}|z) = \hat{z}$, which is efficient to compute for common invertible network architectures. Consequently, we obtain $\hat{z}$ for given z by sampling from the invariant space $\boldsymbol{v}$ given z and then applying $\boldsymbol{t}^{-1}$,

$$\hat{z} \sim p(\hat{z}|z) \quad \Longleftrightarrow \quad \boldsymbol{v} \sim p(\boldsymbol{v}|z), \hat{z} = \boldsymbol{t}^{-1}(\boldsymbol{v}|z). \tag{2}$$

Since $\boldsymbol{v}$ is the invariant space for $\boldsymbol{z}$, both are complementary thus implying independence $p(\boldsymbol{v}|\boldsymbol{z}) = p(\boldsymbol{v})$. Because a powerful transformation $\boldsymbol{t}^{-1}$ can transform between two arbitrary densities, we can assume without loss of generality a Gaussian prior $p(\boldsymbol{v}) = \mathcal{N}(\boldsymbol{v}|\boldsymbol{0}, \boldsymbol{I})$, where $\boldsymbol{I}$ is the identity matrix. Given this prior, our task is then to learn the transformation $\boldsymbol{t}$ that maps $\mathcal{N}(\boldsymbol{v}|\boldsymbol{0}, \boldsymbol{I})$ onto $p(\hat{\boldsymbol{z}}|\boldsymbol{z})$. To this end, we maximize the log-likelihood of $\hat{\boldsymbol{z}}$ given $\boldsymbol{z}$, which results in a per-example loss of

$$J_e(\hat{\boldsymbol{z}}, \boldsymbol{z}) = -\log p(\hat{\boldsymbol{z}}|\boldsymbol{z}) = -\log \mathcal{N}(\boldsymbol{t}(\hat{\boldsymbol{z}}|\boldsymbol{z})|\boldsymbol{0}, \boldsymbol{I}) - \log|\det \nabla \boldsymbol{t}(\hat{\boldsymbol{z}}|\boldsymbol{z})|. \tag{3}$$

Minimizing this loss over the training data distribution $p(\boldsymbol{x})$ gives $\boldsymbol{t}$, a bijective mapping between $\hat{\boldsymbol{z}}$ and $(\boldsymbol{z}, \boldsymbol{v})$,

$$J(\boldsymbol{t}) = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[J_e(\boldsymbol{E}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}))\right] \tag{4}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\frac{1}{2}\|\boldsymbol{t}(\boldsymbol{E}(\boldsymbol{x})|\boldsymbol{\Phi}(\boldsymbol{x}))\|^2 + N_{\hat{\boldsymbol{z}}}\log 2\pi - \log|\det \nabla \boldsymbol{t}(\boldsymbol{E}(\boldsymbol{x})|\boldsymbol{\Phi}(\boldsymbol{x}))|\right]. \tag{5}$$

Note that both $\boldsymbol{E}$ and $\boldsymbol{\Phi}$ remain fixed during minimization of J.

3.2 *Interpreting Representations and Their Invariances*

Visualizing representations and invariances: For an image representation $\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x})$, (2) presents an efficient approach (a single forward pass through the INN $\boldsymbol{t}$) to sample an encoding $\hat{\boldsymbol{z}}$, which is a combination of $\boldsymbol{z}$ with a particular realization of its invariances $\boldsymbol{v}$. Sampling multiple realizations of $\hat{\boldsymbol{z}}$ for a given $\boldsymbol{z}$ highlight what remains constant and what changes due to different $\boldsymbol{v}$: information preserved in the representation $\boldsymbol{z}$ remains constant over different samples and information discarded by the model ends up in the invariances $\boldsymbol{v}$ and shows changes over different samples. Visualizing the samples $\hat{\boldsymbol{z}} \sim p(\hat{\boldsymbol{z}}|\boldsymbol{z})$ with $\hat{\boldsymbol{x}} = \boldsymbol{D}(\hat{\boldsymbol{z}})$ portrays this constancy and changes due to different $\boldsymbol{v}$. To complement this visualization, in the following, we learn a transformation of $\hat{\boldsymbol{z}}$ into a semantically meaningful representation which allows to uncover the semantics captured by $\boldsymbol{z}$ and $\boldsymbol{v}$.

Learning an INN to produce semantic interpretations: The autoencoder representation $\hat{\boldsymbol{z}}$ is an equivalent representation of $(\boldsymbol{z}, \boldsymbol{v})$ but its feature dimensions do not necessarily correspond to semantic concepts [FV18]. More generally, without supervision, we cannot reliably discover semantically meaningful, explanatory factors of $\hat{\boldsymbol{z}}$ [LBL+19]. In order to explain $\hat{\boldsymbol{z}}$ in terms of given semantic concepts, we apply the approach of [ERO20] and learn a bijective transformation of $\hat{\boldsymbol{z}}$ to an interpretable representation $\boldsymbol{e}(\hat{\boldsymbol{z}})$ where different groups of components, called factors, correspond to semantic concepts.

To learn the transformation $\boldsymbol{e}$, we parameterize $\boldsymbol{e}$ by an INN and assume that semantic concepts are defined implicitly by pairs of images, i.e., for each semantic

concept we have access to training pairs $\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{b}}$ that have the respective concept in common. For example, the semantic concept "smiling" is defined by pairs of images, where either both images show smiling persons or both images show non-smiling persons. Applying this formulation, input pairs which are similar in a certain semantic concept are similar in the corresponding factor of the interpretable representation $\boldsymbol{e}(\hat{z})$.

Following [ERO20], the loss for training the invertible network $\boldsymbol{e}$ is then given by

$$J(\boldsymbol{e}) = \mathbb{E}_{\boldsymbol{x}^{\mathrm{a}},\boldsymbol{x}^{\mathrm{b}}}\big[-\log p(\boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x}^{\mathrm{a}})), \boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x}^{\mathrm{b}}))) - \log|\det \nabla \boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x}^{\mathrm{a}}))| - \log|\det \nabla \boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x}^{\mathrm{b}}))|\big]. \tag{6}$$

Interpretation by applying the learned INNs: After training, the combination of $\boldsymbol{e}$ with $\boldsymbol{t}$ from Sect. 3.1 provides semantic interpretations given a model representation z: (2) gives realizations of the invariances $\boldsymbol{v}$ which are combined with z to produce $\hat{z} = \boldsymbol{t}^{-1}(\boldsymbol{v}|z)$. Then $\boldsymbol{e}$ transforms $\hat{z}$ without loss of information into a semantically accessible representation $(\boldsymbol{e}_i)_i = \boldsymbol{e}(\hat{z}) = \boldsymbol{e}(\boldsymbol{t}^{-1}(\boldsymbol{v}|z))$ consisting of different semantic factors $\boldsymbol{e}_i$. Comparing the $\boldsymbol{e}_i$ for different model representations z and invariances $\boldsymbol{v}$ allows us to observe which semantic concepts the model representation $z = \boldsymbol{\Phi}(\cdot)$ is sensitive to, and which it is invariant to.

Semantic Modifications of Latent Representations: The transformations $\boldsymbol{t}^{-1}$ and $\boldsymbol{e}$ not only interpret a representation z in terms of accessible semantic concepts $(\boldsymbol{e}_i)_i$. Given $\boldsymbol{v} \sim p(\boldsymbol{v})$, they also allow to modify $\hat{z} = \boldsymbol{t}^{-1}(\boldsymbol{v}|z)$ in a semantically meaningful manner by altering its corresponding $(\boldsymbol{e}_i)_i$ and then applying the inverse translation $\boldsymbol{e}^{-1}$,

$$\hat{z} \xrightarrow{\boldsymbol{e}} (\boldsymbol{e}_i) \xrightarrow{\text{modification}} (\boldsymbol{e}_i^*) \xrightarrow{\boldsymbol{e}^{-1}} \hat{z}^*. \tag{7}$$

The modified representation $\hat{z}^*$ is then readily transformed back into image space $\hat{\boldsymbol{x}}^* = \boldsymbol{D}(\hat{z}^*)$. Besides visual interpretation of the modification, $\hat{\boldsymbol{x}}^*$ can be fed into the model $\boldsymbol{\Psi}(\boldsymbol{\Phi}(\hat{\boldsymbol{x}}^*))$ to probe for sensitivity to certain semantic concepts.

3.3 Implementation Details

In this section, we provide implementation details about the exact training procedure and architecture of all components of our approach. This is only for the sake of clarity and completeness. For readers who are already familiar with INNs or people rather interested in the higher level ideas of our approach than in its technical details, this section can be safely skipped.

Autoencoder $\boldsymbol{E}$, $\boldsymbol{D}$: In Sect. 3.1, we introduced an autoencoder to obtain a representation $\hat{z}$ of $\boldsymbol{x}$, which includes the invariances abstracted away by a given model representation z. This autoencoder consists of an encoder $E(\boldsymbol{x})$ and a decoder $D(\hat{z})$.

Table 1 Autoencoder architecture on datasets with images of resolution 28×28

Encoder: FC and Norm refer to fully connected layer and batch normalization, respectively. The term "down" denotes downsampling by factor 2. Leaky ReLU (LRelu) uses a slope parameter of 0.2.

RGB image $\boldsymbol{x} \in \mathbb{I}^{28\times 28\times 3}, \mathbb{I} = [0,1]$
Conv down, Norm, LReLU $\to \mathbb{R}^{14\times 14\times 64}$
Conv down, Norm, LReLU $\to \mathbb{R}^{7\times 7\times 128}$
FC $\mapsto (\boldsymbol{\mu}, \boldsymbol{\sigma}\boldsymbol{\sigma}^{\mathrm{T}}) \in \mathbb{R}^{64} \times \mathbb{R}^{64}$

Decoder: FC and Norm refer to fully connected layer and batch normalization, respectively. Leaky ReLU (LRelu) uses a slope parameter of 0.2.

$\hat{\boldsymbol{z}} \in \mathbb{R}^{64} \sim \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}\boldsymbol{\sigma}^{\mathrm{T}}))$
FC $\to \mathbb{R}^{7\times 7\times 128}$
Conv transpose, Norm, LReLU $\to \mathbb{R}^{14\times 14\times 64}$
Conv transpose, Tanh $\to \mathbb{I}^{28\times 28\times 3}, \mathbb{I} = [0,1]$

Because the INNs $\boldsymbol{t}$ and $\boldsymbol{e}$ transform the distribution of $\hat{\boldsymbol{z}}$, we must ensure a strictly positive density for $\hat{\boldsymbol{z}}$ to avoid degenerate solutions. This is readily achieved with a stochastic encoder, i.e., we predict mean $\boldsymbol{E}(\boldsymbol{x})_{\boldsymbol{\mu}}$ and diagonal $\boldsymbol{E}(\boldsymbol{x})_{\boldsymbol{\sigma}^2}$ of a Gaussian distribution, and obtain the desired representation as $\hat{\boldsymbol{z}} \sim \mathcal{N}(\hat{\boldsymbol{z}}|\boldsymbol{E}(\boldsymbol{x})_{\boldsymbol{\mu}}, \mathrm{diag}(\boldsymbol{E}(\boldsymbol{x})_{\boldsymbol{\sigma}^2}))$. Following [DW19], we train this autoencoder as a variational autoencoder using the reparameterization trick [KW14, RMW14] to match the encoded distribution to a standard normal distribution, and jointly learn the scalar output variance γ under an image metric $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|$ to avoid blurry reconstructions. The resulting loss function is thus

$$J(\boldsymbol{E}, \boldsymbol{D}, \gamma) = \mathbb{E}_{\substack{\boldsymbol{x} \sim p(\boldsymbol{x}) \\ \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|0,\boldsymbol{I})}} \left[\frac{1}{\gamma} \|\boldsymbol{x} - \boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x})_{\boldsymbol{\mu}} + \epsilon \sqrt{\mathrm{diag}(\boldsymbol{E}(\boldsymbol{x})_{\boldsymbol{\sigma}^2})}\| + \log \gamma \right.$$
$$\left. + \frac{1}{2} \sum_{i=1}^{N_{\hat{\boldsymbol{z}}}} \left\{ (\boldsymbol{E}(\boldsymbol{x})_{\boldsymbol{\mu}})_i^2 + (\boldsymbol{E}(\boldsymbol{x})_{\boldsymbol{\sigma}^2})_i - \boldsymbol{1} - \log(\boldsymbol{E}(\boldsymbol{x})_{\boldsymbol{\sigma}^2})_i \right\} \right]. \tag{8}$$

Note that $\sqrt{(\cdot)}$ and $\log(\cdot)$ on multi-dimensional entities are applied element-wise. In the experiments shown in this chapter, we use images of spatial resolutions 28×28 and 128×128, resulting in different architectures for the autoencoder, summarized in Tables 1 and 2, respectively. For the encoder $\boldsymbol{E}$ processing images of spatial resolution 128×128, we use an architecture based on `ResNet-101` [HZRS16], and for the corresponding decoder $\boldsymbol{D}$ we use an architecture based on `BigGAN` [BDS19], where we include a small fully connected network to replace the class conditioning used in `BigGAN` by a conditioning on $\hat{\boldsymbol{z}}$.

In the 28×28 case, we use a squared L_2 loss for the image metric, which corresponds to the first term in (8). For our 128×128-models, we further use an improved metric as in [DB16], which includes additional perceptual [ZIE+18] and discriminator losses. The perceptual loss consists of L_1 feature distances obtained from different

Table 2 Autoencoder architecture for datasets with images of resolution 128×128

Encoder based on `Resnet-101`: Bottleneck, FC, and Norm refer to bottleneck residual unit, fully connected layer, and batch normalization, respectively. The term "down" denotes downsampling by factor 2.

RGB image $\boldsymbol{x} \in \mathbb{I}^{128\times128\times3}, \mathbb{I} = [0, 1]$
Conv down $\rightarrow \mathbb{R}^{64\times64\times64}$
Norm, ReLU, MaxPool $\rightarrow \mathbb{R}^{32\times32\times64}$
$3\times$ BottleNeck $\rightarrow \mathbb{R}^{32\times32\times256}$
$4\times$ BottleNeck down $\rightarrow \mathbb{R}^{16\times16\times512}$
$23\times$ BottleNeck down $\rightarrow \mathbb{R}^{8\times8\times1024}$
$3\times$ BottleNeck down $\rightarrow \mathbb{R}^{4\times4\times2048}$
AvgPool, FC $\mapsto (\boldsymbol{\mu}, \boldsymbol{\sigma\sigma}^{\mathrm{T}}) \in \mathbb{R}^{128} \times \mathbb{R}^{128}$

Decoder based on `BigGAN`: Embed, FC, Norm, and ResBlock refer to embedding layers, fully connected layers, batch normalization, and residual block, respectively. The term "up" denotes upsampling by factor 2. Leaky ReLU (LRelu) uses a slope parameter of 0.2.

$\hat{\boldsymbol{z}} \in \mathbb{R}^{128} \sim \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma\sigma}^{\mathrm{T}}))$ $3\times$ (FC, LReLU) $\rightarrow \mathbb{R}^{256}$ FC, Softmax $\rightarrow \mathbb{R}^{1000}$ Embed $\mapsto \boldsymbol{h} \in \mathbb{R}^{128}$
FC($\hat{\boldsymbol{z}}$) $\rightarrow \mathbb{R}^{4\times4\times16\cdot96}$
ResBlock($\hat{\boldsymbol{z}}$,$\boldsymbol{h}$) up $\rightarrow \mathbb{R}^{8\times8\times16\cdot96}$
ResBlock($\hat{\boldsymbol{z}}$,$\boldsymbol{h}$) up $\rightarrow \mathbb{R}^{16\times16\times8\cdot96}$
ResBlock($\hat{\boldsymbol{z}}$,$\boldsymbol{h}$) up $\rightarrow \mathbb{R}^{32\times32\times4\cdot96}$
ResBlock($\hat{\boldsymbol{z}}$,$\boldsymbol{h}$) up $\rightarrow \mathbb{R}^{64\times64\times2\cdot96}$
Non-Local Block $\rightarrow \mathbb{R}^{64\times64\times2\cdot96}$
ResBlock($\hat{\boldsymbol{z}}$,$\boldsymbol{h}$) up $\rightarrow \mathbb{R}^{64\times64\times96}$
Norm, ReLU, Conv up $\rightarrow \mathbb{R}^{128\times128\times3}$
Tanh $\mapsto \hat{\boldsymbol{x}} \in \mathbb{I}^{128\times128\times3}, \mathbb{I} = [0, 1]$

layers of a fixed, pretrained network. We use a `VGG-16` network pretrained on ImageNet and weighted distances of different layers as in [ZIE+18]. The discriminator is trained along with the autoencoder to distinguish reconstructed images from real images using a binary classification loss, and the autoencoder maximizes the log-probability that reconstructed images are classified as real images. The architectures of `VGG-16` and the discriminator are summarized in Table 3.

Details on the INN*e* for Revealing Semantics of Deep Representations: Previous works have successfully applied INNs for density estimation [DSB17], inverse problems [AKW+19], and on top of autoencoder representations [ERO20, XYA19, DMB+21, BMDO21b, BMDO21a] for a wide range of applications such as video synthesis [DMB+21, BMDO21b] and translation between pretrained, powerful networks [REO20]. This section provides details on how we embed the approach of [ERO20] to reveal the semantic concepts of autoencoder representations $\hat{\boldsymbol{z}}$, cf. Sect. 3.2.

Since we will never have examples for all relevant semantic concepts, we include a residual concept that captures the remaining variability of $\hat{\boldsymbol{z}}$, which is not explained by the given semantic concepts.

Following [ERO20], we learn a bijective transformation $\boldsymbol{e}(\hat{\boldsymbol{z}})$, which translates the non-interpretable representation $\hat{\boldsymbol{z}}$ invertibly into a factorized representation

Table 3 Architectures used to compute image metrics for the autoencoder, which were used for training the autoencoder $\boldsymbol{E}$, $\boldsymbol{D}$ on datasets with images of resolution 128×128

`VGG-16` pretrained on ImageNet for feature extraction. Output of bold layers are used to compute feature distances.

RGB image $\boldsymbol{x} \in \mathbb{R}^{128\times128\times3}$
$2\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{128\times128\times64}$
MaxPool $\rightarrow \mathbb{R}^{64\times64\times64}$
$2\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{64\times64\times128}$
MaxPool $\rightarrow \mathbb{R}^{32\times32\times128}$
$3\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{32\times32\times256}$
MaxPool $\rightarrow \mathbb{R}^{16\times16\times256}$
$3\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{16\times16\times512}$
MaxPool $\rightarrow \mathbb{R}^{8\times8\times512}$
$3\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{8\times8\times512}$

Discriminator: All convolutions use kernel size 4. Norm refers to batch normalization, leaky ReLU uses a slope parameter 0.2.

RGB image $\boldsymbol{x} \in \mathbb{R}^{128\times128\times3}$
Conv down, LReLU $\rightarrow \mathbb{R}^{64\times64\times64}$
Conv down, Norm, LReLU $\rightarrow \mathbb{R}^{32\times32\times128}$
Conv down, Norm, LReLU $\rightarrow \mathbb{R}^{16\times16\times256}$
Conv down, Norm, LReLU $\rightarrow \mathbb{R}^{8\times8\times512}$
Conv, Norm, LReLU $\rightarrow \mathbb{R}^{8\times8\times512}$
Conv $\rightarrow \mathbb{R}^{8\times8\times1}$

$(\boldsymbol{e}_i(\hat{\boldsymbol{z}}))_{i=0}^{K} = \boldsymbol{e}(\hat{\boldsymbol{z}})$, where each factor $\boldsymbol{e}_i \in \mathbb{R}^{N_{\boldsymbol{e}_i}}$ represents one of the given semantic concepts for $i = 1, \ldots, K$, and $\boldsymbol{e}_0 \in \mathbb{R}^{N_{\boldsymbol{e}_0}}$ is the residual concept.

The INN $\boldsymbol{e}$ establishes a one-to-one correspondence between an encoding and different semantic concepts and, conversely, enables semantic modifications to correctly alter the original encoding (see next section). Being an INN, $\boldsymbol{e}(\hat{\boldsymbol{z}})$ and $\hat{\boldsymbol{z}}$ need to have the same dimensionality and we set $N_{\boldsymbol{e}_0} = N_{\hat{\boldsymbol{z}}} - \sum_{i=1}^{K} N_{\boldsymbol{e}_i}$. We denote the indices of concept i with respect to $\boldsymbol{e}(\hat{\boldsymbol{z}})$ as $\mathcal{I}_i \subset \{1, \ldots, N_{\hat{\boldsymbol{z}}}\}$ such that we can write $\boldsymbol{e}_i = (\boldsymbol{e}(\hat{\boldsymbol{z}})_k)_{k\in\mathcal{I}_i}$.

In the following, we emphasize on deriving a loss function for training the semantic INN. Let $\boldsymbol{e}_i$ be the factor representing some semantic concept, e.g., gender, that the contents of two images $\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{b}}$ share. Then the projection of their encodings $\hat{\boldsymbol{z}}^{\mathrm{a}}, \hat{\boldsymbol{z}}^{\mathrm{b}}$ onto this semantic concept must be similar [ERO20, KWKT15],

$$\boldsymbol{e}_i(\hat{\boldsymbol{z}}^{\mathrm{a}}) \simeq \boldsymbol{e}_i(\hat{\boldsymbol{z}}^{\mathrm{b}}) \quad \text{where } \hat{\boldsymbol{z}}^{\mathrm{a}} = \boldsymbol{E}(\boldsymbol{x}^{\mathrm{a}}), \hat{\boldsymbol{z}}^{\mathrm{b}} = \boldsymbol{E}(\boldsymbol{x}^{\mathrm{b}}). \tag{9}$$

Moreover, to interpret $\hat{\boldsymbol{z}}$ we are interested in the separate contribution of different semantic concepts $\boldsymbol{e}_i$ that explain $\hat{\boldsymbol{z}}$. Hence, we seek a mapping $\boldsymbol{e}(\cdot)$ that strives to disentangle different concepts,

$$\boldsymbol{e}_i(\hat{\boldsymbol{z}}) \perp \boldsymbol{e}_j(\hat{\boldsymbol{z}}) \quad \forall i \neq j, \boldsymbol{x} \quad \text{where } \hat{\boldsymbol{z}} = E(\boldsymbol{x}). \tag{10}$$

The objectives in (9), (10) imply a correlation in $\boldsymbol{e}_i$ for pairs $\hat{\boldsymbol{z}}^{\mathrm{a}}$ and $\hat{\boldsymbol{z}}^{\mathrm{b}}$ and no correlation between concepts $\boldsymbol{e}_i, \boldsymbol{e}_j$ for $i \neq j$. This calls for a Gaussian distribution with a covariance matrix that reflects these requirements.

Let $\boldsymbol{e}^{\mathrm{a}} = (\boldsymbol{e}_i^{\mathrm{a}}) = (\boldsymbol{e}_i(\boldsymbol{E}(\boldsymbol{x}^{\mathrm{a}})))$ and $\boldsymbol{e}^{\mathrm{b}}$ likewise, where $\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{b}}$ are samples from a training distribution $p(\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{b}})$ for the ith semantic concept. The distribution of pairs $\boldsymbol{e}^{\mathrm{a}}$ and $\boldsymbol{e}^{\mathrm{b}}$ factorizes into a conditional and a marginal,

$$p(\boldsymbol{e}^{\mathrm{a}}, \boldsymbol{e}^{\mathrm{b}}) = p(\boldsymbol{e}^{\mathrm{b}}|\boldsymbol{e}^{\mathrm{a}})p(\boldsymbol{e}^{\mathrm{a}}). \tag{11}$$

Objective (10) implies a diagonal covariance for the marginal distribution $p(\boldsymbol{e}^{\mathrm{a}})$, i.e., a standard normal distribution, and (9) entails a correlation between $\boldsymbol{e}_i^{\mathrm{a}}$ and $\boldsymbol{e}_i^{\mathrm{b}}$. Therefore, the correlation matrix is $\boldsymbol{\Sigma}^{\mathrm{ab}} = \rho\,\mathrm{diag}((\delta_{\mathcal{I}_i}(k))_{k=1}^{N_{\hat{z}}})$, where

$$\delta_{\mathcal{I}_i}(k) = \begin{cases} 1 & \text{if } k \in \mathcal{I}_i, \\ 0 & \text{else.} \end{cases}$$

By symmetry, $p(\boldsymbol{e}^{\mathrm{b}}) = p(\boldsymbol{e}^{\mathrm{a}})$, which gives

$$p(\boldsymbol{e}^{\mathrm{b}}|\boldsymbol{e}^{\mathrm{a}}) = \mathcal{N}(\boldsymbol{e}^{\mathrm{b}}|\boldsymbol{\Sigma}^{\mathrm{ab}}\boldsymbol{e}^{\mathrm{a}}, \boldsymbol{I} - (\boldsymbol{\Sigma}^{\mathrm{ab}})^2). \tag{12}$$

Inserting (12) and a standard normal distribution for $p(\boldsymbol{e}^{\mathrm{a}})$ into (11) yields the negative log-likelihood for a pair $\boldsymbol{e}^{\mathrm{a}}, \boldsymbol{e}^{\mathrm{b}}$.

Given pairs $\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{b}}$ as training data, another change of variables from $\hat{\boldsymbol{z}}^{\mathrm{a}} = \boldsymbol{E}(x^{\mathrm{a}})$ to $\boldsymbol{e}^{\mathrm{a}} = \boldsymbol{e}(\hat{\boldsymbol{z}}^{\mathrm{a}})$ gives the training loss function for $\boldsymbol{e}$ as the negative log-likelihood of $\hat{\boldsymbol{z}}^{\mathrm{a}}, \hat{\boldsymbol{z}}^{\mathrm{b}}$,

$$\begin{aligned} J(\boldsymbol{e}) = \mathbb{E}_{\boldsymbol{x}^{\mathrm{a}},\boldsymbol{x}^{\mathrm{b}}} \big[&- \log p(\boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x}^{\mathrm{a}})), \boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x}^{\mathrm{b}}))) \\ &- \log |\det \nabla \boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x}^{\mathrm{a}}))| - \log |\det \nabla \boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x}^{\mathrm{b}}))| \big]. \end{aligned} \tag{13}$$

For simplicity, we have derived the loss for a single semantic concept $\boldsymbol{e}_i$. Simply summing over the losses of different semantic concepts yields their joint loss function and allows us to learn a joint translator $\boldsymbol{e}$ for all of them.

In the following, we focus on the log-likelihood of pairs. The loss for $\boldsymbol{e}$ in (13) contains the log-likelihood of pairs $\boldsymbol{e}^{\mathrm{a}}, \boldsymbol{e}^{\mathrm{b}}$. Inserting (12) and a standard normal distribution for $p(\boldsymbol{e}^{\mathrm{a}})$ into (11) yields

$$-\log p(\boldsymbol{e}^{\mathrm{a}}, \boldsymbol{e}^{\mathrm{b}}) = \frac{1}{2}\left(\sum_{k\in\mathcal{I}_i} \frac{(\boldsymbol{e}_k^{\mathrm{b}} - \rho \boldsymbol{e}_k^{\mathrm{a}})^2}{1-\rho^2} + \sum_{k\in\mathcal{I}_i^c} (\boldsymbol{e}_k^{\mathrm{b}})^2 + \sum_{k=1}^{N_{\hat{z}}} (\boldsymbol{e}_k^{\mathrm{a}})^2 \right) + C, \tag{14}$$

where $C = C(\rho, N_{\hat{z}})$ is a constant that can be ignored for the optimization process. $\rho \in (0, 1)$ determines the relative importance of loss terms corresponding to the similarity requirement in (9) and the independence requirement in (10). We use a fixed value of $\rho = 0.9$ for all experiments.

In the following, we describe the architecture of the semantic INN. In our implementation, $\boldsymbol{e}$ is built by stacking invertible blocks, see Fig. 2, which consist of three

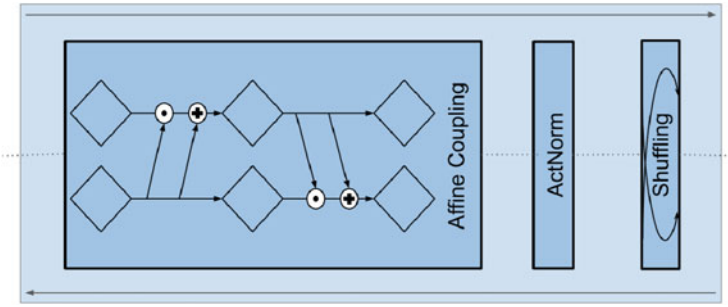


Fig. 2 A single invertible block used to build our invertible neural networks

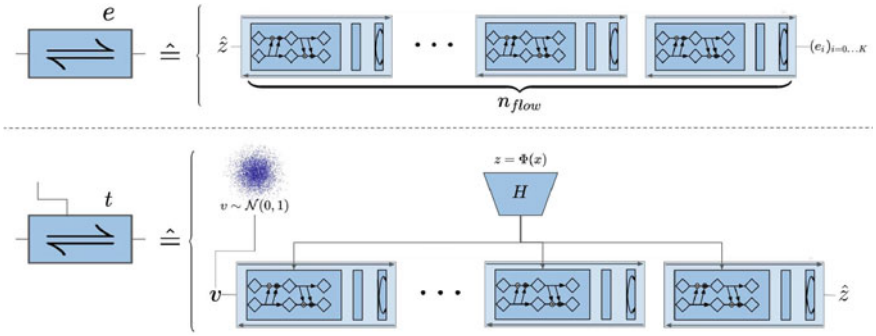


Fig. 3 Architectures of our INN models. *top:* The semantic INN $\boldsymbol{e}$ consists of stacked invertible blocks. *bottom:* The conditional INN $\boldsymbol{t}$ is composed of a embedding module $\boldsymbol{H}$ that downsamples (upsamples if necessary) a given model representation $\boldsymbol{h} = \boldsymbol{H}(z) = \boldsymbol{H}(\boldsymbol{\Phi}(\boldsymbol{x}))$. Subsequently, $\boldsymbol{h}$ is concatenated to the inputs of each block of the invertible model

invertible layers: coupling blocks [DSB17], actnorm layers [KD18], and shuffling layers. The final output is split into the factors $(\boldsymbol{e}_i)$, see Fig. 3.

Coupling blocks split their input $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ along the channel dimension and use fully connected neural networks $\boldsymbol{s}_i$ and $\boldsymbol{\tau}_i$ to perform the following computation:

$$\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}_1 \odot \boldsymbol{s}_1(\boldsymbol{x}_2) + \boldsymbol{\tau}_1(\boldsymbol{x}_2), \tag{15}$$

$$\tilde{\boldsymbol{x}}_2 = \boldsymbol{x}_2 \odot \boldsymbol{s}_2(\tilde{\boldsymbol{x}}_1) + \boldsymbol{\tau}_2(\tilde{\boldsymbol{x}}_1), \tag{16}$$

with the element-wise multiplication operator $\odot$. Actnorm layers consist of learnable shift and scale parameters for each channel, which are initialized to ensure activations with zero mean and unit variance on the first training batch. Shuffling layers use a fixed, randomly initialized permutation to shuffle the channels of its input, which provides a better mixing of channels for subsequent coupling layers.

Conditional INN t for recovering invariances of deep representations: We first elaborate on the architecture of the conditional INN. We build the conditional invert-

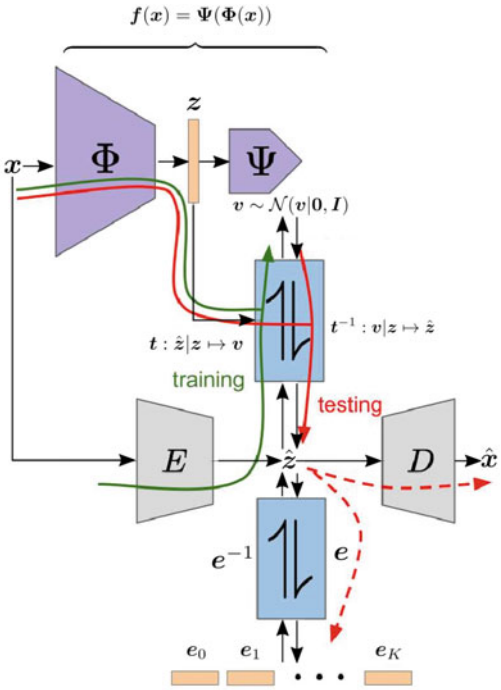


Fig. 4 Graphical distinction of information flow during training and inference. During training of $\boldsymbol{t}$, the encoder $\boldsymbol{E}$ provides an (approximately complete) data representation, which is used to learn the invariances of a given model's representations $\boldsymbol{z}$. At inference, the encoder is not necessarily needed anymore: given a representation $\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x})$, invariances can be sampled from the prior distribution and decoded into data space through $\boldsymbol{t}^{-1}$

ible neural network $\boldsymbol{t}$ by expanding the semantic model $\boldsymbol{e}$ as follows: Given a model representation $\boldsymbol{z}$, which is used as the conditioning of the INN, we first calculate its embedding

$$\boldsymbol{h} = \boldsymbol{H}(\boldsymbol{z}) \tag{17}$$

which is subsequently fed into the affine coupling block:

$$\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}_1 \odot \boldsymbol{s}_1(\boldsymbol{x}_2, \boldsymbol{h}) + \boldsymbol{\tau}_1(\boldsymbol{x}_2, \boldsymbol{h}), \tag{18}$$

$$\tilde{\boldsymbol{x}}_2 = \boldsymbol{x}_2 \odot \boldsymbol{s}_2(\tilde{\boldsymbol{x}}_1, \boldsymbol{h}) + \boldsymbol{\tau}_2(\tilde{\boldsymbol{x}}_1, \boldsymbol{h}), \tag{19}$$

with $\odot$ again being an element-wise multiplication operator, where $\boldsymbol{s}_i$ and $\boldsymbol{\tau}_i$ are modified from (16) such that they are capable of processing a concatenated input $(\boldsymbol{x}_i, \boldsymbol{h})$. The embedding module $\boldsymbol{H}$ is usually a shallow convolutional neural network used to down-/upsample a given model representation $\boldsymbol{z}$ to a size that the networks $\boldsymbol{s}_i$ and $\boldsymbol{\tau}_i$ are able to process. This means that $\boldsymbol{t}$, analogous to $\boldsymbol{e}$, consists of stacked invertible blocks, where each block is composed of coupling blocks, actnorm layers, and shuffling layers, cf. Sect. 3.3, details on the INN $\boldsymbol{e}$ for revealing semantics of deep representations, and Fig. 2. The complete architectures of both $\boldsymbol{t}$ and $\boldsymbol{e}$ are depicted in Fig. 3. Additionally, Fig. 4 provides a graphical distinction of the training and testing process of $\boldsymbol{t}$. During training, the autoencoder $\boldsymbol{D} \circ \boldsymbol{E}$ provides a representation of the

data that contains both the invariances and the representation of some model w.r.t. the input $\boldsymbol{x}$. After training of $\boldsymbol{t}$, the encoder may be discarded and visual decodings and/or semantic interpretations of a model representation $\boldsymbol{z}$ can be obtained by sampling and transforming $\boldsymbol{v}$ as described in (2).

4 Experiments

To explore the applicability of our approach, we conduct experiments on several models: `SqueezeNet` [IHM+16], which provides lightweight classification, `FaceNet` [SKP15], a baseline for face recognition and clustering, trained on the VGGFace2 dataset [CSX+18], and variants of `ResNet` [HZRS16], a popular architecture often used when fine-tuning a classifier on a specific task and dataset.

Experiments are conducted on the following datasets: CelebA [LLWT15], AnimalFaces [LHM+19], Animals (containing carnivorous animals), ImageNet [DDS+09], and ColorMNIST, which is an augmented version of the MNIST dataset [LCB98], where both background and foreground have random, independent colors. Evaluation details follow in Sect. 4.5.

4.1 Comparison to Existing Methods

A key insight of our chapter is that reconstructions from a given model's representation $\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x})$ are impossible if the invariances the model has learned are not considered. In Fig. 5, we compare our approach to existing methods that either try to reconstruct the image via gradient-based optimization [MV16] or by training a reconstruction network directly on the representations $\boldsymbol{z}$ [DB16]. By conditionally sampling images $\hat{\boldsymbol{x}} = \boldsymbol{D}(\hat{\boldsymbol{z}})$, where we obtain $\hat{\boldsymbol{z}}$ via the INN $\boldsymbol{t}$ as described in (2) based on the invariances $\boldsymbol{v} \sim p(\boldsymbol{v}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, we bypass this shortcoming and obtain natural images without artifacts for any layer depth. The increased image quality is further confirmed by the Fréchet inception distance (FID) scores [HRU+17] reported in Table 4.

4.2 Understanding Models

Interpreting a face recognition model: `FaceNet` [SKP15] is a widely accepted baseline in the field of face recognition. This model embeds input images of human faces into a latent space where similar images have a small L_2-distance. We aim to understand the process of face recognition within this model by analyzing and visualizing learned invariances for several layers explicitly; see Table 6 for a detailed breakdown of the various layers of `FaceNet`. For the experiment, we use a pre-

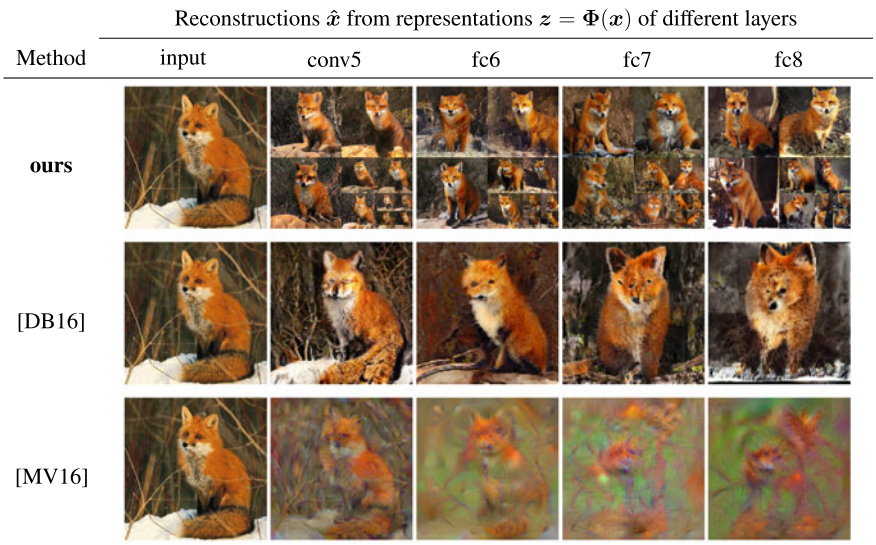


Fig. 5 Comparison to existing network inversion methods for `AlexNet` [KSH12]. In contrast to the methods of [DB16] (D&B) and [MV16] (M&V), our invertible method explicitly samples the invariances of $\mathbf{\Phi}$ w.r.t. the data, which circumvents a common cause for artifacts and produces natural images independent of the depth of the layer which is reconstructed

Table 4 FID scores for layer visualizations of `AlexNet`, obtained with our method and [DB16] (D&B). Scores are calculated on the Animals dataset

Layer	conv5	fc6	fc7	fc8	Output
ours	$\mathbf{23.6 \pm 0.5}$	$\mathbf{24.3 \pm 0.7}$	$\mathbf{24.9 \pm 0.4}$	$\mathbf{26.4 \pm 0.4}$	$\mathbf{27.4 \pm 0.3}$
D&B	25.2	**24.9**	27.2	36.1	352.6

trained `FaceNet` and train the generative model presented in (2) by conditioning on various layers. Figure 6 depicts the amount of variance present in each selected layer when generating $n = 250$ samples for each of the 100 different input images. This variance serves as a proxy for the amount of abstraction capability `FaceNet` has learned in its respective layers: more abstract representations allow for a rich variety of corresponding synthesized images, resulting in a large variance in image space when being decoded. We observe an approximate exponential growth of learned invariances with increasing layer depth, suggesting abstraction mainly happens in the deepest layers of the network. Furthermore, we are able to synthesize images that correspond to the given model representation for each selected layer.

How does relevance of different concepts emerge during training? Humans tend to provide explanations of entities by describing them in terms of their semantics, e.g., size or color. In a similar fashion, we want to semantically understand how a network (here: `SqueezeNet`) learns to solve a given problem.

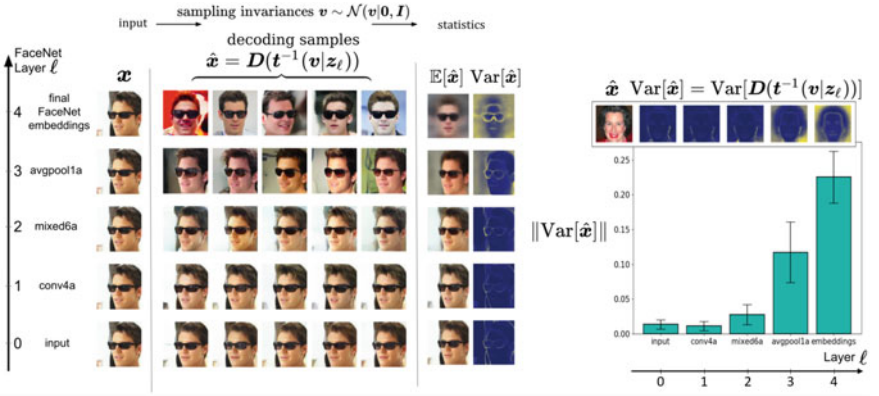


Fig. 6 Left: Visualizing `FaceNet` representations and their invariances. Sampling multiple reconstructions $\hat{\boldsymbol{x}} = \boldsymbol{D}(\boldsymbol{t}^{-1}(\boldsymbol{v}|\boldsymbol{z}_\ell))$ shows the degree of invariance learned by different layers ℓ. The invariance w.r.t. pose increases for deeper layers as expected for face identification. Surprisingly, `FaceNet` uses glasses as an identity feature throughout all its layers as evident from the spatial mean and variance plots, where the glasses are still visible. This reveals a bias and weakness of the model. Right: spatially averaged variances over multiple $\boldsymbol{x}$ and layers

Intuitively, a network should, for example, be able to solve a given classification problem by focusing on the relevant information while discarding task-irrelevant information. To build on this intuition, we construct a toy problem: digit classification on ColorMNIST. We expect the model to ignore both the random background and foreground colors of the input data, as it does not help making a classification decision. Thus, we apply the invertible approach presented in Sect. 3.2 and recover three distinct factors: `digit class`, `background color`, and `foreground color`. To capture the semantic changes occurring over the course of training of this classifier, we couple 20 instances of the invertible interpretation model on the last convolutional layer, each representing a checkpoint between iteration 0 and iteration 40000 (equally distributed). The result is shown in Fig. 7: we see that the `digit` factor becomes increasingly more relevant, with its relevance being strongly correlated to the accuracy of the model.

4.3 Effects of Data Shifts on Models

This section investigates the effects that altering input data has on the model we want to understand. We examine these effects by manipulating input data through adversarial attacks or image stylization.

How do adversarial attacks affect network representations? Here, we experiment with the fast gradient sign method (FGSM) [GSS15], which manipulates the input image by maximizing the objective of a given classification model. To understand

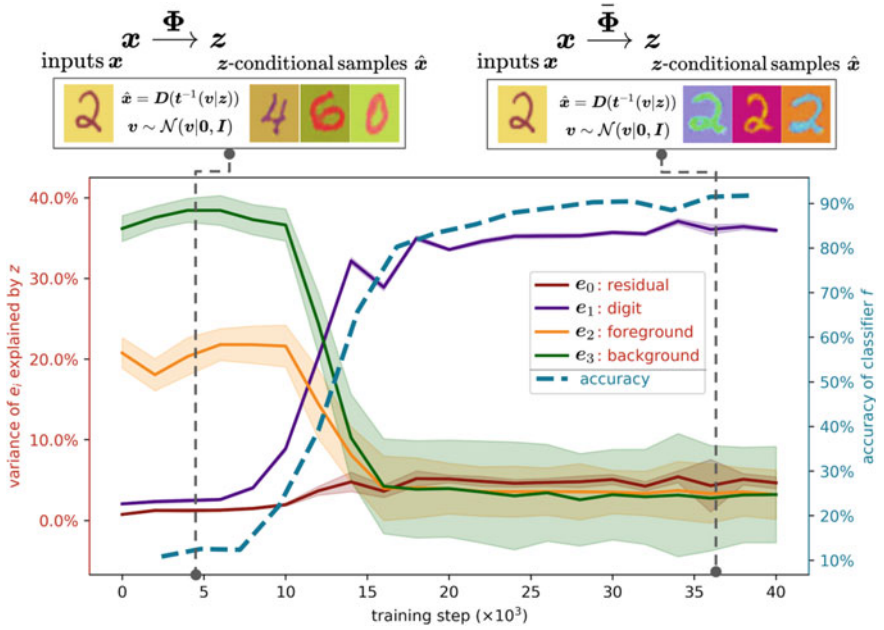


Fig. 7 Analyzing the degree to which different semantic concepts are captured by a network representation changes as training progresses. For `SqueezeNet` on ColorMNIST, we measure how much the data varies in different semantic concepts $\boldsymbol{e}_i$ and how much of this variability is captured by $\boldsymbol{z}$ at different training iterations. Early on, $\boldsymbol{z}$ is sensitive to foreground and background color, and later on it learns to focus on the digit attribute. The ability to encode this semantic concept is proportional to the classification accuracy achieved by $\boldsymbol{z}$. At training iterations 4k and 36k, we apply our method to visualize model representations and thereby illustrate how their content changes during training

how such an attack modifies representations of a given model, we first compute the image's invariances with respect to the model as $\boldsymbol{v} = \boldsymbol{t}(\boldsymbol{E}(\boldsymbol{x})|\boldsymbol{\Phi}(\boldsymbol{x}))$. For an attacked image $\boldsymbol{x}^*$, we then compute the attacked representation as $z^* = \boldsymbol{\Phi}(\boldsymbol{x}^*)$. Decoding this representation with the original invariance $\boldsymbol{v}$ allows us to precisely visualize what the adversarial attack changed. This decoding, $\hat{\boldsymbol{x}}^* = \boldsymbol{D}(\boldsymbol{t}(\boldsymbol{v}|\boldsymbol{z}^*))$, is shown in Fig. 8. We observe that, over layers of the network, the adversarial attack gradually changes the representation toward its target. Its ability to do so is strongly correlated with the amount of invariances, quantified as the total variance explained by $\boldsymbol{v}$, for a given layer as also observed in [JBZB19].

How does training on different data affect the model? Geirhos et al. [GRM+19] proposed the hypothesis that classification networks based on convolutional blocks mainly focus on texture patterns to obtain class probabilities. We further validate this hypothesis by training our invertible network $\boldsymbol{t}$ conditioned on pre-logits $\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x})$ (i.e., the penultimate layer) of two `ResNet-50` realizations. As shown in Fig. 9, a `ResNet` architecture trained on standard ImageNet is susceptible to the

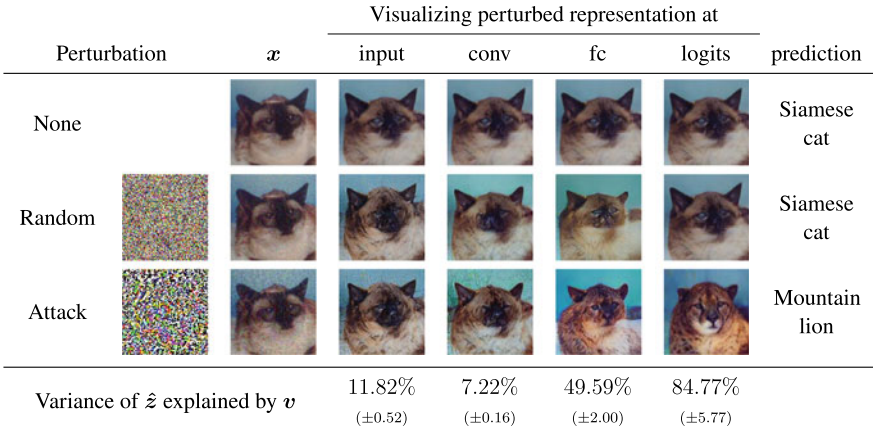


Fig. 8 Visualizing FGSM adversarial attacks on `ResNet-101`. To the human eye, the original image and its attacked version are almost indistinguishable. However, the input image is correctly classified as "Siamese cat", while the attacked version is classified as "mountain lion". Our approach visualizes how the attack spreads throughout the network. Reconstructions of representations of attacked images demonstrate that the attack targets the semantic content of deep layers. The variance of $\hat{z}$ explained by v combined with these visualizations shows how increasing invariances cause vulnerability to adversarial attacks

so-called "texture-bias", as samples generated conditioned on representation of pure texture images consistently show valid images of corresponding input classes. We furthermore visualize that this behavior can indeed be removed by training the same architecture on a stylized version of ImageNet[1]; the classifier does focus on shape. Rows 10–12 of Fig. 9 show that the proposed approach can be used to generate sketch-based content with the texture-agnostic network.

4.4 Modifying Representations

Invertible access to semantic concepts enables targeted modifications of representations $\hat{z}$. In combination with a decoder for $\hat{z}$, we obtain semantic image editing capabilities. We provide an example in Fig. 10, where we modify the factors `hair color`, `glasses`, `gender`, `beard`, `age`, and `smile`. We infer $\hat{z} = \boldsymbol{E}(\boldsymbol{x})$ from an input image. Our semantic INN $\boldsymbol{e}$ then translates this representation into semantic factors $(\boldsymbol{e}_i)_i = \boldsymbol{e}(\hat{z})$, where individual semantic concepts can be modified independently via the corresponding factor $\boldsymbol{e}_i$. In particular, we can replace each factor with that from another image, effectively transferring semantics from one representation onto another. Due to the invertibility of $\boldsymbol{e}$, the modified representation can be trans-

[1] We used weights available at https://github.com/rgeirhos/texture-vs-shape.

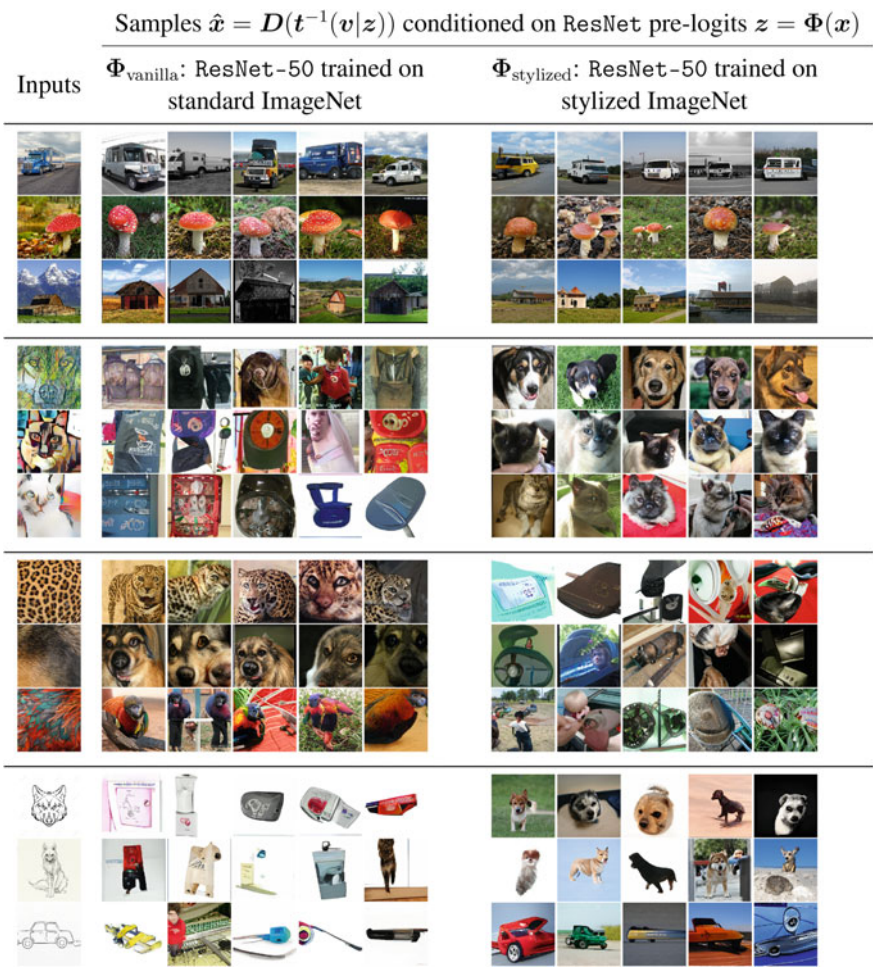


Fig. 9 Revealing texture bias in ImageNet classifiers. We compare visualizations of z from the penultimate layer of `ResNet-50` trained on standard ImageNet (left) and a stylized version of ImageNet (right). On natural images (rows 1–3) both models recognize the input, removing textures through stylization (rows 4–6) makes images unrecognizable to the standard model, however it recognizes objects from textured patches (rows 7–9). Rows 10–12 show that a model without texture bias can be used for sketch-to-image synthesis

lated back into the space of the autoencoder and is readily decoded to a modified image $\boldsymbol{x}^*$.

To observe which semantic concepts `FaceNet` is sensitive to, we compute the average distance $\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}^*)\|$ between its embeddings of $\boldsymbol{x}$ and semantically modified $\boldsymbol{x}^*$ over the test set (last row in Fig. 10). Evidently, `FaceNet` is particularly sensitive to differences in gender and glasses. The latter suggests a failure of `FaceNet` to identify persons correctly after they put on glasses.

Mean embedding Distance (± std)	0.872 (±0.048)	1.000 (±0.046)	1.061 (±0.030)	0.803 (±0.041)	0.874 (±0.053)	0.833 (±0.034)

Fig. 10 Semantic modifications on CelebA. For each column, after inferring the semantic factors $(\boldsymbol{e}_i)_i = \boldsymbol{e}(\boldsymbol{E}(\boldsymbol{x}))$ of the input $\boldsymbol{x}$, we replace one factor $\boldsymbol{e}_i$ by that from another randomly chosen image that differs in this concept. The inverse of $\boldsymbol{e}$ translates this semantic change back into a modified $\hat{\boldsymbol{z}}$, which is decoded to a semantically modified image. Distances between `FaceNet` embeddings before and after modification demonstrate its sensitivity to differences in gender and glasses (see also Fig. 6)

4.5 Evaluation Details

Here, we provide additional details on the investigated neural networks in Sect. 4 and present a way to quantify the amount of invariances in those networks. This section is only for completeness. Similar to Sect. 3.3, this section can be skipped for readers who are interested in the higher level concepts of our approach rather than its technical details. An overview of INN hyperparameters for all experiments is provided in Table 5.

Table 5 Hyperparameters of INNs for each experiment. Parameter n_{flow} denotes the number of invertible blocks within in the model, see Fig. 2. Parameters h_w and h_d refer to the width and depth of the fully connected subnetworks $\boldsymbol{s}_i$ and $\boldsymbol{\tau}_i$, respectively

Experiment	INN	Input dim.	n_{flow}	h_w	h_d
Comparison Sect. 4.1	$\boldsymbol{t}$	128	20	1024	2
Understanding models: `FaceNet`, Sect. 4.2	$\boldsymbol{t}$	128	20	512	2
Understanding models: `FaceNet`, Sect. 4.2	$\boldsymbol{e}$	128	12	512	2
Data effects: Adversarial attack, Sect. 4.3	$\boldsymbol{t}$	128	20	1024	2
Data effects: Texture bias, Sect. 4.3	$\boldsymbol{t}$	268	20	1024	2
Modifications: `FaceNet` & CelebA Sect. 4.4	$\boldsymbol{e}$	128	12	512	2

Table 6 High-level architectures of `FaceNet` and `ResNet`, depicted as `pytorch`-modules. Layers investigated in our experiments are marked in bold. Spatial sizes are provided as a visual aid and vary from model to model in our experiments. If not stated otherwise, we always extract from the *last* layer in a series of blocks (e.g., on the right: 23× **BottleNeck down** $\rightarrow \mathbb{R}^{8\times8\times1024}$ refers to the last module in the series of 23 blocks)

`FaceNet`: Implementations of layers `Mixed, Block35, Block17, Block8` can be found at `https://github.com/timesler/facenet-pytorch`. In Section 4.2, the representation from the **2nd** convolutional layers are extracted. Furthermore, BN and AdaAvgPool refer to batch normalization and adaptive average pooling, respectively. The term "down" denotes downsampling by factor 2.

RGB image $\boldsymbol{x} \in \mathbb{R}^{128\times128\times3}$
3× Conv, BN, ReLU $\rightarrow \mathbb{R}^{61\times61\times64}$
MaxPool $\rightarrow \mathbb{R}^{30\times30\times64}$
3× **Conv**, BN, ReLU $\rightarrow \mathbb{R}^{13\times13\times256}$
5× Block35 $\rightarrow \mathbb{R}^{13\times13\times256}$
Mixed down $\rightarrow \mathbb{R}^{6\times6\times896}$
10× Block17 $\rightarrow \mathbb{R}^{6\times6\times896}$
Mixed down $\rightarrow \mathbb{R}^{2\times2\times1792}$
5× Block8 $\rightarrow \mathbb{R}^{2\times2\times1792}$
AdaAvgPool $\rightarrow \mathbb{R}^{1\times1\times1792}$
Dropout, Linear, BN $\rightarrow \mathbb{R}^{512}$
identity embedding $\rightarrow \mathbb{R}^{512}$

`ResNet-101`: See `https://pytorch.org/docs/stable/torchvision/models.html` for details on other variants of `ResNet`. Bottleneck, FC, and Norm refer to bottleneck residual unit, fully connected layer, and batch normalization, respectively. The term "down" denotes downsampling by factor 2.

RGB image $\boldsymbol{x} \in \mathbb{R}^{128\times128\times3}$
Conv down $\rightarrow \mathbb{R}^{64\times64\times64}$
Norm, ReLU, **MaxPool** $\rightarrow \mathbb{R}^{32\times32\times64}$
3× BottleNeck $\rightarrow \mathbb{R}^{32\times32\times256}$
4× BottleNeck down $\rightarrow \mathbb{R}^{16\times16\times512}$
23× **BottleNeck down** $\rightarrow \mathbb{R}^{8\times8\times1024}$
3× BottleNeck down $\rightarrow \mathbb{R}^{4\times4\times2048}$
AvgPool, FC
output $\rightarrow \mathbb{R}^{1000}$

Throughout our experiments, we interpret four different models: `SqueezeNet`, `AlexNet`, `ResNet`, and `FaceNet`. Summaries of each of model's architecture are provided in Table 6 and Table 7. Implementations and pretrained weights of these models are taken from:

- `SqueezeNet (1.1)` https://pytorch.org/docs/stable/_modules/torchvision/models/squeezenet;
- `ResNet:` https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html;
- `AlexNet:` https://pytorch.org/docs/stable/_modules/torchvision/models/alexnet.html;
- `FaceNet:` https://github.com/timesler/facenet-pytorch.

Explained variance: To quantify the amount of invariances and semantic concepts, we use the fraction of the total variance explained by invariances (Fig. 8) and the fraction of the variance of a semantic concept explained by the model representation (Fig. 7).

Table 7 High-level architectures of `SqueezeNet` and `AlexNet`, depicted as `pytorch`-modules. See Table 6 for further details

`SqueezeNet`: We extract the penultimate Fire block for interpretation in Section 4.2. AdaAvgPool and Fire refer to adaptive average pooling and the Fire module introduced in [IHM+16].

RGB image $\boldsymbol{x} \in \mathbb{R}^{128\times128\times3}$
Conv, ReLU, MaxPool $\rightarrow \mathbb{R}^{31\times31\times64}$
$2\times$ Fire $\rightarrow \mathbb{R}^{31\times31\times128}$
MaxPool $\rightarrow \mathbb{R}^{15\times15\times128}$
$2\times$ Fire $\rightarrow \mathbb{R}^{15\times15\times256}$
MaxPool $\rightarrow \mathbb{R}^{7\times7\times256}$
$4\times$ **Fire** $\rightarrow \mathbb{R}^{7\times7\times512}$
Dropout, Conv, ReLU $\rightarrow \mathbb{R}^{7\times7\times1000}$
AdaAvgPool $\rightarrow \mathbb{R}^{7\times7\times1000}$
output $\rightarrow \mathbb{R}^{1000}$

`AlexNet`: The first convolution uses kernel size 11. AdaAvgPool, Flatten, and Linear refer to adaptive average pooling, reshaping the three-dimensional representation of the input to a vector, and fully-connected layer, respectively.

RGB image $\boldsymbol{x} \in \mathbb{R}^{128\times128\times3}$
Conv, ReLU, MaxPool $\rightarrow \mathbb{R}^{15\times15\times64}$
Conv, ReLU, MaxPool $\rightarrow \mathbb{R}^{7\times7\times192}$
Conv, ReLU $\rightarrow \mathbb{R}^{7\times7\times384}$
$2\times$ **Conv**, ReLU $\rightarrow \mathbb{R}^{7\times7\times256}$
MaxPool $\rightarrow \mathbb{R}^{3\times3\times256}$
AdaAvgPool, Flatten $\rightarrow \mathbb{R}^{9216}$
Dropout, **Linear**, ReLU $\rightarrow \mathbb{R}^{4096}$
Dropout, **Linear**, ReLU $\rightarrow \mathbb{R}^{4096}$
Linear $\rightarrow \mathbb{R}^{1000}$

Using the INN $\boldsymbol{t}$, we can consider $\hat{\boldsymbol{z}} = \boldsymbol{t}^{-1}(\boldsymbol{v}|\boldsymbol{z})$ as a function of $\boldsymbol{v}$ and $\boldsymbol{z}$. The total variance of $\hat{\boldsymbol{z}}$ is then obtained by sampling $\boldsymbol{v}$, via its prior which is a standard normal distribution, and $\boldsymbol{z}$, via $\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x})$ with $\boldsymbol{x} \sim p_{\text{valid}}(\boldsymbol{x})$ sampled from a validation set. We compare this total variance to the average variance obtained when sampling $\boldsymbol{v}$ for a given $\boldsymbol{z}$ to obtain the fraction of the total variance explained by invariances:

$$\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{valid}}(\boldsymbol{x}')}\left[\frac{\text{Var}_{\boldsymbol{v}\sim\mathcal{N}(\boldsymbol{v}|\boldsymbol{0},\boldsymbol{I})}\ \boldsymbol{t}^{-1}(\boldsymbol{v}|\boldsymbol{\Phi}(\boldsymbol{x}'))}{\text{Var}_{\substack{\boldsymbol{x}\sim p_{\text{valid}}(\boldsymbol{x})\\ \boldsymbol{v}\sim\mathcal{N}(\boldsymbol{v}|\boldsymbol{0},\boldsymbol{I})}}\ \boldsymbol{t}^{-1}(\boldsymbol{v}|\boldsymbol{\Phi}(\boldsymbol{x}))}\right]. \tag{20}$$

In combination with the INN $\boldsymbol{e}$, which transform $\hat{\boldsymbol{z}}$ to semantically meaningful factors, we can analyze the semantic content of a model representation $\boldsymbol{z}$. To analyze how much of a semantic concept represented by factor $\boldsymbol{e}_i$ is captured by $\boldsymbol{z}$, we use $\boldsymbol{e}$ to transform $\hat{\boldsymbol{z}}$ into $\boldsymbol{e}_i$ and measure its variance. To measure how much the semantic concept is explained by $\boldsymbol{z}$, we simply swap the roles of $\boldsymbol{z}$ and $\boldsymbol{v}$ in (20), to obtain

$$\mathbb{E}_{\boldsymbol{v}' \sim \mathcal{N}(\boldsymbol{v}'|0,\boldsymbol{I})} \left[\frac{\mathrm{Var}_{\boldsymbol{x} \sim p_{\mathrm{valid}}(\boldsymbol{x})} \; \boldsymbol{e}(\boldsymbol{t}^{-1}(\boldsymbol{v}'|\boldsymbol{\Phi}(\boldsymbol{x})))_i}{\mathrm{Var}_{\substack{\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{v}|\boldsymbol{0},\,\boldsymbol{I}) \\ \boldsymbol{x} \sim p_{\mathrm{valid}}(\boldsymbol{x})}} \; \boldsymbol{e}(\boldsymbol{t}^{-1}(\boldsymbol{v}|\boldsymbol{\Phi}(\boldsymbol{x})))_i} \right]. \tag{21}$$

Figure 8 reports (20) and its standard error when evaluated via 10k samples, and Fig. 7 reports (21) and its standard error when evaluated via 10k samples.

Conclusions

Understanding a representation in terms of both its semantics and learned invariances is crucial for interpretation of deep networks. We presented an approach *(i)* to recover the invariances a model has learned and *(ii)* to translate the representation and its invariances onto an equally expressive yet semantically accessible encoding. Our diagnostic method is applicable in a plug-and-play fashion on top of existing deep models with no need to alter or retrain them. Since our translation onto semantic factors is bijective, it loses no information and also allows for semantic modifications. Moreover, recovering invariances probabilistically guarantees that we can correctly visualize representations and sample them without leaving the underlying distribution, which is a common cause for artifacts. Altogether, our approach constitutes a powerful, widely applicable diagnostic pipeline for explaining deep representations.

References

[AKW+19] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E.W. Pellegrini, R.S. Klessen, L. Maier-Hein, C. Rother, U. Köthe, Analyzing inverse problems with invertible neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–20

[AS18] Alessandro Achille, Stefano Soatto, Emergence of invariance and disentanglement in deep representations. J. Mach. Learn. Res. (JMLR) **19**(50), 1–34 (2018)

[BBM+15] Sebastian Bach, Alexander Binder, Grégoire. Montavon, Frederick Klauschen, Klaus-Robert. Müller, Wojciech Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLOS ONE **10**(7), 1–46 (2015)

[BDS19] A. Brock, J. Donahue, K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–35

[BMDO21a] A. Blattmann, T. Milbich, M. Dorkenwald, B. Ommer, Behavior-driven synthesis of human dynamics, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2021), pp. 12236–12246

[BMDO21b] A. Blattmann, T. Milbich, M. Dorkenwald, B. Ommer, iPOKE: poking a still image for controlled stochastic video synthesis, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, virtual conference (2021), pp. 14707–14717

[BZK+17] D. Bau, B. Zhou, A. Khosla, A. Oliva, A. Torralba, Network dissection: quantifying interpretability of deep visual representations, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI, USA, 2017), pp. 6541–6549

[BZS+19] D. Bau, J-Y. Zhu, H. Strobelt, B. Zhou, J.B. Tenenbaum, W.T. Freeman, A. Torralba, GAN dissection: visualizing and understanding generative adversarial networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–18

[CSX+18] Q. Cao, L. Shen, W. Xie, O.M. Parkhi, A. Zisserman, VGGFace2: a dataset for recognising faces across pose and age, in *Proceedings of the IEEE International Conference on Automatic Face & Gesture Recognition (FG)* (Xi'an, China, 2018), pp. 67–74

[CWG+18] S.A. Cadena, M.A. Weis, L.A. Gatys, M. Bethge, A.S. Ecker, Diverse feature visualizations reveal invariances in early layers of deep neural networks, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Munich, Germany, 2018), pp. 225–240

[DB16] A. Dosovitskiy, T. Brox, Generating images with perceptual similarity metrics based on deep networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Barcelona, Spain, 2016), pp. 658–666

[DDS+09] J. Deng, W. Dong, R. Socher, L-J. Li, K. Li, F-F. Li, ImageNet: a large-scale hierarchical image database, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Miami, FL, USA, 2009), pp. 248–255

[DMB+21] M. Dorkenwald, T. Milbich, A. Blattmann, R. Rombach, K.G. Derpanis, B. Ommer, Stochastic image-to-video synthesis using cINNs, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2021), pp. 3742–3753

[DSB17] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real NVP, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Toulon, France, 2017), pp. 1–32

[DW19] B. Dai, D. Wipf, Diagnosing and enhancing VAE models, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–12

[EHO19] P. Esser, J. Haux, B. Ommer, Unsupervised robust disentangling of latent characteristics for image synthesis, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, 2019), pp. 2699–2709

[ERO20] P. Esser, R. Rombach, B. Ommer, A disentangling invertible interpretation network for explaining latent representations, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2020), pp. 9223–9232

[Eur20] European Commission, White Paper on Artificial Intelligence – A European Approach to Excellence and Trust. Technical report, European Union (2020). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2020:65:FIN

[FV18] R. Fong, A. Vedaldi, Net2Vec: quantifying and explaining how concepts are encoded by filters in deep neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT, USA, 2018), pp. 8730–8738

[GAOI19] L. Goetschalckx, A. Andonian, A. Oliva, P. Isola, GANalyze: toward visual definitions of cognitive image properties, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, 2019), pp. 5744–5753

[GF17] Bryce Goodman, Seth Flaxman, European union regulations on algorithmic decision-making and a "right to explanation". AI Mag. **38**(3), 50–57 (2017)

[GRM+19] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F.A. Wichmann, W. Brendel, ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–22

[GSS15] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in *Proceedings of the International Conference on Learning Representations (ICLR)* (San Diego, CA, USA, 2015), pp. 1–11

[HRU+17] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, GANs trained by a two time-scale update rule converge to a local nash equilibrium, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Long Beach, CA, USA, 2017), pp. 6626–6637

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 770–778

[IHM+16] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, SqueezeNet: AlexNet-Level Accuracy With 50x Fewer Parameters and < 0.5 MB Model Size (2016) pp. 1–13. arXiv:1602.07360

[JBZB19] J-H. Jacobsen, J. Behrmann, R. Zemel, M. Bethge, Excessive invariance causes adversarial vulnerability, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–17

[KD18] D.P. Kingma, P. Dhariwal, Glow: generative flow with invertible 1x1 convolutions, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, 2018), pp. 10236–10245

[KSH12] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Lake Tahoe, NV, USA, 2012), pp. 1106–1114

[KSLO19] D. Kotovenko, A. Sanakoyeu, S. Lang, B. Ommer, Content and style disentanglement for artistic style transfer, in:*Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, 2019), pp. 4422–4431

[KW14] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Banff, AB, Canada, 2014), pp. 1–14

[KWKT15] T.D. Kulkarni, W. Whitney, P. Kohli, J.B. Tenenbaum, Deep convolutional inverse graphics network, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, 2015), pp. 2539–2547

[LBL+19] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, O. Bachem, Challenging common assumptions in the unsupervised learning of disentangled representations, in *Proceedings of the International Conference on Machine Learning (ICML)* (Long Beach, CA, USA, 2019), pp. 4114–4124

[LBMO19] D. Lorenz, L. Bereska, T. Milbich, B. Ommer, Unsupervised part-based disentangling of object shape and appearance, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, 2019), pp. 10955–10964

[LCB98] Y. LeCun, C. Cortes, C.J.C. Burges, The MNIST Database of Handwritten Digits (1998) Retrieved from 2021-11-18

[LeC12] Y. LeCun, Learning invariant feature hierarchies, in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops* (Firenze, Italy, 2012), pp. 496–505

[LHM+19] M-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, J. Kautz, Few-shot unsupervised image-to-image translation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, 2019), pp. 10551–10560

[Lip18] Z.C. Lipton, The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery. ACM Queue **16**(3), 31–57 (2018)

[LLWT15] Z. Liu, P. Luo, X. Wang, X. Tang, Deep learning face attributes in the wild, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Santiago, Chile, 2015), pp. 3730–3738

[LSOL20] Y. Li, K. Kumar Singh, U. Ojha, Y.J. Lee, MixNMatch: multifactor disentanglement and encoding for conditional image generation, in *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), virtual conference (2020), pp. 8039–8048

[Mil19] Tim Miller, Explanation in artificial intelligence: insights from the social sciences. Artif. Intell. **267**, 1–38 (2019)

[MLB+17] Grégoire. Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, Klaus-Robert Müller, Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recognit. **65**(5), 211–222 (2017)

[MOT15] A. Mordvintsev, C. Olah, M. Tyka, Inceptionism: going deeper into neural networks (2015). Accessed 2021-11-18

[MSM18] Grégoire. Montavon, Wojciech Samek, Klaus-Robert. Müller, Methods for interpreting and understanding deep neural networks. Digital Signal Process. **73**, 1–15 (2018)

[MV16] Aravindh Mahendran, Andrea Vedaldi, Visualizing deep convolutional neural networks using natural pre-images. Int. J. Comput. Vis. (IJCV) **120**, 233–255 (2016)

[NDY+16] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, J. Clune, Synthesizing the preferred inputs for neurons in neural networks via deep generator networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Barcelona, Spain, 2016), pp. 3387–3395

[NKW19] C. Nash, N. Kushman, C.K.I. Williams, Inverting supervised representations with autoregressive neural density models, in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)* (Naha, Japan, 2019), pp. 1620–1629

[PASC+20] G. Plumb, M. Al-Shedivat, Á.A. Cabrera, A. Perer, E. Xing, A. Talwalkar, Regularizing black-box models for improved interpretability, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, virtual conference (2020), pp. 10526–10536

[Red93] A. Norman Redlich, Supervised factorial learning. Neural Comput. **5**(5), 750–766 (1993)

[REO20] R. Rombach, P. Esser, B. Ommer, Network-to-network translation with conditional invertible neural networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)*, virtual conference (2020), pp. 2784–2797

[RMW14] D.J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Banff, AB, Canada, 2014), pp. 1–14

[SCD+20] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra, Grad-CAM: visual explanations from deep networks via gradient-based localization. Int. J. Comput. Vis. (IJCV) **128**, 336–359 (2020)

[SGM+20] A. Shocher, Y. Gandelsman, I. Mosseri, M. Yarom, M. Irani, W.T. Freeman, T. Dekel, Semantic pyramid for image generation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2020), pp. 7457–7466

[SIT+19] S. Santurkar, A. Ilyas, D. Tsipras, L. Engstrom, B. Tran, A. Madry, Image synthesis with a single (robust) classifier, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Vancouver, BC, Canada, 2019), pp. 1260–1271

[SKP15] F. Schroff, D. Kalenichenko, J. Philbin, FaceNet: a unified embedding for face recognition and clustering, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, MA, USA, 2015), pp. 815–823

[SR15] M. Simon, E. Rodner, Neural activation constellations: unsupervised part model discovery with convolutional networks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Santiago, Chile, 2015), pp. 1143–1151

[SRD14] M. Simon, E. Rodner, J. Denzler, Part detector discovery in deep convolutional neural networks, in *Proceedings of the Asian Conference on Computer Vision (ACCV)* (Singapore, Singapore, 2014), pp. 162–177

[SVZ14] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: visualising image classification models and saliency maps, in *Proceedings of the International Conference on Learning Representations (ICLR) Workshops* (Banff, AB, Canada, 2014), pp. 1–8

[SWM17] W. Samek, T. Wiegand, K-R. Müller, *Explainable Artificial Intelligence: Understanding*. Visualizing and Interpreting Deep Learning Models (Springer, 2017)

[SZS+14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Banff, AB, Canada, 2014), pp. 1–10

[UGP+17] P. Upchurch, J. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, K. Weinberger, Deep feature interpolation for image content changes, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI, USA, 2017), pp. 7064–7073

[XYA19] Z. Xiao, Q. Yan, Y. Amit, Generative Latent Flow (2019), pp. 1–18. arXiv:1905.10485

[YCN+15] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, H. Lipson, Understanding neural networks through deep visualization, in *Proceedings of the International Conference on Machine Learning (ICML) Workshops* (2015), pp. 1–12

[ZF14] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Zurich, Switzerland, 2014), pp. 818–833

[ZIE+18] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT, USA, 2018), pp. 586–595

[ZKL+15] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Object detectors emerge in deep scene CNNs, in *Proceedings of the International Conference on Learning Representations (ICLR)* (San Diego, CA, USA, 2015), pp. 1–12

[ZKL+16] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 2921–2929

[ZNWZ18] Q. Zhang, Y. Nian Wu, S-C. Zhu, Interpretable convolutional neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT, USA, 2018), pp. 8827–8836

Confidence Calibration for Object Detection and Segmentation

Fabian Küppers, Anselm Haselhoff, Jan Kronenberger, and Jonas Schneider

Abstract Calibrated confidence estimates obtained from neural networks are crucial, particularly for safety-critical applications such as autonomous driving or medical image diagnosis. However, although the task of confidence calibration has been investigated on classification problems, thorough investigations on object detection and segmentation problems are still missing. Therefore, we focus on the investigation of confidence calibration for object detection and segmentation models in this chapter. We introduce the concept of multivariate confidence calibration that is an extension of well-known calibration methods to the task of object detection and segmentation. This allows for an extended confidence calibration that is also aware of additional features such as bounding box/pixel position and shape information. Furthermore, we extend the expected calibration error (ECE) to measure miscalibration of object detection and segmentation models. We examine several network architectures on MS COCO as well as on Cityscapes and show that especially object detection as well as instance segmentation models are intrinsically miscalibrated given the introduced definition of calibration. Using our proposed calibration methods, we have been able to improve calibration so that it also has a positive impact on the quality of segmentation masks as well.

F. Küppers (✉) · A. Haselhoff · J. Kronenberger
Hochschule Ruhr West, Duisburger Str. 100, 45479 Mülheim a.d. Ruhr, Germany
e-mail: fabian.kueppers@hs-ruhrwest.de

A. Haselhoff
e-mail: anselm.haselhoff@hs-ruhrwest.de

J. Kronenberger
e-mail: jan.kronenberger@hs-ruhrwest.de

J. Schneider
Elektronische Fahrwerksysteme GmbH, Dr.-Ludwig-Kraus-Str. 6,
85080 Gaimersheim, Germany
e-mail: jonas.schneider@efs-auto.com

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_8

1 Introduction

Common neural networks for object detection must not only determine the position and class of an object but also denote their confidence about the correctness of each detection. This also holds for instance or semantic segmentation models that output a score for each pixel indicating the confidence of object mask membership. A reliable confidence estimate for detected objects is crucial, particularly for safety-critical applications such as autonomous driving, to reliably process detected objects. Another example is medical diagnosis, where, for example, the shape of a brain tumor within an MRI image is of special interest [MWT+20].

Each confidence estimate might be seen as a probability of correctness, reflecting the model's uncertainty about a detection or the pixel mask. During inference, we expect the estimated confidence to match the observed precision for a prediction. For example, given 100 predictions with 80% confidence each, we expect 80 predictions to be correctly predicted [GPSW17, KKSH20]. However, recent work has shown that the confidence estimates of either classification or detection models based on neural networks are *miscalibrated*, i.e., the confidence does not match the observed accuracy in classification [GPSW17] or the observed precision in object detection [KKSH20]. While confidence calibration within the scope of classification has been extensively investigated [NCH15, NC16, GPSW17, KSFF17, KPNK+19], we recently defined the term of calibration for object detection and proposed methods to measure and resolve miscalibration [KKSH20, SKR+21]. In this context, we measured miscalibration w.r.t. the position and scale of detected objects by also including the regression branch of an object detector into a calibration mapping. We have been able to show that modern object detection models also tend to be miscalibrated. On the one hand, this can be mitigated using standard calibration methods. On the other hand, we show that our proposed methods for position- and shape-dependent calibration [KKSH20] are able to further reduce miscalibration.

In this chapter we review our methods for position- and shape-dependent confidence calibration and provide a definition for confidence calibration for the task of *instance/semantic segmentation*. To this end, we extend the definition of the *expected calibration error* (ECE) to measure miscalibration within the scope of instance/semantic segmentation. Furthermore, we adapt the extended calibration methods originally designed for object detection [KKSH20] to enable position-dependent confidence calibration for segmentation tasks as well.

This chapter is structured as follows. In Sect. 2 we review the most important related works regarding confidence calibration. In Sect. 3 we provide the definitions of calibration for the tasks of object detection, instance segmentation, and semantic segmentation. Furthermore, in Sect. 4 we present the extended confidence calibration methods for object detection and segmentation. Extensive experimental evaluations on a variety of architectures and computer vision problems, including object detection and instance/semantic segmentation, are discussed in Sect. 5. Finally, we provide conclusions and discuss our most important findings.

2 Related Works

In the past, most research focused on measuring and resolving miscalibration for classification tasks. In this scope, the *expected calibration error* (ECE) [NCH15] is commonly used in conjunction with the Brier score and negative log likelihood (NLL) loss to measure miscalibration. For calculating the ECE, all samples are grouped into equally sized bins by their confidence. Afterward, for each bin the accuracy is calculated and used as an approximation of the accuracy of a single sample in its respective bin. Recently, several extensions such as classwise ECE [KPNK+19], marginal calibration error [KLM19], or adaptive calibration error [NDZ+19] for the evaluation of multi-class problems have also been proposed, where the ECE is evaluated for each class separately. In contrast to previous work, we extend the common ECE definition to the task of object detection [KKSH20] and instance/semantic segmentation. This extension allows for a position-dependent miscalibration evaluation so that it is possible to quantify miscalibration separately for certain image regions. The definition is given in Sect. 3.

Besides measuring miscalibration, it is also possible to resolve a potential miscalibration by using calibration methods that are applied after inference. These post-hoc calibration methods can be divided into binning and scaling methods. Binning methods such as histogram binning [ZE01], isotonic regression [ZE02], Bayesian binning into quantiles (BBQ) [NCH15], or ensemble of near-isotonic regression (ENIR) [NC16] group all samples into several bins by their confidence (similar to the ECE calculation) and perform a mapping from uncalibrated confidence estimates to calibrated ones. In contrast, scaling methods such as logistic calibration (Platt scaling) [Pla99], temperature scaling [GPSW17], beta calibration [KSFF17], or Dirichlet calibration [KPNK+19] scale the network logits before sigmoid/softmax activation to obtain calibrated confidences. The scaling parameters are commonly obtained by logistic regression. Other approaches comprise binwise temperature scaling [JJY+19] or scaling-binning calibrator [KLM19], combining both approaches to further improve calibration performance for classification tasks.

Recently, we proposed an extension of common calibration methods to object detection by also including the bounding box regression branch of an object detector into a calibration mapping [KKSH20]. On the one hand, we extended the histogram binning [ZE01] to perform calibration using a multidimensional binning scheme. On the other hand, we also extended scaling methods to include position information into a calibration mapping. Both approaches are presented in Sect. 4 in more detail.

Unlike classification, the task of instance or semantic segmentation calibration has not yet been addressed by many authors. In the work of [WLK+20], the authors perform online confidence calibration of the classification head of an instance segmentation model and show that this has a significant impact on the mask quality. The authors in [KG20] use a multi-task learning approach for semantic segmentation models to improve model calibration and out-of-distribution detection within the scope of medical image diagnosis. A related approach is proposed by [MWT+20], where the authors train multiple fully-connected networks as an ensemble to obtain

well-calibrated semantic segmentation models. However, none of these methods provide an explicit definition of calibration for segmentation models (instance or semantic). This problem is addressed by [DLXS20], where the authors explicitly define semantic segmentation calibration and propose a local temperature scaling for semantic image masks. This approach utilizes the well-known temperature scaling [GPSW17] and assigns a temperature for each mask pixel separately. Furthermore, they use a dedicated convolutional neural network (CNN) to infer the temperature parameters for each image. Our definition of segmentation calibration in Sect. 3 is conform with their definition. Our approach differs from their proposed image-based temperature scaling as we use a single position-dependent calibration mapping to model the probability distributions for all images.

3 Calibration Definition and Evaluation

In this section, we review the term of confidence calibration for *classification* tasks [NCH15, GPSW17] and extend this definition to *object detection*, *instance segmentation*, and *semantic segmentation*. Furthermore, we derive the *detection expected calibration error* (D-ECE) to measure miscalibration.

3.1 Definitions of Calibration

Classification: In a first step, we define the datasets $\mathcal{D}$ of size $|\mathcal{D}| = N$ with indices $i \in \mathcal{I} = \{1, \ldots, N\}$, consisting of images $\mathbf{x} \in \mathbb{I}^{H \times W \times C}$, $\mathbb{I} = [0, 1]$, with height H, width W, and number of channels C. Each image has ground-truth information that consists of the class information $\overline{Y} \in \mathcal{Y} = \{1, \ldots, Y\}$. As an introduction into the task of confidence calibration, we start with the definition of perfect calibration for classification. A classification model $\mathbf{F}_{\text{cls}}$ outputs a label $\hat{Y}$ and a corresponding confidence score $\hat{P}$ indicating its belief about the prediction's correctness. In this case, perfect calibration is defined by [GPSW17]

$$\mathrm{P}(\hat{Y} = \overline{Y} | \hat{P} = p) = p \quad \forall p \in [0, 1]. \tag{1}$$

In other words, the accuracy $\mathrm{P}(\hat{Y} = \overline{Y} | \hat{P} = p)$ for a certain confidence level p should match the estimated confidence. If we observe a deviation, a model is called *miscalibrated*. In binary classification, we rather consider the *relative frequency* of $\overline{Y} = 1$ instead of the accuracy as the calibration measure. We illustrate this difference using the following example. Consider $N = 100$ images of a dataset $\mathcal{D}$ with binary ground-truth labels $\overline{Y} \in \{0, 1\}$, where 50 images are labeled as $\overline{Y} = 0$ and 50 images with $\overline{Y} = 1$. Furthermore, consider a classification model $\mathbf{F}_{\text{cls}}$ with a sigmoidal output in $[0, 1]$, indicating its confidence for $\overline{Y} = 1$. In our example, this model is able to always predict the correct ground-truth label with confidences $\hat{P} = 0$ and $\hat{P} = 1$ for

$\hat{Y} = 0$ and $\hat{Y} = 1$, respectively. Thus, the network is able to achieve an accuracy of 100% but with an average confidence of 50%. Therefore, we consider the relative frequency for $\overline{Y} = 1$ in a binary classification task as the calibration goal that is also 50% in this scenario.

Object detection: In the next step, we extend our dataset and model to the task of object detection. In contrast to a classification dataset, an object detection dataset consists of ground-truth annotations $\overline{Y} \in \mathcal{Y} = \{1, \ldots, Y\}$ for each object within an image as well as the ground-truth position and shape information $\overline{\mathbf{R}} \in \mathcal{R} = [0, 1]^A$ (A denotes the size of the normalized box encoding, comprising the center x and y positions c_x, c_y, as well as width w and height h). An object detection model $\mathbf{F}_{\text{det}}$ further outputs a confidence score $\hat{P} \in [0, 1]$, a label $\hat{Y} \in \mathcal{Y}$, and the corresponding position $\hat{\mathbf{R}} \in \mathcal{R}$ for each detection in an image $\mathbf{x}$. Extending the original formulation for confidence calibration within classification tasks [GPSW17], perfect calibration for *object detection* is defined by [KKSH20]

$$\mathrm{P}(M = 1 | \hat{P} = p, \hat{Y} = y, \hat{\mathbf{R}} = \mathbf{r}) = p \qquad \forall p \in [0, 1], y \in \mathcal{Y}, \mathbf{r} \in \mathcal{R}, \tag{2}$$

where $M = 1$ denotes a correctly classified prediction that matches a ground-truth object with a certain intersection-over-union (IoU) score. Commonly, an object detection model is calibrated by means of its precision, as the computation of the accuracy is not possible without knowing all anchors of a model [KKSH20, SKR+21]. Thus, $\mathrm{P}(M = 1)$ is a shorthand notation for $\mathrm{p}(\hat{Y} = \overline{Y}, \hat{\mathbf{R}} = \overline{\mathbf{R}})$ that expresses the precision for a dedicated IoU threshold.

Instance segmentation: At this point, we adapt this idea to define confidence calibration for *instance segmentation*. Consider a dataset with K annotated objects $\mathcal{K}$ over all images. For notation simplicity, we further use $j \in \mathcal{J}_k = \{1, \ldots, H_k \cdot W_k\}$ as the index for pixel j within a bounding box $\overline{\mathbf{R}}_k$ of object $k \in \mathcal{K}$ in the instance segmentation dataset, where H_k and W_k denote the width and height of object k, respectively. In addition to object detection, a ground-truth dataset for instance segmentation also consists of pixel-wise mask labels denoted by $\overline{Y}_j \in \mathcal{Y}^* = \{0, 1\}$ for each pixel j in the bounding box $\overline{\mathbf{R}}_k$ of object k. Note that we introduce the star superscript (*) here to distinguish between the bounding box label/prediction encoding and the instance segmentation label/prediction encoding. An instance segmentation model $\mathbf{F}_{\text{ins}}$ predicts the membership $\hat{Y}_j \in \mathcal{Y}^*$ for each pixel j in the predicted bounding box $\hat{\mathbf{R}}_k$ to the object mask with a certain confidence $\hat{P}_j \in [0, 1]$. We further denote $\mathbf{R}_j \in \mathcal{R}^* = [0, 1]^{A^*}$ as the position of pixel j *within the bounding box* $\hat{\mathbf{R}}_k$, where A^* denotes the size of the used position encoding of a pixel within its bounding box. In contrast to object detection, it is possible to treat the confidence scores of each pixel within an instance segmentation mask as a binary classification problem. In this case, the confidence $\hat{P}_j$ can be interpreted as the probability of a pixel belonging to the object mask. Therefore, the aim of confidence calibration for *instance segmentation* is that the pixel confidence should match the relative frequency that a pixel is part

of the object mask. According to the task of object detection, we further include a position dependency into the definition, for instance, segmentation and obtain

$$\begin{aligned} \mathrm{P}(\hat{Y}_j = \overline{Y}_j | \hat{Y} = y, \hat{P}_j = p, \mathbf{R}_j = \mathbf{r}) = p, \\ \forall p \in [0, 1], y \in \mathcal{Y}, \mathbf{r} \in \mathcal{R}^*, j \in \mathcal{J}_k, k \in \mathcal{K}. \end{aligned} \tag{3}$$

The former term can be interpreted as the probability that the prediction $\hat{Y}_j$ for a pixel with index j within an object k matches the ground-truth annotation $\overline{Y}_j$ given a certain confidence p, a certain pixel position $\mathbf{r}$ within the bounding box, as well as a certain object category y that is predicted by the bounding box head of the instance segmentation model.

Semantic Segmentation: Compared to object detection or instance segmentation, a *semantic segmentation* dataset does not hold ground-truth information for individual objects but rather consists of pixel-wise class annotations $\overline{Y}_j \in \mathcal{Y} = \{1, \ldots, Y\}$, $j \in \mathcal{J} = \{1, \ldots, H \cdot W\}$. A semantic segmentation model $\mathbf{F}_{\text{sem}}$ outputs pixel-wise labels $\hat{Y}_j$ and probabilities $\hat{P}_j$ with relative position $\mathbf{R}_j$ *within an image*. Therefore, we can define perfect calibration for *semantic segmentation* as

$$\begin{aligned} \mathrm{P}(\hat{Y}_j = \overline{Y}_j | \hat{P}_j = p, \mathbf{R}_j = \mathbf{r}) = p, \\ \forall p \in [0, 1], \mathbf{r} \in \mathcal{R}^*, j \in \mathcal{J}, \end{aligned} \tag{4}$$

that is related to the calibration definition of classification [GPSW17]. In addition, the confidence score of each pixel must not only reflect its accuracy given a certain confidence level but also at a certain pixel position.

3.2 Measuring Miscalibration

We can measure the miscalibration of an *object detection* model as the expected deviation between confidence and observed precision which is also known as the *detection expected calibration error* (D-ECE) [KKSH20]. Let further $\mathbf{s} = (p, y, \mathbf{r})$ denote a single detection with confidence p, class y, and bounding box $\mathbf{r}$ so that $\mathbf{s} \in \mathcal{S}$, where $\mathcal{S}$ is the aggregated set of the confidence space $[0, 1]$, the set of ground-truth labels $\mathcal{Y}$, and the set of possible bounding box positions $\mathcal{R}$. Since M is a continuous random variable, we need to approximate the D-ECE using the Riemann-Stieltjes integral [GPSW17]

$$\mathbb{E}_{\hat{P}, \hat{Y}, \hat{\mathbf{R}}}[|\mathrm{P}(M = 1 | \hat{P} = p, \hat{Y} = y, \hat{\mathbf{R}} = \mathbf{r}) - p|] \tag{5}$$

$$= \int_{\mathcal{S}} \left| \mathrm{P}(M = 1 | \hat{P} = p, \hat{Y} = y, \hat{\mathbf{R}} = \mathbf{r}) - p \right| dF_{\hat{P}, \hat{Y}, \hat{\mathbf{R}}}(\mathbf{s}) \tag{6}$$

with $F_{\hat{P}, \hat{Y}, \hat{\mathbf{R}}}(\mathbf{s})$ as the joint cumulative distribution of $\hat{P}$, $\hat{Y}$, and $\hat{\mathbf{R}}$. Let $\mathbf{a} = (a_p, a_y, a_{r_1}, \ldots, a_{r_A})^\mathsf{T} \in \mathcal{S}$ and $\mathbf{b} = (b_p, b_y, b_{r_1}, \ldots, b_{r_A})^\mathsf{T} \in \mathcal{S}$ denote interval

boundaries for the confidence $a_p, b_p \in [0, 1]$, the estimated labels $a_y, b_y \in \mathcal{Y}$ and each quantity of the bounding box encoding $a_{r_1}, b_{r_1}, \ldots a_{r_A}, b_{r_A} \in [0, 1]$ defined on the joint cumulative $F_{\hat{P},\hat{Y},\hat{\mathbf{R}}}(\mathbf{s})$ so that[1] $F_{\hat{P},\hat{Y},\hat{\mathbf{R}}}(\mathbf{b}) - F_{\hat{P},\hat{Y},\hat{\mathbf{R}}}(\mathbf{a}) = \mathrm{P}(\hat{P} \in [a_p, b_p], \hat{Y} \in [a_y, b_y], \hat{R}_1 \in [a_{r_1}, b_{r_1}], \ldots, \hat{R}_A \in [a_{r_A}, b_{r_A}])$. The integral in (6) is now approximated by

$$\sum_{m \in \mathcal{B}} \mathrm{P}(\mathbf{s}_m) \cdot \left| \mathrm{P}(M = 1 | \hat{P} = p_m, \hat{Y} = y_m, \hat{\mathbf{R}} = \mathbf{r}_m) - p_m \right|, \tag{7}$$

with B as the number of equidistant bins used for integral approximation so that $\mathcal{B} = \{1, \ldots, B\}$, and $p_m \in [0, 1]$, $y_m \in \mathcal{Y}$ and $\mathbf{r}_m \in \mathcal{R}$ being the respective bin entities for average confidence, the current label, and the average (unpacked) bounding box scores within bin $m \in \mathcal{B}$, respectively. Let N_m denote the amount of samples within a single bin m. For large datasets $|\mathcal{D}| = N$, the probability $\mathrm{P}(M = 1 | \hat{P} = p_m, \hat{Y} = y_m, \hat{\mathbf{R}} = \mathbf{r}_m)$ is approximated by the average precision within a single bin m, whereas p_m is approximated by the average confidence, so that the D-ECE can finally be computed by

$$\sum_{m \in \mathcal{B}} \frac{1}{N_m} \left| \mathrm{prec}(m) - \mathrm{conf}(m) \right|, \tag{8}$$

where $\mathrm{prec}(m)$ and $\mathrm{conf}(m)$ denote the precision and average confidence within bin m, respectively.

Similarly, we can extend the D-ECE to *instance segmentation* by

$$\mathbb{E}_{\hat{Y},\hat{P}_j,\mathbf{R}_j}[\left| \mathrm{P}(\hat{Y}_j = \overline{Y}_j | \hat{Y} = y, \hat{P}_j = p, \mathbf{R}_j = \mathbf{r}) - p \right|] \tag{9}$$

$$\approx \sum_{m \in \mathcal{B}} \frac{1}{N_m} \left| \mathrm{freq}(m) - \mathrm{conf}(m) \right|, \tag{10}$$

with a binning scheme over all pixels and freq(m) as the average frequency within each bin. In this case, each pixel is treated as a separate prediction and binned by its confidence, label class, and relative position. In the same way, the D-ECE for *semantic segmentation* is approximated by

$$\mathbb{E}_{\hat{P}_j,\mathbf{R}_j}[\left| \mathrm{P}(\hat{Y}_j = \overline{Y}_j | \hat{P}_j = p, \mathbf{R}_j = \mathbf{r}^*) - p \right|] \tag{11}$$

$$\approx \sum_{m \in \mathcal{B}} \frac{1}{N_m} \left| \mathrm{acc}(m) - \mathrm{conf}(m) \right|, \tag{12}$$

with accuracy $\mathrm{acc}(m)$ within each bin m.

[1] For discrete random variables (e.g., Y), we use a continuous density function of the form $\mathrm{p}_Y(y) = \sum_{y^* \in \mathcal{Y}} \mathrm{P}(y^*)\delta(y - y^*)$ with $\delta(x)$ as the Dirac delta function.

For the calibration evaluation of object detection models, we can use the relative position c_x, c_y, and the shape h, w, of the bounding boxes [KKSH20]. For segmentation, we consider the relative position x, y, of each pixel within a bounding box (instance segmentation) or within the image (semantic segmentation), as well as its distance d to the next segment boundary, as we expect a higher uncertainty in the peripheral areas of a segmentation mask. We use these definitions to evaluate different models in Sect. 5.

4 Position-Dependent Confidence Calibration

For post-hoc calibration, we distinguish between binning and scaling methods. According to the approximation in (7), binning methods such as histogram binning [ZE01] divide all samples into several bins by their confidence and measure the average accuracy/precision within each bin. In contrast, scaling methods rescale the logits of a neural network before sigmoid/softmax activation to calibrate a network's output. In this section, we extend standard calibration methods so that they are capable of dealing with additional information such as position and/or shape. These extended methods can be used for confidence calibration of *object detection*, *instance segmentation*, and *semantic segmentation* tasks. We illustrate the difference between standard calibration and position-dependent calibration using an artificially created dataset in Fig. 1. This dataset consists of points with a confidence score and a binary ground-truth information in $\{0, 1\}$, so that we are able to compute the frequency of the points across the whole image. We observe that standard calibration only shifts the average confidence to fit the average precision/accuracy. This leads to an

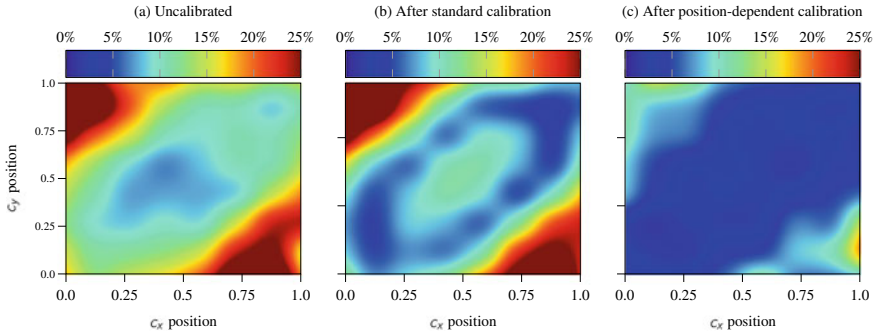


Fig. 1 **a** Consider an artificial dataset with low calibration error in the center and increasing miscalibration toward the image boundaries. **b** A common logistic calibration model only shifts the average confidence to match the accuracy. This leads to a better global ECE score but also degrades the miscalibration in the center of the image. **c** In contrast, position-dependent calibration is capable of possible dependencies and leads to an overall improvement of calibration (example taken from previous work [KKSH20])

increased calibration error in the center of the image. In contrast, position-dependent calibration as defined in this section is able to capture correlations between position information and calibration error and reduces the D-ECE across the whole image.

4.1 Histogram Binning

Given a binning scheme with B different bins so that $\mathcal{B} = \{1, \dots, B\}$ with the according bin boundaries $0 = a_1 \leq a_2 \leq \dots \leq a_{B+1} = 1$ and the corresponding calibrated estimate θ_m within each bin, the objective for histogram binning to estimate $\Theta = \{\theta_m | m \in \mathcal{B}\}$ is the minimization

$$\min_{\Theta} \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{B}} \mathbb{1}(a_m \leq \hat{p}_i < a_{m+1}) \cdot (\theta_m - \overline{y}_i)^2, \tag{13}$$

with $\mathbb{1}(\cdot)$ as the indicator function, yielding a 1 if its argument is true and a 0 if it is false [ZE01, GPSW17]. This objective converges to the fraction of positive samples within each bin under consideration. However, within the scope of object detection or segmentation, we also want to measure the accuracy/precision w.r.t. position and shape information. As an extension to the standard histogram binning [ZE01], we therefore propose a multidimensional binning scheme that divides all samples into several bins by their confidence **and** by all additional information such as position and shape [KKSH20]. We further denote $\hat{\mathbf{S}} = (\hat{P}, \hat{\mathbf{R}})$ of size $Q = A + 1$ as the input vector to a calibration function consisting of the confidence and the bounding box encoding, so that $\mathcal{Q} = \{1, \dots, Q\}$. In a multidimensional histogram binning, we indicate the number of bins as $\mathbf{B} = (B_1, \dots, B_Q)$ so that $\mathcal{B}^* = \{\mathcal{B}_q = \{1, \dots, B_q\} | q \in \mathcal{Q}\}$ with bin boundaries $\{0 = a_{1,q} \leq a_{2,q} \leq \dots \leq a_{B_q+1,q} = 1 | q \in \mathcal{Q}\}$. For each bin combination $m_1 \in \mathcal{B}_1, \dots, m_Q \in \mathcal{B}_Q$, we have a dedicated calibration parameter $\theta_{m_1,\dots,m_Q} \in \mathbb{R}$ so that the calibration parameters Θ^*, with $|\Theta^*| = \prod_{q \in \mathcal{Q}} B_q$, are given as $\Theta^* = \{\theta_{m_1,\dots,m_Q} | m_1 \in \mathcal{B}_1, \dots, m_Q \in \mathcal{B}_Q\}$. This results in an objective function given by

$$\min_{\Theta^*} \sum_{i \in \mathcal{I}} J(\hat{\mathbf{s}}_i), \tag{14}$$

where

$$J(\hat{\mathbf{s}}_i) = \begin{cases} (\theta_{m_1,\dots,m_Q} - \overline{y}_i)^2, & \text{if } (a_{m,q} \leq \hat{s}_{i,q} < a_{m,q}) \quad \forall q \in \mathcal{Q}, \forall m \in \mathcal{B}_q \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

which again converges to the fraction of positive samples within each bin. The term $J(\hat{\mathbf{s}}_i)$ simply denotes that the loss is only applied if a sample $\hat{\mathbf{s}}_i$ falls in a certain bin combination. A drawback of using this method is that the number of bins is given by $\prod_{q \in \mathcal{Q}} B_q$, which grows exponentially as the number of dimensions Q grows.

4.2 Scaling Methods

As opposed to binning methods like histogram binning, scaling methods perform a rescaling of the logits before sigmoid/softmax activation to obtain calibrated confidences. We can distinguish between logistic calibration (Platt scaling) [Pla99] and beta calibration [KSFF17].

Logistic calibration: According to the well-known logistic calibration (Platt scaling) [Pla99], the calibration parameters are commonly obtained using logistic regression. For a binary logistic regression model, we assume normally distributed scores for the positive and negative class, so that $\mathrm{p}(p|+) \sim \mathcal{N}(p; \mu_+, \sigma^2)$ and $\mathrm{p}(p|-) \sim \mathcal{N}(p; \mu_-, \sigma^2)$. The mean values for the two classes are given by μ_+, μ_- and the variance σ^2 is equal for all classes. First, we follow the derivation of logistic calibration introduced by [Pla99, KSFF17] using the likelihood ratio

$$LR(p) = \frac{\mathrm{p}(p|+)}{\mathrm{p}(p|-)} = \exp\left(\frac{1}{2\sigma^2}[-(p-\mu_+)^2 + (p-\mu_-)^2]\right) \tag{16}$$

$$= \exp\left(\frac{1}{2\sigma^2}[2p(\mu_+ - \mu_-) - (\mu_+^2 - \mu_-^2)]\right) \tag{17}$$

$$= \exp\left(\frac{\mu_+ - \mu_-}{2\sigma^2}[p - (\mu_+ + \mu_-)]\right) \tag{18}$$

$$= \exp(\gamma(p - \eta)), \tag{19}$$

where $\gamma = \frac{1}{2\sigma^2}(\mu_+ - \mu_-)$ and $\eta = \mu_+ + \mu_-$. Assuming a uniform prior over the positive and negative classes, the likelihood ratio equals the posterior odds. Hence, a calibrated probability is derived by

$$\mathrm{P}(+|\hat{p}) = \frac{1}{1 + LR(\hat{p})^{-1}} = \frac{1}{1 + \exp(-\gamma(\hat{p} - \eta))}, \tag{20}$$

which recovers the logistic function.

Recently, [KKSH20] used this formulation to derive a position-dependent confidence calibration by using multivariate Gaussians for the positive and negative classes. Introducing the concept of multivariate confidence calibration [KKSH20], we can derive a likelihood ratio for position-dependent logistic calibration by

$$LR_{LC}(\hat{\mathbf{s}}) = \exp\left(\frac{1}{2}\left[(\hat{\mathbf{s}}_-^\mathsf{T}\boldsymbol{\Sigma}_-^{-1}\hat{\mathbf{s}}_-) - (\hat{\mathbf{s}}_+^\mathsf{T}\boldsymbol{\Sigma}_+^{-1}\hat{\mathbf{s}}_+)\right] + c\right), \quad c = \log\frac{|\boldsymbol{\Sigma}_-|}{|\boldsymbol{\Sigma}_+|}, \tag{21}$$

where $\hat{\mathbf{s}}_+ = \hat{\mathbf{s}} - \boldsymbol{\mu}_+$ and $\hat{\mathbf{s}}_- = \hat{\mathbf{s}} - \boldsymbol{\mu}_-$ using $\boldsymbol{\mu}_+, \boldsymbol{\mu}_- \in \mathbb{R}^Q$ as the mean vectors and $\boldsymbol{\Sigma}_+, \boldsymbol{\Sigma}_- \in \mathbb{R}^{Q\times Q}$ as the covariance matrices for the positive and negative classes, respectively.

Beta calibration: Similarly, we can also extend the beta calibration method [KSFF17] to a multivariate calibration scheme. However, we need a special form of a multivariate beta distribution as the Dirichlet distribution is only defined over a Q-simplex and thus is not suitable for this kind of calibration. Therefore, we use a multivariate beta distribution proposed by [LN82] which is defined as

$$\mathrm{p}(\hat{\mathbf{s}}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{q \in Q} \left[\lambda_q^{\alpha_q} (\hat{s}_q^*)^{\alpha_q - 1} \left(\frac{\hat{s}_q^*}{\hat{s}_q} \right)^2 \right] \left[1 + \sum_{q \in Q} \lambda_q \hat{s}_q^* \right]^{-\sum_{q \in Q_0} \alpha_q}, \tag{22}$$

with $Q_0 = \{0, \ldots, Q\}$ and the shape parameters $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_q)^\mathsf{T}$, $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_q)^\mathsf{T}$ that are restricted to $\alpha_q, \beta_q > 0$. Furthermore, we denote $\lambda_q = \frac{\beta_q}{\beta_0}$ and $\hat{s}_q^* = \frac{\hat{s}_q}{1 - \hat{s}_q}$. In this context, $B(\boldsymbol{\alpha})$ denotes the multivariate beta function. However, this kind of beta distribution is only able to capture positive correlations [LN82]. Nevertheless, it is possible to derive a likelihood ratio given by

$$\begin{aligned} LR_{BC}(\hat{\mathbf{s}}) = \exp\Bigg(& \sum_{q \in Q} \left[\alpha_q^+ \log(\lambda_q^+) - \alpha_q^- \log(\lambda_q^-) + (\alpha_q^+ - \alpha_q^-) \log(\hat{s}_q^*) \right] + \\ & \sum_{q \in Q_0} \left[\alpha_q^- \log\left(\sum_{j \in Q} \lambda_j^- \hat{s}_j^* \right) - \alpha_q^+ \log\left(\sum_{j \in Q} \lambda_j^+ \hat{s}_j^* \right) \right] + c \Bigg), \end{aligned} \tag{23}$$

with $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-$ and $\boldsymbol{\lambda}^+, \boldsymbol{\lambda}^-$ as the shape parameters for the multivariate beta distribution in (22) and for the positive and negative class, respectively, so that $c = \log\left(\frac{B(\boldsymbol{\alpha}^-)}{B(\boldsymbol{\alpha}^+)} \right)$. We investigate the effect of position-dependent calibration in the next section using these methods.

5 Experimental Evaluation and Discussion

In this section, we evaluate our proposed calibration methods for the tasks of object detection, instance segmentation, and semantic segmentation using pretrained neural networks. For calibration evaluation, we use the MS COCO validation dataset [LMB+14] consisting of 5,000 images with 80 different object classes for object detection and instance segmentation. For semantic segmentation, we use the panoptic segmentation annotations consisting of 171 different object and stuff categories in total. Our investigations are limited to the validation dataset since the training set has already been used for network training and no ground-truth annotations are available for the respective test dataset. Thus, we use a 50%/50% random split and use the first set for calibration training, while the second set is used for the evaluation of the calibration methods. Furthermore, we also utilize the Cityscapes validation dataset

[COR+16] consisting of 500 images with 19 different classes that are used for model training. The Munster and Lindau images are used for calibration training, whereas the Frankfurt images are held back for calibration evaluation.

5.1 Object Detection

We evaluate our proposed calibration methods *histogram binning* (HB) (14), *logistic calibration* (LC) (21) and *beta calibration* (BC) (23) for the task of object detection using a pretrained `Faster R-CNN X101-FPN` [RHGS15, WKM+19] as well as a pretrained `RetinaNet` [LGG+17, WKM+19] on the MS COCO validation set. For Cityscapes, we use a pretrained `Mask R-CNN R50-FPN` [HGDG17, WKM+19], where only the predicted bounding box information is used for calibration. For calibration evaluation, we use the proposed D-ECE with the same subsets of data that have been used for calibration training. Thus, we use a binning scheme of $B = 20$ using the confidence $\hat{p}$ only. In contrast, we use $B_q = 5$ for $Q = 5$ when all auxiliary information is used. Each bin with less than 8 samples is neglected to increase D-ECE robustness. We further measure the *Brier score* (BS) and the *negative log likelihood* (NLL) of each model to evaluate its calibration properties. It is of special interest to assess if calibration has an influence to the precision/recall. Thus, we also denote the *area under precision/recall curve* (AUPRC) for each model. As opposed to previous examinations [KKSH20], we evaluate calibration using *all* available classes within each dataset. Each of these scores is measured for each class separately. In our experiments, we denote weighted average scores that are weighted by the amount of samples for each class. We perform calibration using IoU scores of 0.50 and 0.75, respectively. Furthermore, only bounding boxes with a score over 0.3 are used for calibration to reduce the amount of non-informative predictions. We give a qualitative example Fig. 2 that illustrates the effect of confidence calibration in the scope of object detection.

In our experiments, we first apply the standard calibration methods (Table 1) and compare the results with the baseline miscalibration of the detection model. We can already observe a default miscalibration of each examined network. This miscalibration is reduced by standard calibration where the scaling methods offer the best performance compared to the histogram binning. In a next step, we apply our box-sensitive calibration that includes the confidence $\hat{p}$, position c_x, c_y and shape w and h into a calibration mapping (Table 2). Similar to the confidence-only case in Table 1, the scaling methods consistently outperform baseline and histogram binning calibration.

In each calibration case, we observe a miscalibration of the base network that is alleviated using our proposed calibration methods. By examining the reliability diagram (Fig. 3), we observe that the networks are consistently miscalibrated for all confidence levels. This can be alleviated either by standard calibration or by position-dependent calibration. By examining the position-dependence of the

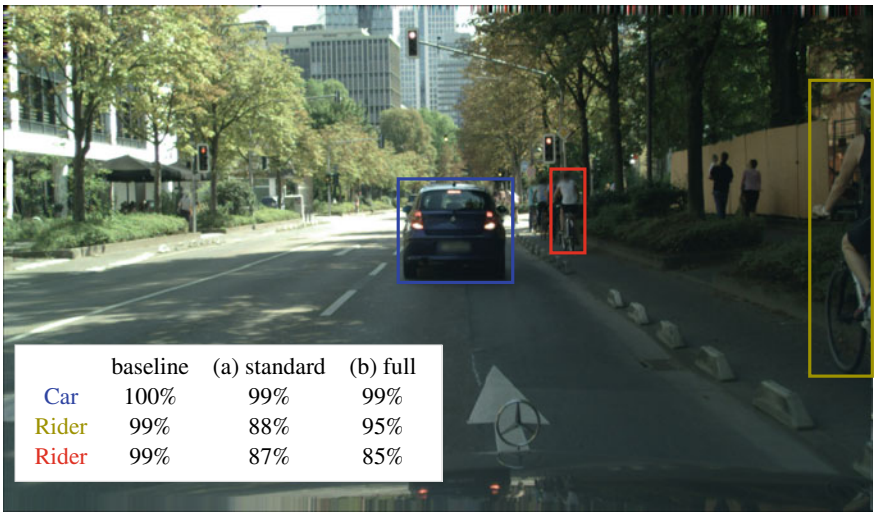


Fig. 2 Qualitative example of position-dependent confidence calibration on a Cityscapes image with detections obtained by a `Mask-RCNN`. First, standard calibration is performed using similar confidence estimates $\hat{p}$ without any position information. This results in rescaled but still similar calibrated confidence estimates (note that we have distinct calibration models for each class). Second, we can observe the effect of position-dependent calibration using the full position information (position and shape) as the two riders with similar confidence are rescaled differently. When using all available information, we can observe a significant difference in calibration since both, position and shape, have an effect to the calibration

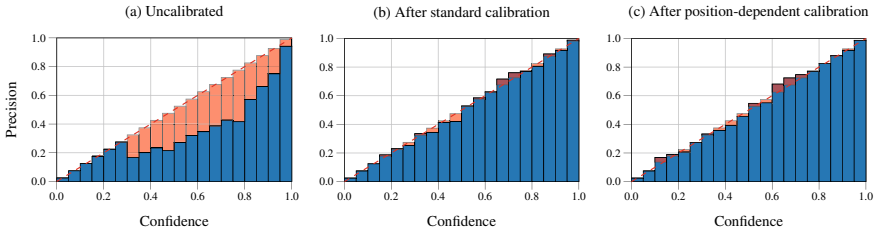


Fig. 3 Reliability diagram (object detection) for a `Mask-RCNN` class *pedestrian* inferred on Cityscapes Frankfurt images measured using the confidence only. The blue bar denotes the precision of samples that fall into this bin, with the gap to perfect calibration (diagonal) highlighted in red. **a** The `Mask-RCNN` model is consistently overconfident for each confidence level. **b** This can be alleviated by standard logistic calibration **c** as well as by position-dependent logistic calibration

Table 1 Calibration results for object detection using the confidence $\hat{p}$ only. The best scores are highlighted in bold. All calibration methods are able to improve miscalibration. The best performance is achieved using the scaling methods. Unlike histogram binning, the scaling methods are monotonically increasing and thus do not affect the AUPRC score

Network	Dataset	IoU	Cal. method	D-ECE (%)	Brier	NLL	AUPRC
`Faster R-CNN`	COCO	0.50	Baseline	17.839	0.206	0.622	0.833
			HB	5.622	0.166	0.598	0.781
			LC	4.449	**0.158**	**0.478**	0.833
			BC	**4.441**	**0.158**	0.482	0.833
		0.75	Baseline	29.721	0.272	0.861	0.767
			HB	5.173	0.158	0.594	0.676
			LC	**4.387**	**0.147**	**0.461**	0.766
			BC	4.540	**0.147**	0.463	0.766
`RetinaNet`	COCO	0.50	Baseline	10.061	0.171	0.514	0.831
			HB	5.332	0.166	0.579	0.795
			LC	4.464	**0.161**	**0.486**	0.831
			BC	**4.443**	**0.161**	0.487	0.831
		0.75	Baseline	17.543	0.181	0.544	0.765
			HB	5.773	0.147	0.551	0.721
			LC	4.395	**0.142**	**0.447**	0.765
			BC	**4.361**	**0.142**	**0.447**	0.765
`Mask R-CNN`	Cityscapes	0.50	Baseline	10.822	0.146	0.500	0.951
			HB	3.516	0.133	0.491	0.902
			LC	**3.305**	**0.125**	**0.380**	0.951
			BC	3.306	**0.125**	0.381	0.951
		0.75	Baseline	29.530	0.271	1.063	0.893
			HB	**3.718**	0.161	0.547	0.755
			LC	4.399	**0.136**	**0.425**	0.893
			BC	4.299	**0.136**	0.426	0.893

miscalibration (Figs. 4 and 5), a considerable increase of the calibration error toward the image boundaries can be observed. This calibration error is already well mitigated by standard calibration methods and can be further improved by position-dependent calibration. Also note that the AUPRC is not affected by standard scaling methods (logistic/beta calibration) as these methods perform a monotonically increasing mapping of the confidence estimates and thus do not affect the order of the samples. However, this is not the case with histogram binning which may lead to a significant drop of the AUPRC. Furthermore, even the position-dependent scaling methods cannot guarantee a monotonically increasing mapping. However, compared to the improvement of the calibration, the impact on the AUPRC is marginal. Therefore,

Table 2 Calibration results for object detection using all available information $\hat{p}$, c_x, c_y, w and h. Similar to the confidence-only case in Table 1, the scaling methods consistently outperform baseline and histogram binning calibration. However, in the position-dependent case, the calibration mapping is not monotonically increasing. This has an effect to the AUPRC scores, but is marginal compared to the improvement in calibration

Network	Dataset	IoU	Cal. method	D-ECE [%]	Brier	NLL	AUPRC
`Faster R-CNN`	COCO	0.50	Baseline	7.519	0.206	0.622	**0.833**
			HB	3.891	0.206	1.076	0.701
			LC	**3.100**	0.175	0.627	0.807
			BC	3.240	**0.168**	**0.508**	0.807
		0.75	Baseline	13.277	0.272	0.861	**0.767**
			HB	3.848	0.200	1.037	0.587
			LC	**2.965**	0.162	0.587	0.730
			BC	3.217	**0.159**	**0.492**	0.719
`RetinaNet`	COCO	0.50	Baseline	4.303	**0.171**	**0.514**	**0.831**
			HB	3.369	0.199	1.100	0.727
			LC	**3.124**	0.177	0.615	0.804
			BC	3.491	0.175	0.521	0.800
		0.75	Baseline	6.724	0.181	0.544	**0.765**
			HB	3.241	0.176	1.026	0.640
			LC	**2.872**	0.157	0.595	0.732
			BC	3.205	**0.153**	**0.479**	0.725
`Mask R-CNN`	Cityscapes	0.50	Baseline	10.168	0.146	0.500	**0.951**
			HB	5.241	0.151	0.535	0.854
			LC	**4.551**	**0.134**	0.440	0.946
			BC	6.117	**0.134**	**0.413**	0.925
		0.75	Baseline	27.066	0.271	1.063	0.893
			HB	8.062	0.195	0.607	0.681
			LC	**6.267**	**0.140**	**0.464**	**0.894**
			BC	9.427	0.160	0.491	0.833

we conclude that our calibration methods are a valuable contribution especially for safety-critical systems, since they lead to statistically better calibrated confidences.

5.2 Instance Segmentation

After object detection, we investigate the calibration properties of instance segmentation models. According to the definition of calibration for segmentation models in (9), we can use each pixel within a segmentation mask as a separate prediction.

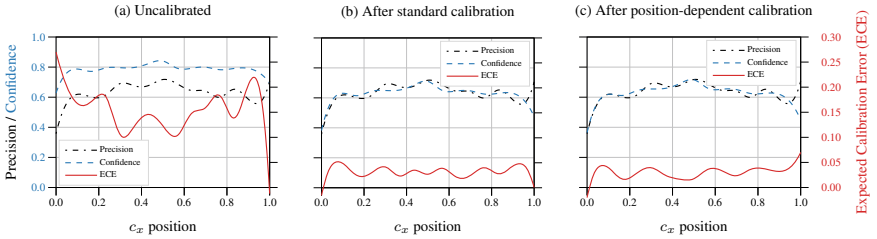


Fig. 4 Reliability diagram (object detection) for a `Mask-RCNN` class *pedestrian* inferred on Cityscapes Frankfurt images measured by the confidence and the relative c_x position of each bounding box. **a** We observe an increasing calibration error toward the boundaries of the images. **b** This can be alleviated by standard logistic calibration **c** as well as by position-dependent logistic calibration

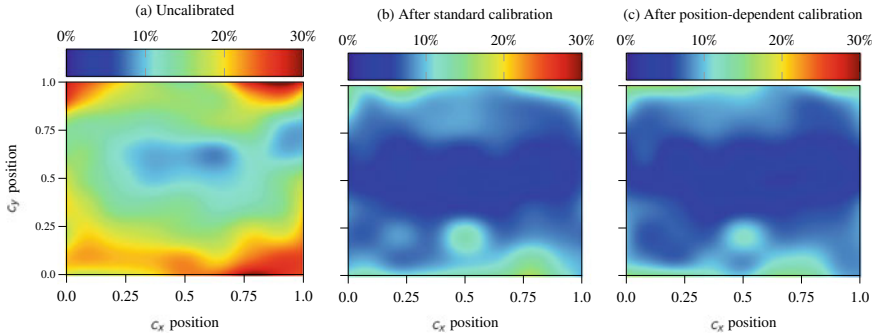


Fig. 5 Reliability diagram (object detection) for a `Mask-RCNN` class *pedestrian* measured using all available information. This diagram shows the D-ECE [%] for all image regions. **a** We can observe a default miscalibration that increases toward the image boundaries. **b** Standard logistic calibration already achieves good calibration results, **c** that can be further improved by position-dependent calibration

This alleviates the problem of limited data availability and allows for a more robust calibration evaluation. Recently, Kumar et al. [KLM19] started a discussion about the sample-efficiency of binning and scaling methods. The authors show that binning methods yield a more robust calibration mapping for large datasets but also tend to overfit for a small database. We can confirm this observation as histogram binning provides poor calibration performance in our examinations on object detection calibration, particularly for classes with fewer samples. In contrast, scaling methods are more sample-efficient but also more inaccurate [KLM19]. Furthermore, scaling methods are computationally more expensive as they require an iterative update of the calibration parameters over the complete dataset, whereas a binning of samples comes at low computational costs, especially for large datasets. Therefore, our examinations are focused on the calibration performance of a (multivariate) histogram binning using 15 bins for each dimension. For inference, we use pretrained `Mask R-CNN` [HGDG17, WKM+19] as well as pretrained `PointRend` [KWHG20] models to obtain predictions of instance segmentation masks for both datasets. As within

object detection evaluation, all objects with a bounding box score below 0.3 are neglected. We perform standard calibration using the confidence only, as well as a position-dependent calibration including the x and y position of each pixel. The pixel position is scaled by the mask's bounding box size to get position information in the [0, 1] interval. Furthermore, we also include the distance of each pixel to the nearest mask segment boundary as a feature in a calibration mapping since we expect a higher uncertainty especially at the segment boundaries. The distance is normalized by the bounding box's diagonal to obtain distance values in [0, 1].

Since many data samples are available, we measure calibration using a D-ECE with 15 bins neglecting each bin with less than 8 samples. We further assess the Brier score as well as the NLL loss as complementary metrics. The task of instance segmentation is related to object detection. In a first step, the network also needs to infer a bounding box for each detected object. Similar to the calibration evaluation for object detection, we further use IoU scores of 0.50 and 0.75 to specify whether a prediction has matched a ground-truth object or not. In contrast to object detection, the IoU score within instance segmentation is computed by the overlap of the inferred segmentation mask and the according ground-truth object. Using this definition, we can compute the AUPRC to evaluate the average quality of the object detection branch as well as the according segmentation masks. All results for instance segmentation calibration for IoU of 0.50 and 0.75 are shown in Tables 3 and 4, respectively. These tables compare standard calibration (subset: confidence only) with position-dependent calibration (subset: full). We observe a significant miscalibration of the networks by default. Using standard calibration or our calibration methods, it is possible to improve the calibration score D-ECE, Brier, and NLL for each case. The miscalibration is successfully corrected by histogram binning using either the standard binning or the position-dependent binning. This is also underlined by inspecting the reliability diagrams for the confidence-only case (Fig. 6). Furthermore, we can also observe miscalibration that is dependent on the relative x and y pixel position within a mask (Figs. 7 and 8). We can observe that standard histogram binning calibration already achieves good calibration results but also offers a weak dependency on the pixel position as well. Although the position-dependent calibration only achieves a minor improvement in calibration compared to the standard calibration case, it shows an equal calibration performance across the whole image

In contrast to object detection, we observe a slightly increased miscalibration toward the center of a mask. As it can be seen in Fig. 9, most pixels belonging to an object mask are also located in the center. This underlines the need for a position-dependent calibration. Interestingly, although position-dependent calibration does not seem to offer better Brier or NLL scores compared to the confidence-only case, it significantly improves the mask quality as we can observe higher AUPRC scores for position-dependent calibration. We further provide a qualitative example that illustrates the effect of confidence calibration in Fig. 9. In this example, we can see the difference between standard calibration and position-dependent calibration. In the former case, the mask scores are only rescaled by their confidence which might lead to a better calibration but sometimes also to unwanted losses of mask segments

Table 3 Calibration results for instance segmentation @ IoU=0.50. The best D-ECE scores are underlined *for each subset separately*, since it is not convenient to compare D-ECE scores with different subsets to each other [KKSH20]. Furthermore, the best Brier, NLL, and AUPRC scores are highlighted in bold. These scores are only calculated using the confidence information and thus can be compared to each other. The histogram binning calibration consistently improves miscalibration. Furthermore, we find that although position dependence does not improve the Brier and NLL scores as in the standard case, it leads to a significant improvement in the mIoU score

Network	Dataset	Cal. method	Subset: confidence only				Subset: full			
			D-ECE [%]	Brier	NLL	AUPRC	D-ECE [%]	Brier	NLL	AUPRC
Mask R-CNN	CS	Baseline	7.088	0.110	0.432	0.724	11.205	0.110	0.432	0.724
		HB	5.661	**0.099**	**0.320**	0.723	10.119	0.117	0.530	**0.787**
	COCO	Baseline	21.983	0.222	0.940	0.663	23.409	0.222	0.940	0.663
		HB	6.420	**0.150**	**0.442**	0.662	13.640	0.171	0.776	**0.760**
PointRend	CS	Baseline	12.893	0.160	0.785	0.709	20.858	0.160	0.785	0.709
		HB	2.706	**0.105**	**0.326**	0.698	19.021	0.190	1.299	**0.758**
	COCO	Baseline	22.284	0.222	0.946	0.672	23.973	0.222	0.946	0.672
		HB	6.348	**0.144**	**0.428**	0.664	16.060	0.180	1.005	**0.751**

Table 4 Calibration results for instance segmentation @ IoU=0.75. The best scores are highlighted. We observe the same behavior in calibration for all models as for the IoU=0.50 case shown in Table 3

Network	Dataset	Cal. method	Subset: confidence only				Subset: full			
			D-ECE [%]	Brier	NLL	AUPRC	D-ECE [%]	Brier	NLL	AUPRC
Mask R-CNN	CS	Baseline	12.862	0.147	0.622	0.375	14.470	0.147	0.622	0.375
		HB	5.895	**0.108**	**0.340**	0.375	10.273	0.120	0.523	**0.479**
	COCO	Baseline	26.559	0.250	1.070	0.237	27.219	0.250	1.070	0.237
		HB	6.006	**0.144**	**0.423**	0.235	12.888	0.165	0.720	**0.425**
PointRend	CS	Baseline	18.668	0.192	0.929	0.347	25.386	0.192	0.929	0.347
		HB	3.899	**0.115**	**0.349**	0.344	18.415	0.191	1.247	**0.486**
	COCO	Baseline	26.643	0.248	1.060	0.258	27.377	0.248	1.060	0.258
		HB	6.708	**0.138**	**0.411**	0.238	15.344	0.173	0.936	**0.388**

(especially small objects in the background). In contrast, position-dependent calibration is capable of possible correlations between confidence and pixel position or the size of an object. This leads to improved estimates of the mask confidences even for smaller objects. Therefore, we conclude that multidimensional confidence calibration has a positive influence on the calibration properties as well as on the quality of the object masks.

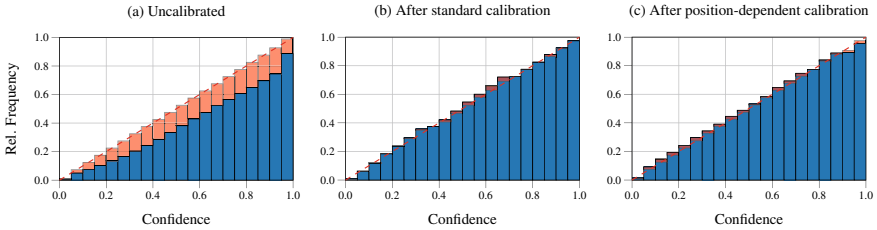


Fig. 6 Reliability diagram (instance segmentation) for a `Mask R-CNN` class *pedestrian* inferred on MS COCO validation images measured using the confidence only. The blue bar denotes the frequency of samples that fall into this bin, with the gap to perfect calibration (diagonal) highlighted in red. **a** The `Mask R-CNN` model is consistently overconfident for each confidence level. **b** This can be alleviated by standard histogram binning calibration **c** as well as by position-dependent logistic calibration

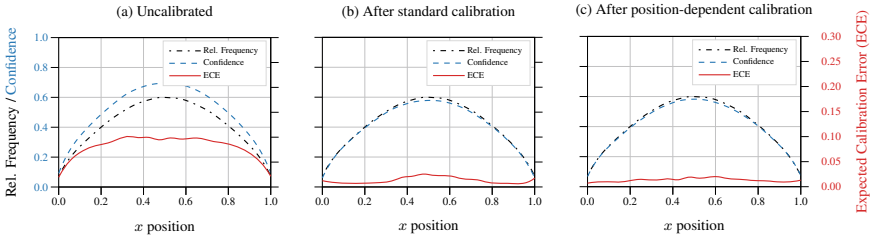


Fig. 7 Reliability diagram (instance segmentation) for a `Mask R-CNN` class *pedestrian* inferred on MS COCO validation images measured by the relative x position of each pixel within its bounding box. **a** Similar to the examinations for object detection, we observe an increasing calibration error toward the boundaries of the images. **b** Standard histogram binning significantly reduces the calibration error, **c** whereas position-dependent histogram binning is capable of a slightly further refinement of the position-dependent D-ECE

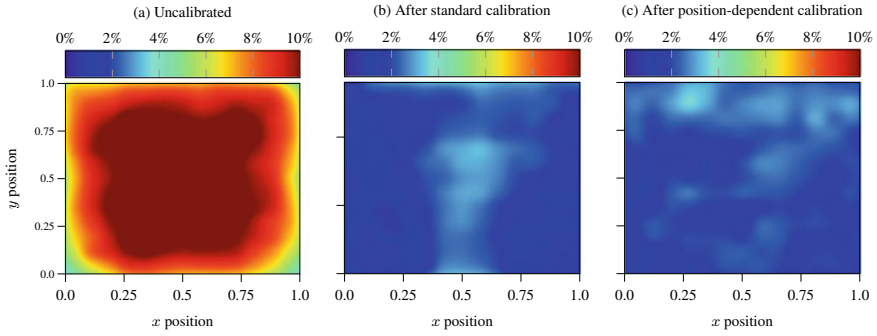


Fig. 8 Reliability diagram (instance segmentation) for a `Mask R-CNN` class *pedestrian* inferred on MS COCO validation images. The diagram shows the D-ECE [%] that is measured using the relative x and y position of each pixel within its bounding box. Similar to the observations in Fig. 7, standard calibration already reduces the calibration error. However, we can observe a slightly increased calibration error toward the image's center. In contrast, position-dependent calibration error shows equal calibration performance across the whole image

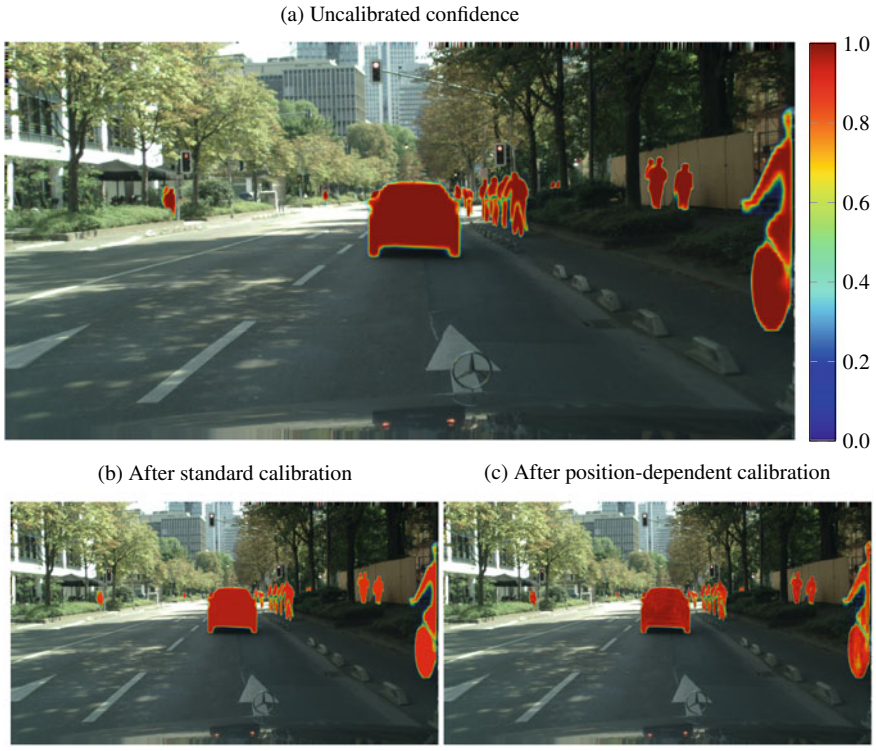


Fig. 9 Qualitative example of a Cityscapes image with instance segmentation masks obtained by a `Mask R-CNN`. **a** Uncalibrated confidence of the predicted masks (mIoU = 0.289). **b** Instance masks are calibrated using standard histogram binning (mIoU = 0.343). **c** Mask confidences using a position-dependent confidence calibration (mIoU = 0.370). Unlike a simple scaling of confidences in **b**, the calibration in **c** offers a more fine-grained mapping that achieves a better fit to the true instance masks

5.3 Semantic Segmentation

For the evaluation of semantic segmentation, we use the same datasets, the same binning scheme, and the same features that have already been used for the instance segmentation in Sect. 5.2. For COCO mask inference, we use a pretrained `Deeplabv2` [CPK+18] as well as a pretrained `HRNet` model [SXLW19, WSC+20, YCW20]. For Cityscapes, we also use a `HRNet` as well as a pretrained `Deeplabv3+` model [CZP+18]. Similar to the instance segmentation, we also use the D-ECE, Brier score, NLL loss, and mIoU to compare the baseline calibration with a histogram binning model. As opposed to our previous experiments, we only use 15% of the provided samples to reduce computational complexity. The results for default miscalibration, standard calibration (subset: confidence only) and position-dependent calibration (subset: full) are shown in Table 5. Unlike instance segmentation calibration, the

Table 5 Calibration results for semantic segmentation. The best D-ECE scores are underlined *for each subset separately*, since it is not convenient to compare D-ECE scores with different subsets to each other [KKSH20]. Furthermore, the best Brier, NLL, and mIoU scores are highlighted in bold. These scores are only calculated using the confidence information and thus can be compared to each other. In contrast to object detection and instance segmentation evaluation, we observe a low baseline calibration error that is only slightly affected by confidence calibration

Network	Dataset	Cal. method	Subset: confidence only				Subset: full			
			D-ECE [%]	Brier	NLL	mIoU	D-ECE [%]	Brier	NLL	mIoU
`Deeplabv3+`	CS	Baseline	0.162	**0.060**	**0.139**	**0.623**	0.186	**0.060**	**0.139**	**0.623**
		HB	0.081	**0.060**	0.170	0.619	0.199	0.062	0.189	0.589
`Deeplabv2`	COCO	Baseline	0.094	0.458	**1.173**	**0.933**	0.149	0.458	**1.173**	**0.933**
		HB	0.060	**0.456**	1.515	**0.933**	0.150	0.485	1.790	0.913
`HRNet`	CS	Baseline	0.067	**0.057**	**0.115**	**0.629**	0.149	**0.057**	**0.115**	**0.629**
		HB	0.083	**0.057**	0.148	0.628	0.191	0.060	0.171	0.582
	COCO	Baseline	0.455	0.779	5.812	**0.939**	0.455	0.779	5.812	**0.939**
		HB	0.062	**0.563**	**2.261**	**0.939**	0.138	0.571	2.372	0.931

baseline model is already intrinsically well-calibrated, offering a very low calibration error. A qualitative example is shown in Fig. 10 to illustrate the effect of calibration for semantic segmentation. In this example, we can observe only minor differences between the uncalibrated mask and the masks either after standard calibration or after position-dependent calibration. This aspect is supported by the confidence reliability diagram shown in Fig. 11. Although we can observe an overconfidence, most samples are located in the low confident space with a low calibration error that also results in an overall low miscalibration score. Furthermore, our calibration methods are able to achieve even better calibration results, but mostly in the confidence-only calibration case. In addition, we also observe only a low correlation between position and calibration error (Figs. 12 and 13). One reason for the major difference between instance and semantic segmentation calibration may be the difference in model training. An instance segmentation model needs to infer an appropriate bounding box first to achieve qualitatively good results in mask inference. In contrast, a semantic segmentation model does not need to infer the position of objects within an image but is able to directly learn and improve mask quality. We also suspect an influence of the amount of available data points, since a semantic segmentation model is able to use each pixel within an image as a separate sample, whereas an instance segmentation model is only restricted to the pixels that are available within an estimated bounding box. Therefore, we conclude that semantic segmentation models do not require a post-hoc confidence calibration as they already offer a good calibration performance.

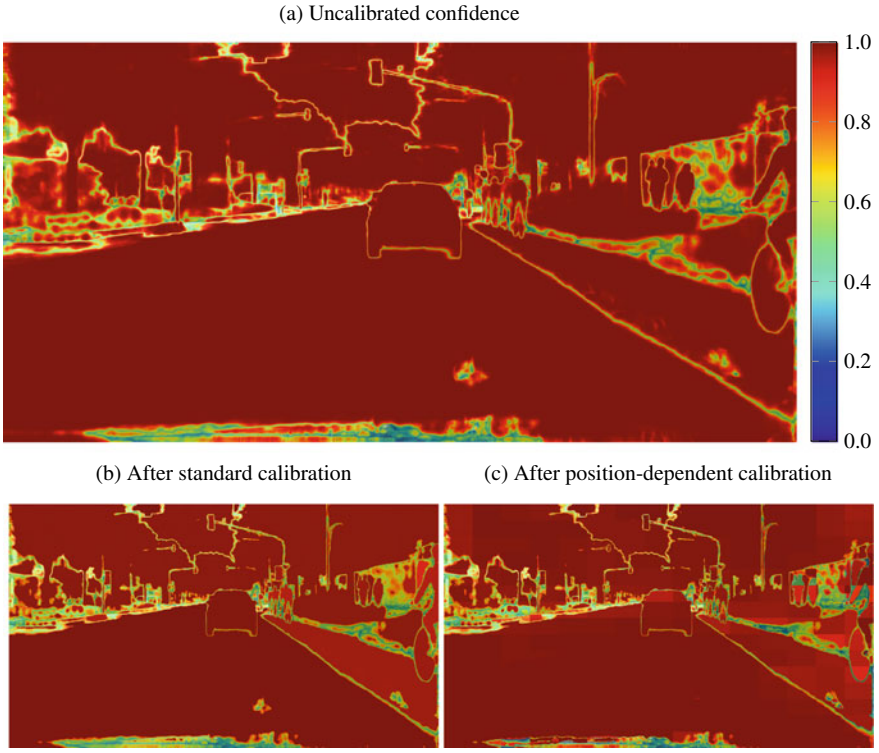


Fig. 10 Qualitative example of a Cityscapes image with semantic segmentation masks obtained by a `Deeplabv3+` model. **a** Uncalibrated confidence of the predicted masks (mIoU = 0.576), **b** here, the pixel masks are calibrated using standard histogram binning (mIoU = 0.522). Furthermore, **c** mask confidences using a position-dependent confidence calibration (mIoU = 0.570). Although we observe some minor differences, we conclude that confidence calibration does not have a major influence to the calibration properties for semantic segmentation models

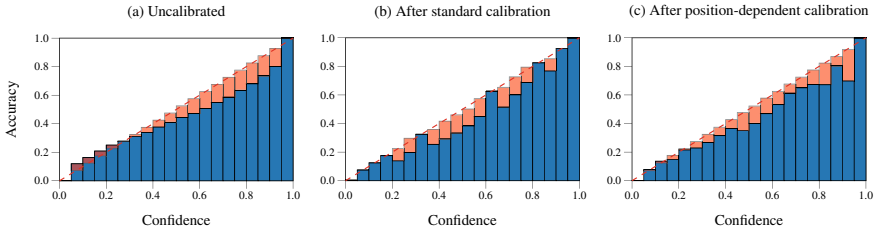


Fig. 11 Reliability diagram (semantic segmentation) for a `Deeplabv3+` class *pedestrian* inferred on Cityscapes Frankfurt images measured using the confidence only. The blue bar denotes the accuracy of samples that fall into this bin, with the gap to perfect calibration (diagonal) highlighted in red. By default (**a**), the `Deeplabv3+` model already offers a good calibration performance that is only slightly affected either by standard calibration (**b**) or by position-dependent calibration (**c**)

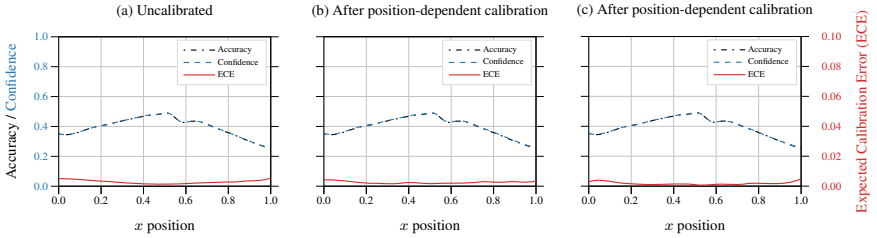


Fig. 12 Reliability diagram (semantic segmentation) for a `Deeplabv3+` class *pedestrian* inferred on Cityscapes Frankfurt images measured by the relative x position of each pixel. Since the model has a low miscalibration by default (**a**), the effect of calibration in **b** or **c** is marginal

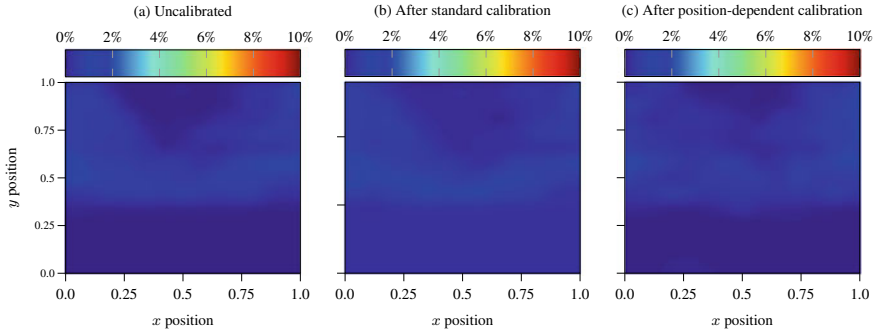


Fig. 13 Reliability diagram (semantic segmentation) for a `Deeplabv3+` class *pedestrian* inferred on Cityscapes Frankfurt images. This diagram shows the D-ECE [%] that is measured using the relative x and y position of each pixel. Since the model has a low miscalibration by default (**a**), the effect of calibration in **b** or **c** is marginal

Conclusions

Within the scope of confidence calibration, recent work has mainly focused on the task of classification calibration. In this chapter, we presented an analysis of confidence calibration for object detection calibration as well as, for instance, and semantic segmentation calibration. Firstly, we introduced definitions for confidence calibration within the scope of object detection, instance segmentation, and semantic segmentation. Secondly, we presented methods to measure and alleviate miscalibration of detection and segmentation networks. These methods are extensions of well-known calibration methods such as histogram binning [ZE01], Platt scaling [Pla99], and beta calibration [KSFF17]. We extend these methods so that they encompass additional calibration information such as position and shape. Finally, the experiments revealed that common object detection models as well as instance segmentation networks tend to miscalibration. In addition, we showed that auxiliary information such as estimated position or shape of a predicted object also have an influence to confidence calibration. However, we also found that semantic segmentation models are

already intrinsically calibrated. Thus, the examined models do not require additional post-hoc calibration and already offer well-calibrated mask confidence scores. We argue that this difference between instance and semantic segmentation is a result of data quality and availability during training. This leads to the assumption that limited data availability is a direct cause for miscalibration during training and thus an effect of overfitting. Nevertheless, our proposed calibration framework is capable to calibrate object detection and instance segmentation models. In safety-critical applications, the confidence in the applied algorithms is paramount. The proposed calibration algorithms allow to detect situations of low confidence and thus perform the appropriate system reaction. Therefore, calibrated confidence values can be used as additional information especially in safety-critical applications.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)". The authors would like to thank the consortium for the successful cooperation.

References

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, June 2016), pp. 3213–3223

[CPK+18] Liang-Chieh. Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille, DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **40**(4), 834–848 (2018)

[CZP+18] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Munich, Germany, September 2018), pp. 833–851

[DLXS20] Y. Ding, J. Liu, J. Xiong, Y. Shi, Revisiting the evaluation of uncertainty estimation and its application to explore model complexity-uncertainty trade-off, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* virtual conference (June 2020), pp. 22–31

[GPSW17] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, in *Proceedings of the International Conference on Machine Learning (ICML)* (Sydney, NSW, Australia, August 2017), pp. 1321–1330

[HGDG17] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Venice, Italy, October 2017), pp. 2980–2988

[JJY+19] B. Ji, H. Jung, J. Yoon, K. Kim, Y. Shin, Bin-wise temperature scaling (BTS): improvement in confidence calibration performance through simple scaling techniques, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops* (Seoul, Korea, October 2019), pp. 4190–4196

[KG20] D. Karimi, A. Gholipour, Improving Calibration and Out-of-Distribution Detection in Medical Image Segmentation With Convolutional Neural Networks (May 2020), pp. 1–12. arXiv:2004.06569

[KKSH20] F. Küppers, J. Kronenberger, A. Shantia, A. Haselhoff, Multivariate confidence calibration for object detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, virtual conference (June 2020), pp. 1322–1330

[KLM19] A. Kumar, P.S. Liang, T. Ma, Verified uncertainty calibration, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Vancouver, BC, Canada, December 2019), pp. 3787–3798

[KPNK+19] M. Kull, M. Perello Nieto, M. Kängsepp, T. Silva Filho, H. Song, P. Flach, Beyond temperature scaling: obtaining well-calibrated multi-class probabilities with Dirichlet calibration, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Vancouver, BC, Canada, December 2019), pp. 12316–12326

[KSFF17] M. Kull, T. Silva Filho, P. Flach, Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers, in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)* (Fort Lauderdale, FL, USA, May 2017), pp. 623–631

[KWHG20] A. Kirillov, Y. Wu, K. He, R. Girshick, PointRend: image segmentation as rendering, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (June 2020), pp. 9799–9808

[LGG+17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Venice, Italy, October 2017), pp. 2980–2988

[LMB+14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. Lawrence Zitnick, Microsoft COCO: common objects in context, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Zurich, Switzerland, September 2014), pp. 740–755

[LN82] David L. Libby, Melvin R. Novick, Multivariate generalized beta distributions with applications to utility assessment. J. Educ. Stat. **7**(4), 271–294 (1982)

[MWT+20] Alireza Mehrtash, William M. Wells, Clare M. Tempany, Purang Abolmaesumi, Tina Kapur, Confidence calibration and predictive uncertainty estimation for deep medical image segmentation. IEEE Trans. Med. Imaging **39**(12), 3868–3878 (2020)

[NC16] M. Naeini, G. Cooper, Binary classifier calibration using an ensemble of near isotonic regression models, in *Proceedings of the IEEE International Conference on Data Mining (ICDM)* (Barcelona, Spain, December 2016), pp. 360–369

[NCH15] M. Pakdaman Naeini, G. Cooper, M. Hauskrecht, Obtaining well calibrated probabilities using Bayesian binning, in *Proceedings of the AAAI Conference on Artificial Intelligence* (Austin, TX, USA, January 2015), pp. 2901–2907

[NDZ+19] J. Nixon, M.W. Dusenberry, L. Zhang, G. Jerfel, D. Tran, Measuring calibration in deep learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (Long Beach, CA, USA, June 2019), pp. 38–41

[Pla99] J. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in *Advances in Large Margin Classifiers*. ed. by A.J. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (MIT Press, 1999), pp. 61–74

[RHGS15] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, December 2015), pp. 91–99

[SKR+21] F. Schwaiger, M. Henne Fabian Küppers, F. Schmoeller Roza, K. Roscher, A. Haselhoff, From black-box to white-box: examining confidence calibration under different conditions, in *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI)*, virtual conference (February 2021), pp. 1–8

[SXLW19] K. Sun, B. Xiao, D. Liu, J. Wang, Deep high-resolution representation learning for human pose estimation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, June 2019), pp. 5693–5703

[WKM+19] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, R. Girshick, Detectron2 (2019). [Online; assessed 2021-11-18]

[WLK+20] T. Wang, Y. Li, B. Kang, J. Li, J. Liew, S. Tang, S. Hoi, J. Feng, The devil is in classification: a simple framework for long-tail instance segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)*, virtual conference (August 2020), pp. 728–744

[WSC+20] Jingdong Wang, Ke. Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Mu. Yadong, Mingkui Tan, Xinggang Wang et al., Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **43**(10), 3349–3364 (2020)

[YCW20] Y. Yuan, X. Chen, J. Wang, Object-contextual representations for semantic segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)*, virtual conference (August 2020), pp. 173–190

[ZE01] B. Zadrozny, C. Elkan, Obtaining calibrated probability estimates from decision trees and Naive Bayesian classifiers, in *Proceedings of the International Conference on Machine Learning (ICML)* (Williamstown, MA, USA, June 2001), pp. 609–616

[ZE02] B. Zadrozny, C. Elkan, Transforming classifier scores into accurate multiclass probability estimates, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (Edmonton, AB, Canada, July 2002), pp. 694–699

Uncertainty Quantification for Object Detection: Output- and Gradient-Based Approaches

Tobias Riedlinger, Marius Schubert, Karsten Kahl, and Matthias Rottmann

Abstract Safety-critical applications of deep neural networks require reliable confidence estimation methods with high predictive power. However, evaluating and comparing different methods for uncertainty quantification is oftentimes highly context-dependent. In this chapter, we introduce flexible evaluation protocols which are applicable to a wide range of tasks with an emphasis on object detection. In this light, we investigate uncertainty metrics based on the network output, as well as metrics based on a learning gradient, both of which significantly outperform the confidence score of the network. While output-based uncertainty is produced by post-processing steps and is computationally efficient, gradient-based uncertainty, in principle, allows for localization of uncertainty within the network architecture. We show that both sources of uncertainty are mutually non-redundant and can be combined beneficially. Furthermore, we show direct applications of uncertainty quantification by improving detection accuracy.

1 Introduction

Deep artificial neural networks (DNNs) employed for tasks such as object detection or semantic segmentation yield by construction probabilistic predictions on feature data, such as camera images. Modern deep object detectors [RHGS15, LAE+16, LGG+17, RF18] predict bounding boxes for depicted objects of a given set of learned classes. The so-called "score" (sometimes also called objectness or confidence score)

T. Riedlinger (✉) · M. Schubert · K. Kahl · M. Rottmann
University of Wuppertal, Gaußstr. 20, 42119 Wuppertal, Germany
e-mail: riedlinger@uni-wuppertal.de

M. Schubert
e-mail: schubert@uni-wuppertal.de

K. Kahl
e-mail: kkahl@uni-wuppertal.de

M. Rottmann
e-mail: rottmann@uni-wuppertal.de

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_9

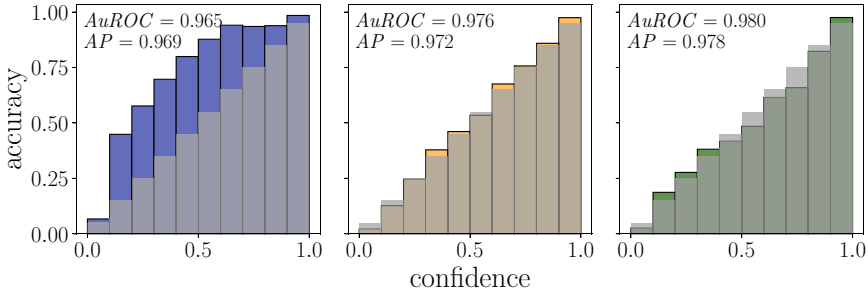


Fig. 1 Confidence calibration plots for score (left), Monte-Carlo (MC) dropout uncertainty (center), and our proposed gradient uncertainty metrics G (right) on $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$. For the sake of reference, we provide the optimally calibrated diagonal in gray. See Sect. 4 for more details

indicates the probability that a bounding box provided by the DNN contains an object. Throughout this chapter, we sometimes use the term "confidence" instead of "score" to refer to quantities that represent the probability of a detection being correct. The term confidence in a broader sense is meant to reflect an estimated degree of competency of the model when evaluating an input.

Reliable and accurate probabilistic predictions are desirable for many applications, such as medical diagnosis or automated driving. It is well-known that predictions of DNNs often tend to be statistically *miscalibrated* [SZS+14, GSS15, GPSW17], i.e., the score computed by the DNN is not representative for the relative frequency of correct predictions. For an illustration of this, see Fig. 1, where we compare different sources of uncertainty in terms of their accuracy conditioned to confidence bins for a `YOLOv3` model.

For individual DNN predictions, confidences that are not well-calibrated are misleading and constitute a reliability issue. Over-confident predictions might cause inoperability of a perception system due to producing non-existent instances (false positives / FP). Perhaps even more adverse, under-confidence might lead to false negative (FN) predictions, for instance overlooking pedestrians or other road users which could lead to dangerous situations or even fatalities. Given that the set of available data features is fixed, the sources of uncertainty in machine learning (and deep learning) can be divided into two types [HW21], referring to their primary source [Gal17, KG17]. Whereas *aleatoric* uncertainty refers to the inherent and irreducible stochasticity of the distribution the data stems from, *epistemic* uncertainty refers to the reducible part of the uncertainty. The latter originates from the finite size of a random data sample used for training, as well as from the choice of model and learning algorithm.

Up to now, there exist a moderate number of established methods in the field of deep learning that have demonstrated to properly estimate the uncertainty of DNNs. In [KG17], DNNs for semantic segmentation are equipped with additional regression output that model heteroscedastic (in machine learning also called predictive) uncertainty, with the aim to capture aleatoric uncertainty. For training, the uncer-

tainty output is integrated into the usual empirical cross-entropy loss. A number of adaptations for different modern DNN architectures originate from that work. From a theoretical point of view, Bayesian DNNs [DL90, Mac92], in which the model weights are treated as random variables, yield an attractive framework for capturing the epistemic uncertainty. In practice, the probability density functions corresponding to the model weights are estimated via Markov chain Monte-Carlo, which currently is infeasible already for a moderate number of model weights, hence being inapplicable in the setting of object detection. As a remedy, variational inference approaches have been considered, where the model weights are sampled from predefined distributions, which are often modeled in simplified ways, e.g., assuming mutual independence. A standard method for the estimation of epistemic uncertainty based on variational inference is Monte-Carlo (MC) dropout [SHK+14, GG16]. Algorithmically, this method performs several forward passes under dropout at inference time. Deep ensemble sampling [LPB17] follows a similar idea. Here, separately trained models with the same architecture are deployed to produce a probabilistic inference.

In semantic segmentation, Rottmann et al. [RCH+20] introduced the tasks of meta classification and regression that are both applied as post-processing to the output of DNNs. Meta classification refers to the task of classifying a prediction as true positive (TP) or FP. Meta regression is a more object detection-specific task with the aim to predict the intersection-over-union (IoU) of a prediction with its ground truth directly. These tasks are typically learned with the help of ground truth, but afterwards performed in the absence of ground truth, only on the basis of uncertainty measures stemming from the output of the DNN. The framework developed in [RCH+20], called MetaSeg, was extended into multiple directions: Time-dynamic uncertainty quantification for semantic segmentation [MRG20] and instance segmentation [MRV+21]; a study on the influence of image resolution [RS19]; controlled reduction of FN and meta fusion, the latter utilizes the uncertainty metrics to increase the performance of the DNN [CRH+19]; out-of-distribution detection [BCR+20, ORF20, CRG21]; and active learning [CRGR21]. Last but not least, a similar approach solely based on the output of the DNN has been developed for object detection in [SKR20] (called MetaDetect), however, equipped with different uncertainty metrics specifically designed for the task of object detection. This approach was also used in an uncertainty-aware sensor fusion approach for object detection based on camera images and RaDAR point clouds; see [KRBG21].

In [ORG18], it was proposed to utilize a learning gradient, replacing the usually needed ground truth labels by the prediction of the network itself, which is supposed to contain epistemic uncertainty information. Since the investigation of gradient metrics for image classification tasks, that method was considered in different applications as well. Gradient metrics were compared to other uncertainty quantification methods in [SFKM20]. It was found that gradient uncertainty is strongly correlated with the softmax entropy of classification networks. In natural language understanding, gradient metrics turned out to be beneficial; see [VSG19]. Therein, gradient metrics and deep ensemble uncertainty were aggregated and yield well-calibrated confidences on out-of-distribution data. Recently, gradient metrics have

been developed for object detection in [RRSG21]. In order to demonstrate the predictive power of uncertainty metrics, the framework of meta classification and regression provided by MetaDetect was used.

In this chapter, we review the works published in [SKR20, RRSG21] and put them into a common context. More precisely, our contributions are as follows:

- We review the concepts of meta classification and regression, meta fusion, and confidence calibration. We explain how they serve as a general benchmark for evaluating the predictive power of any uncertainty metric developed for object detection.
- We review output-based uncertainty metrics developed in [SKR20], as well as gradient-based ones from inside the network [RRSG21].
- We compare baseline uncertainty measures such as the DNN's score, well-established ones like Monte-Carlo dropout [Gal17], the output-based uncertainty metrics [SKR20], and finally the gradient-based uncertainty metrics [RRSG21] from inside the DNN. We do so in terms of comparing their standalone performances but also in terms of analyzing their mutual information and how much performance they add to the prediction of the network itself.

2 Related Work

In this section, we gather and discuss previous research related to the present work.

Epistemic and aleatoric uncertainty in object detection: In recent years, there has been increasing interest in methods to properly estimate or quantify uncertainty in DNNs. *Aleatoric uncertainty*, the type of uncertainty resulting from the data generating process, has been proposed to be captured by providing additional regression output to the network architecture for learning uncertainty directly from the training dataset [KG17]. Since the initial proposition, this approach has been adapted to, among others, the object detection setting [LDBK18, CCKL19, HZW+19, LHK+21, HSW20, CCLK20]. The latter approaches add further regression variables for aleatoric uncertainty which are assigned to their bounding box localization variables, thereby learning localization as Gaussian distributions. An alternative approach to quantifying localization uncertainty is to learn for each predicted box the corresponding *IoU* with additional regression variables [JLM+18].

Several methods have been proposed that aim to capture *epistemic uncertainty*, i.e., "reducible" kinds of uncertainty. Monte-Carlo (MC) dropout [SHK+14, GG16] employs forward passes under dropout and has become one of the standard tools for estimating epistemic uncertainty. As such, MC dropout has been investigated also for deep object detection [MNDS18, KD19, HSW20]. Similarly to MC dropout, deep ensemble methods [LPB17] employ separately trained networks ("experts") to obtain variational inference, an approach which also has found adaptations in deep object detection [LGRB20].

Uncertainty calibration: A large variety of methods have been proposed to rectify the intrinsic confidence miscalibration of modern deep neural networks by introducing post-processing methods (see, e.g., [GPSW17] for a comparison). Prominent examples of such methods include histogram binning [ZE02], isotonic regression [ZE01], Bayesian binning [NCH15], Platt scaling [Pla99, NMC05], or temperature scaling [HVD14]. Alternatively, calibrated confidences have been obtained by implementing calibration as an optimization objective [KP20] via a suitable loss term. In the realm of object detection, the authors of [NZV18] found that temperature scaling improves calibration. Also, natural extensions to make calibration localization-dependent have been proposed [KKSH20].

Meta classification and meta regression: Meta classification denotes the task of discriminating FP and FN predictions based on uncertainty or confidence information, an idea that has been explored initially in [HG17]. In case a real-valued metric (e.g., different versions of the IoU which is in [0, 1]) can be assigned to the quality of a prediction, meta regression denotes the task of predicting this quantity in the same spirit as meta classification. In contrast to the IoUNet framework introduced in [JLM+18], meta classification and meta regression do not require architectural changes or dedicated training of the respective DNN. In [RRSG21], it has been found that meta classifiers tend to yield well-calibrated confidences by default as opposed to the confidence score of DNNs. Applications of meta classification and meta regression have since been introduced for several different disciplines [CRH+19, RS19, MRG20, RMC+20, RCH+20, MRV+21, KRBG21], including semantic image segmentation, video instance segmentation, and object detection. In all of these applications, the central idea is to use a simple classification model to learn the map between uncertainty and the binary labels TP/FP. A related idea trying to exploit meta classification was introduced as "meta fusion" in [CRH+19]. The ability to discriminate TPs against FPs allows for an increase in the neural network's sensitivity, thereby producing an increase in FPs which are subsequently detected by a meta classifier decreasing the total number of errors made.

3 Methods

3.1 Uncertainty Quantification Protocols

Various methods for uncertainty or confidence estimation exist. While confidence scores are directly produced by a DNN, oftentimes other uncertainty-related quantities, such as softmax margin or entropy, can be generated from the network output. In variational inference (e.g., MC dropout, deep ensembles, or batch norm uncertainty), for a single network input, several predictions are made. Their variance or standard deviation is then interpreted as a measure of prediction uncertainty. Oftentimes, a common strategy for different kinds of uncertainty is missing and comparison between methods is highly context-dependent. Moreover, different methods to esti-

mate uncertainty may produce mutually redundant information which, for lack of comparability, cannot be established experimentally. We propose uncertainty aggregation methods as unifying strategy for different tasks.

Meta classification: In classification tasks, an input is categorized as one class out of a finite number of classes. Evaluation usually takes a binary value of either true or false. Meta classification can then be understood as the task of predicting the label true or false from uncertainty metrics. To this end, a lightweight classification model (e.g., logistic regression, a shallow neural network, or a gradient boosting classifier) is fitted to the training dataset, where uncertainty metrics can be computed and the actual, true label for each sample can be established. This can be regarded as *learning a binary decision rule* based on uncertainty information. On an evaluation dataset, such a meta classification model can be evaluated in terms of classification metrics like the area under the receiver operating characteristic *AuROC* or the area under precision-recall curve *AuPR* [DG06]. Different sources of uncertainty information, thus, serve as co-variables for meta classification models. Their performance can be regarded as a unified evaluation of confidence information. Network-intrinsic confidence scores naturally fit into this framework and can be either directly evaluated as any meta classification model or also serve as a single co-variable for such a model. In particular, different sources can be combined by fitting a meta classification model on the combined set of uncertainty metrics. This allows for the investigation of mutual redundancy between uncertainty information. Meta classification can be applied to any setting where a prediction can be marked as true or false and has served as a baseline for image classification [HG17], but also the tasks of semantic segmentation ("MetaSeg") [RCH+20] and object detection ("MetaDetect") [SKR20] can be subject to meta classification. In semantic segmentation, segments can be classified as true or false based on their intersection-over-union (IoU) with the ground truth being above a certain threshold. Similarly, bounding boxes are ordinarily declared as true or false predictions based on their maximum IoU with any ground truth box (see Sect. 3.2).

Meta fusion: Obtaining well-performing meta classifiers allows for their incorporation in the form of decision rules (replacing the standard score threshold) into the original prediction pipeline. Doing so is especially useful in settings where the DNN prediction is based on positive (foreground/detection) and negative (background) predictions, such as semantic segmentation and object detection. Implementing meta classifiers into such a framework is called meta fusion, whereby the improvement of the detection of false predictions over the DNN confidence is supplemented by increased prediction sensitivity of the DNN (see Fig. 2). Greater sensitivity tends to

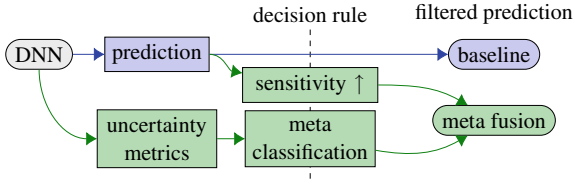


Fig. 2 Schematic sketch of the prediction baseline and the alternative meta fusion pipeline

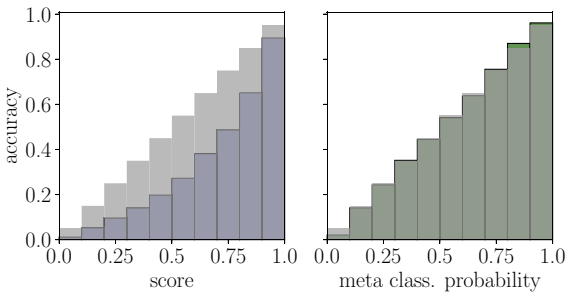


Fig. 3 Reliability plots (accuracy over confidence) for the confidence score and a gradient boosting meta classifier based on the confidence score (`Faster R-CNN` on $\mathcal{D}_{\text{val}}^{\text{COCO}}$)

reduce false negative (FN) predictions and *increase* true (TP) and false positive (FP) predictions. The meta classifier's task is then to identify the additional FPs as such and filter them from the prediction which leads to a net increase in prediction performance of the combination DNN with the meta classifier over the standalone DNN baseline. Such an increase can be evaluated utilizing metrics usually employed to assess the performance of a DNN in the respective setting, e.g., mean *IoU* in semantic segmentation [CRH+19] or mean average precision (*mAP*) in object detection [RRSG21].

Calibration: Confidence estimates with high predictive power (as measured in meta classification) have useful applications, e.g., in meta fusion where detection accuracy can be improved by investing uncertainty information. In addition to this advantage, there is another aspect to the probabilistic estimation of whether the corresponding prediction is correct or not. Confidence calibration tries to gauge the accuracy of the frequentistic interpretation, for example, from 100 examples each with a confidence of about $\frac{1}{4}$, statistically we expect about 25 of those examples to be true predictions.

This is often investigated by dividing the confidence range into bins of equal width and computing the accuracy of predictions conditioned to the respective bin (see Fig. 3). For calibrated confidences, the resulting distribution should then be close to the ideally calibrated diagonal. Meta classification is closely related to post-processing calibration methods in that a map is learned on a validation dataset which yields confidences with improved calibration. In particular, meta classification with the confidence score as the only co-variable is (up to the explicit optimization objective) isotonic regression as introduced in [ZE02].

Meta regression: In a regression task, a continuous variable (e.g., in $\mathbb{R}$) is estimated given an input. By the same logic as in meta classification, whenever the quality of the prediction of a DNN can be expressed as a real number, we may fit a simple regression model (e.g., linear regression, shallow neural network, or a gradient boosting regression model) on a training dataset where both uncertainty metrics and the actual prediction quality can be computed. This is called a meta regression model which maps uncertainty information to an (e.g., $\mathbb{R}$-valued) *estimation of the prediction quality of the DNN* (see Fig. 4). The relationship between actual and predicted quality can, again, be evaluated on a dedicated dataset in terms of the coefficient

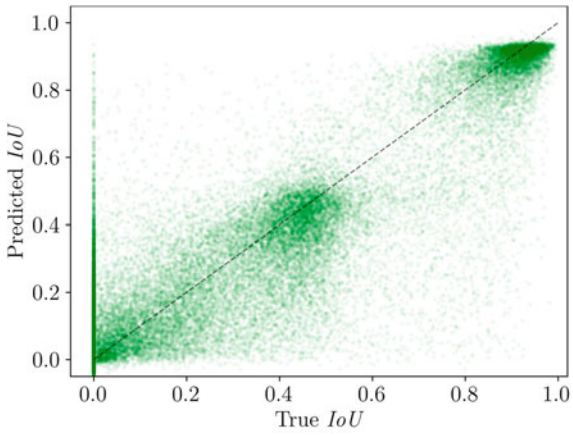


Fig. 4 Scatter plot of predicted bounding box *IoU* over true *IoU* of a gradient boosting meta regression model (`RetinaNet` on $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$)

of determination R^2, a well-established quality metric for regression models. Meta regression shares many features as an evaluation protocol as meta classification, perhaps with the exception that network confidence scores fit less directly into the meta regression logic. However, a meta regression model based on such a score still follows the same design. We note that fitting non-linear regression models (e.g., a neural network or a gradient boosting as opposed to a linear regression) on one variable tends to not significantly improve meta regression performance. The *IoU* or modifications thereof in semantic segmentation [RCH+20] or object detection [SKR20] are suitable quality measures to be used in meta regression.

3.2 Deep Object Detection Frameworks

We develop output- and gradient-based uncertainty metrics for the task of 2D bounding box ("object") detection on camera images $\mathbf{x} \in \mathbb{I}^{H \times W \times C}$, with $\mathbb{I} = [0, 1]$. A DNN tailored to this task usually produces a fixed number $N_{\text{out}} \in \mathbb{N}$ of output boxes

$$\mathfrak{O}(\boldsymbol{x}; \boldsymbol{\theta}) = \left(\mathfrak{O}^1(\boldsymbol{x}; \boldsymbol{\theta}), \ldots, \mathfrak{O}^{N_{\text{out}}}(\boldsymbol{x}; \boldsymbol{\theta})\right) \in (\mathbb{R}^4 \times \mathcal{S} \times \mathbb{I}^{N_{\text{C}}})^{N_{\text{out}}}, \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{N_{\text{DNN}}}$ with $N_{\text{DNN}} \in \mathbb{N}$ denotes the number of weights in the respective DNN architecture and $N_{\text{C}} \in \mathbb{N}$ is the fixed number of classes (the set of classes is denoted by $\mathcal{N}_{\text{C}} = \{1, \ldots, N_{\text{C}}\}$) to be learned. Each output box $\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta}) = (\hat{\xi}^j, \hat{s}^j, \hat{p}^j) \in \mathbb{R}^4 \times \mathcal{S} \times \mathbb{I}^{N_{\text{C}}}$ for $j \in \mathcal{N}_{\text{out}} = \{1, \ldots, N_{\text{out}}\}$ is encoded by

- Four localization variables, e.g., $\hat{\boldsymbol{\xi}} = (\hat{c}_{\text{min}}^j, \hat{r}_{\text{min}}^j, \hat{c}_{\text{max}}^j, \hat{r}_{\text{max}}^j) \in \mathbb{R}^4$ (top-left and bottom-right corner coordinates),
- Confidence score $\hat{s} \in \mathcal{S} = (0, 1)$ indicating the probability of an object existing at $\hat{\xi}^j$, and

- Class probability distribution $\hat{\mathbf{p}}^j = (\hat{p}_1^j, \ldots, \hat{p}_{N_C}^j) \in \mathbb{I}^{N_C}$.

The predicted class of $\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})$ is determined to be $\hat{\kappa}^j = \arg\max_{k \in \mathcal{N}_C} \hat{p}_k^j$. The N_{out} boxes subsequently undergo filtering mechanisms resulting in $|\mathcal{N}_{\mathbf{x}}| = \hat{N}_{\mathbf{x}}$ detected instances $\mathcal{N}_{\mathbf{x}} \subseteq \mathcal{N}_{\text{out}}$

$$\hat{\mathbf{y}} = \left(\mathfrak{B}^1(\mathbf{x}; \boldsymbol{\theta}), \ldots, \mathfrak{B}^{\hat{N}_{\mathbf{x}}}(\mathbf{x}; \boldsymbol{\theta}) \right) \in \mathbb{R}^{\hat{N}_{\mathbf{x}} \times (4+1+N_C)}. \tag{2}$$

Commonly, the much smaller number $\hat{N}_{\mathbf{x}} \ll N_{\text{out}}$ of predicted boxes is the result of two mechanisms. First, *score thresholding*, i.e., discarding any $\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})$ with $s^j < \varepsilon_s$ for some fixed threshold $\varepsilon_s > 0$, is performed as an initial distinction between "foreground" and "background" boxes. Afterward, the *non-maximum suppression* (NMS) algorithm is used to reduce boxes with the same class and significant mutual overlap to only one box. This way, for boxes that are likely indicating the same object in $\mathbf{x}$, there is only one representative in $\hat{y}$. By "overlap", we usually mean the intersection-over-union (*IoU*) between two bounding boxes $\hat{\boldsymbol{\xi}}^j$ and $\hat{\boldsymbol{\xi}}^k$ which is defined as

$$IoU\left(\hat{\boldsymbol{\xi}}^j, \hat{\boldsymbol{\xi}}^k\right) = \frac{\left|\hat{\boldsymbol{\xi}}^j \cap \hat{\boldsymbol{\xi}}^k\right|}{\left|\hat{\boldsymbol{\xi}}^j \cup \hat{\boldsymbol{\xi}}^k\right|}, \tag{3}$$

i.e., the ratio of their area of intersection and their joint area. The *IoU* between two boxes is always in $[0, 1]$, where 0 means no overlap and 1 means the boxes have identical localization. One also rates the localization quality of a prediction in terms of the maximum *IoU* it has with any ground truth box on $\mathbf{x}$ which has the same class.

The NMS algorithm is based on what we call "candidate boxes". An output instance $\mathfrak{O}^k(\mathbf{x}; \boldsymbol{\theta})$ is a *candidate box* for another output instance $\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})$ if it

1. has a sufficiently high score ($\hat{s}^k \geq \varepsilon_s$ above a chosen, fixed threshold $\varepsilon_s \geq 0$),
2. has the same class as $\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})$, i.e., $\hat{\kappa}^k = \hat{\kappa}^j$,
3. has sufficient *IoU* with $\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})$, i.e., $IoU(\hat{\boldsymbol{\xi}}^k, \hat{\boldsymbol{\xi}}^j) \geq \varepsilon_{IoU}$ for some fixed threshold $\varepsilon_{IoU} \geq 0$ (oftentimes $\varepsilon_{IoU} = \frac{1}{2}$).

We then denote by

$$\text{cand}\left[\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})\right] := \left\{\mathfrak{O}^k(\mathbf{x}; \boldsymbol{\theta}) : \mathfrak{O}^k(\mathbf{x}; \boldsymbol{\theta}) \text{ is a candidate box for } \mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})\right\} \tag{4}$$

the set of candidate boxes for $\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})$. In the NMS algorithm, all output boxes $\mathfrak{O}(\mathbf{x}; \boldsymbol{\theta})$ are sorted according to their score in descending order. Iteratively, the box $\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})$ with the highest score is selected as a prediction and cand$[\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})]$ is removed from the ranking of output boxes. This procedure is repeated until there are no boxes left. Note that NMS usually follows score thresholding so this stage may be reached quickly, depending on ε_s.

DNN object detection frameworks are usually trained in a supervised manner on images with corresponding labels or ground truth. The latter usually consist of a

number $N_{\mathbf{x}}$ of ground truth instances $\overline{\mathbf{y}} = (\overline{\mathbf{y}}^1, \ldots, \overline{\mathbf{y}}^{N_{\mathbf{x}}}) \in (\mathbb{R}^4 \times \mathcal{N}_{\mathrm{C}})^{N_{\mathbf{x}}}$ consisting of similar data as output boxes which we denote by the bar $\overline{\cdot}$. In particular, each ground truth instance $\overline{\mathbf{y}}^j$ has a specified localization $\overline{\boldsymbol{\xi}}^j = (\overline{c}^j_{\min}, \overline{r}^j_{\min}, \overline{c}^j_{\max}, \overline{r}^j_{\max}) \in \mathbb{R}^4$ and category $\kappa^j \in \mathcal{N}_{\mathrm{C}}$. From the network output $\mathfrak{O}(\mathbf{x}; \boldsymbol{\theta})$ on an image $\mathbf{x}$ and the ground truth $\overline{\mathbf{y}}$, a loss function $J(\mathfrak{O}(\mathbf{x}; \boldsymbol{\theta}), \overline{\mathbf{y}})$ is computed and iteratively minimized over $\boldsymbol{\theta}$ by stochastic gradient descent or some of its variants. In most object detection frameworks, J splits up additively into

$$J = J_{\boldsymbol{\xi}} + J_s + J_{\mathbf{p}}, \tag{5}$$

with $J_{\boldsymbol{\xi}}$ punishing localization mistakes, J_s punishing incorrect score assignments to boxes (incorrect boxes with high score and correct boxes with low score), and $J_{\mathbf{p}}$ punishing an incorrect class probability distribution, respectively. In standard gradient descent optimization, the weights $\boldsymbol{\theta}$ are then updated by the following rule:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \, \nabla_{\boldsymbol{\theta}} J \left(\mathfrak{O}(\mathbf{x}; \boldsymbol{\theta}), \overline{\mathbf{y}} \right), \tag{6}$$

where $\eta > 0$ is the either fixed or variable learning rate parameter. The learning gradient $\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \overline{\mathbf{y}}) := \nabla_{\boldsymbol{\theta}} J(\mathfrak{O}(\mathbf{x}; \boldsymbol{\theta}), \overline{\mathbf{y}})$ will play a central role for the gradient-based uncertainty metrics which will be introduced in Sect. 3.4.

3.3 Output-Based Uncertainty: MetaDetect

In this section, we construct uncertainty metrics for every $\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta}) \in \hat{\mathbf{y}}(\boldsymbol{x}, \boldsymbol{\theta})$. We do so in two stages, first by introducing the general metrics that can be obtained from the object detection pipeline. Second, we extend this by additional metrics that can be computed when using MC dropout. We consider a predicted bounding box $\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta}) \in \hat{\mathbf{y}}(\boldsymbol{x}, \boldsymbol{\theta})$ and its corresponding filtered candidate boxes $\mathfrak{O}^k(\mathbf{x}; \boldsymbol{\theta}) \in \mathrm{cand}[\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta})]$ that were discarded by the NMS.

The number of corresponding candidate boxes $\mathfrak{O}^k(\mathbf{x}; \boldsymbol{\theta}) \in \mathrm{cand}[\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta})]$ filtered by the NMS intuitively gives rise to the likelihood of observing a true positive. The more candidate boxes $\mathfrak{O}^k(\mathbf{x}; \boldsymbol{\theta})$ belong to $\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta})$, the more likely it seems that $\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta})$ is a true positive. We denote by $N^{(j)}$ the number of candidate boxes $\mathfrak{O}^k(\mathbf{x}; \boldsymbol{\theta})$ belonging to $\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta})$, but suppressed by NMS. We increment this number by 1 and also count in $\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta})$.

For a given image $\boldsymbol{x}$, we have the set of predicted bounding boxes $\hat{\mathbf{y}}(\boldsymbol{x}, \boldsymbol{\theta})$ and the ground truth $\overline{\mathbf{y}}$. As we want to calculate values that represent the quality of the prediction of the neural network, we first have to define uncertainty metrics for the predicted bounding boxes in $\hat{\mathbf{y}}(\boldsymbol{x}, \boldsymbol{\theta})$. For each $\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta}) \in \hat{\mathbf{y}}(\boldsymbol{x}, \boldsymbol{\theta})$, we define the following quantities:

- the number of candidate boxes $N^{(j)} \geq 1$ that belong to $\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta})$ (i.e., $\mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta})$ belongs to itself; one metric),

- the predicted box $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$ itself, i.e., the values of the tuple

$$\left(\hat{c}^j_{\min}, \hat{r}^j_{\min}, \hat{c}^j_{\max}, \hat{r}^j_{\max}, \hat{s}^j, \hat{p}^j_1, \ldots, \hat{p}^j_{N_C}\right) \in \mathbb{R}^4 \times \mathcal{S} \times \mathbb{I}^{N_C}, \tag{7}$$

  as well as $\sum_{i\in\mathcal{N}_C} \hat{p}^j_i \in \mathbb{R}$ whenever class probabilities are not normalized ($6 + N_C$ metrics),
- size $d = (\hat{r}^j_{\max} - \hat{r}^j_{\min}) \cdot (\hat{c}^j_{\max} - \hat{c}^j_{\min})$ and circumference $g = 2 \cdot (\hat{r}^j_{\max} - \hat{r}^j_{\min}) + 2 \cdot (\hat{c}^j_{\max} - \hat{c}^j_{\min})$ (two metrics),
- IoU^j_{pb}: the IoU of $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$ and the box with the second highest score that was suppressed by $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$. This value is zero if there are no boxes corresponding to $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$ suppressed by the NMS (i.e., $N^{(j)} = 1$; one metric),
- the minimum, maximum, arithmetic mean, and standard deviation for ($\hat{r}^j_{\min}$, $\hat{r}^j_{\max}$, $\hat{c}^j_{\min}$, $\hat{c}^j_{\max}$, $\hat{s}^j$), size d and circumference g from $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$ and all the filtered candidate boxes that were discarded from $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$ in the NMS (4×7 metrics),
- the minimum, maximum, arithmetic mean, and standard deviation for the IoU of $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$ and all the candidate boxes corresponding to $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$ that were suppressed in the NMS (four metrics),
- relative sizes $rd = d/g$, $rd_{\max} = d/g_{\min}$, $rd_{\min} = d/g_{\max}$, $rd_{\text{mean}} = d/g_{\text{mean}}$, and $rd_{\text{std}} = d/g_{\text{std}}$ (five metrics),
- the maximal IoU of $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$ and all ground truth boxes in $\overline{\mathbf{y}}$; this is not an input to a meta model but serves as the ground truth provided to the respective loss function.

Altogether, this results in $47 + N_C$ uncertainty metrics which can be aggregated further in a meta classifier or a meta regression model.

We now elaborate on how to calculate uncertainty metrics for every $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta}) \in \hat{\mathbf{y}}(\boldsymbol{x},\boldsymbol{\theta})$ when using MC dropout. To this end, we consider the bounding box $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta}) \in \hat{\mathbf{y}}(\boldsymbol{x},\boldsymbol{\theta})$ that was predicted without dropout and then we observe under dropout K times the output of the same anchor box that produced $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$ and denote them by $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})_1$, ..., $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})_K$. For these $K + 1$ boxes $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})$, $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})_1$, ..., $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta})_K$, we calculate the standard deviation for the localization variables, the objectness score, and class probabilities. This is done for every $\mathfrak{B}^j(\mathbf{x};\boldsymbol{\theta}) \in \hat{\mathbf{y}}(\boldsymbol{x},\boldsymbol{\theta})$ and results in $4 + 1 + N_C$ additional dropout uncertainty metrics which we denote by $\text{std}_{\text{MC}}(\cdot)$ for each of the respective $4 + 1 + N_C$ box features. Executing this procedure for all available test images, we end up with a structured dataset. Each row represents exactly one predicted bounding box and the columns are given by the registered metrics. After defining a training/test splitting of this dataset, we learn meta classification ($IoU > 0.5$ vs. $IoU \leq 0.5$) and meta regression (quantitative IoU prediction) from the training part of this data. All the presented metrics, except for the true IoU, can be computed without the knowledge of the ground truth. We now want to explore to which extent they are suited for the tasks of meta classification and meta regression.

3.4 Gradient-Based Uncertainty for Object Detection

In the setting provided in Sect. 3.2, a DNN learns from a data point $(\mathbf{x}, \overline{\mathbf{y}}) \in \mathbb{I}^{H\times W\times C} \times (\mathbb{R}^4 \times \mathcal{N}_C)^{N_\mathbf{x}}$ by computing the gradient $\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \overline{\mathbf{y}})$. The latter is subsequently used to iteratively adjust the current weights $\boldsymbol{\theta}$, for example, by the formula given in (6). As such, the quantity $\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \overline{\mathbf{y}})$ can be interpreted as *learning stress* on $\boldsymbol{\theta}$ induced by the training data $(\mathbf{x}, \overline{\mathbf{y}})$. We explain next how a related quantity gives rise to confidence information.

Some intuition about gradient uncertainty: Generally, the loss function J measures the "closeness" of output boxes to the ground truth $\overline{\mathbf{y}}$ and $\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \overline{\mathbf{y}})$ expresses the induced change in $\boldsymbol{\theta}$ for any deviation from the ground truth. If we assume that $\hat{\mathbf{y}}(\mathbf{x}; \boldsymbol{\theta})$ is close to $\overline{\mathbf{y}}$, only little change in $\boldsymbol{\theta}$ will be required in an update step. We, therefore, expect $\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \overline{\mathbf{y}})$ to be of comparably small magnitude. A DNN which is already "well-trained" in the ordinary sense is expected to express high confidence when its prediction is correct for all practical purposes. Conversely, if the network output deviates from the ground truth significantly, learning on $(\mathbf{x}, \overline{\mathbf{y}})$ induces a larger adjustment in $\boldsymbol{\theta}$, leading to a gradient of larger magnitude. In this situation, the confidence in this prediction should be small.

The gradient $\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \overline{\mathbf{y}})$ is not realizable as a measure of uncertainty or confidence, since it uses the ground truth $\overline{\mathbf{y}}$, which is not accessible during inference time. An approach to circumvent this shortcoming was presented in [ORG18], in that the ground truth $\overline{\mathbf{y}}$ was replaced by the prediction of the DNN $\hat{\mathbf{y}}$ on $\mathbf{x}$. We format the prediction so that it has the same format as the ground truth. Particularly, in the initial application to image classification, the softmax output of the DNN was collapsed by an arg max to the predicted class before insertion into J. With respect to bounding box localization, there is no adjustment required. Additionally, when computing

$$\mathbf{g}\left(\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{y}}\right) = \nabla_{\boldsymbol{\theta}} J\left(\mathfrak{O}(\mathbf{x}; \boldsymbol{\theta}), \hat{\mathbf{y}}\right), \tag{8}$$

we disregard the dependency of $\hat{\mathbf{y}}$ on $\boldsymbol{\theta}$, thereby sticking to the design motivation from above. The quantity in (8) represents the weight adjustment induced in the DNN when learning that its prediction on $\mathbf{x}$ is correct. *Uncertainty* or *confidence information* is distilled from $\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{y}})$ by taking it to a scalar. To this end, it is natural to employ norms, e.g., $\|\cdot\|_1$ or $\|\cdot\|_2$, but we also use other maps such as minimal and maximal entry as well as the arithmetic mean and standard deviation over the gradient's entries (see (9)). Gradient uncertainty metrics allow for some additional flexibility. Regarding (8), we see that if J splits into additive terms (such as in the object detection setting, see (5)), gradient metrics allow the extraction of gradient uncertainty from each individual contribution. Additionally, we can restrict the variables for which we compute partial derivatives. One might, for example, be interested in computing the gradient $\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}_\ell, \hat{\mathbf{y}})$ for the weights $\boldsymbol{\theta}_\ell$ from only one network layer ℓ. In principle, this also allows for computing uncertainty metrics for individual convolutional filters or even individual weights, thus, also offering a way of localizing uncertainty within the DNN architecture.

$$m_1^{\boldsymbol{\theta}}(J) = \|\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{y}})\|_1, \quad m_2^{\boldsymbol{\theta}}(J) = \|\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{y}})\|_2, \quad m_{\max}^{\boldsymbol{\theta}}(J) = \max(\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{y}})),$$
$$m_{\min}^{\boldsymbol{\theta}}(J) = \min(\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{y}})), \quad m_{\text{std}}^{\boldsymbol{\theta}}(J) = \text{std}(\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{y}})), \quad m_{\text{mean}}^{\boldsymbol{\theta}}(J) = \text{mean}(\mathbf{g}(\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{y}})). \tag{9}$$

Application to instance-based settings: The approach outlined above requires some adaptation in order to produce meaningful results for settings in which the network output $\mathfrak{O}(\mathbf{x}; \boldsymbol{\theta})$ and the ground truth $\overline{\mathbf{y}}$ consist of several distinct instances, e.g., as in object detection. In such a situation, each component $\mathfrak{O}^j(\mathbf{x}; \boldsymbol{\theta})$ needs to be assigned individual confidence metrics. If we want to estimate the confidence for $\hat{\mathbf{y}}^j := \mathfrak{B}^j(\mathbf{x}; \boldsymbol{\theta})$, we may regard $\hat{\mathbf{y}}^j$ as the ground truth replacement entering the second slot of J. Then, the first argument of J needs to be adjusted according to $\hat{\mathbf{y}}^j$. This is necessary because the loss $J(\mathfrak{O}(\mathbf{x}; \boldsymbol{\theta}), \hat{\mathbf{y}})$ expresses the mistakes made by all the output boxes, i.e., the prediction of instances which actually appear in the ground truth $\overline{\mathbf{y}}$ but which have nothing to do with $\hat{\mathbf{y}}^j$ would be punished. The corresponding gradient would, therefore, misleadingly adjust $\boldsymbol{\theta}$ toward forgetting to see such ground truth instances. Thus, it is a natural idea to identify cand$[\hat{\mathbf{y}}^j]$ in the network output and only enter those boxes into J which likely indicate one and the same instance on $\mathbf{x}$. We define the corresponding gradient

$$\mathbf{g}^{\text{cand}}\left(\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{y}}^j\right) := \nabla_{\boldsymbol{\theta}} J\left(\text{cand}[\hat{\mathbf{y}}^j](\mathbf{x}, \boldsymbol{\theta}), \hat{\mathbf{y}}^j\right), \tag{10}$$

which serves as the basis of gradient uncertainty in instance-based settings. The flexibility mentioned in the previous paragraph carries over to this definition and will be exploited in our experiments.

4 Experimental Setup

In this section, we explain our choice of datasets, models, and metrics as well as experimental setup and implementation details.

4.1 Databases, Models, and Metrics

We investigate three different object detection datasets in the Pascal VOC [EVGW+15], the MS COCO [LMB+14], and the KITTI vision benchmark [GLSU15]. Pascal VOC $\mathcal{D}^{\text{VOC12}}$ and MS COCO $\mathcal{D}^{\text{COCO17}}$ are vision datasets containing images of a wide range of mostly everyday scenarios with 20 and 80 annotated classes, respectively. They both possess training and dedicated evaluation splits for testing and constitute established benchmarks in object detection. In order to test our methods in driving scenarios, we also investigate the KITTI dataset $\mathcal{D}^{\text{KITTI}}$ which shows different German urban environments and comes with annotated training

Table 1 Dataset split sizes for training and testing

Dataset	$\mathcal{D}_{\text{train}}^{\text{VOC12}}$	$\mathcal{D}_{\text{test}}^{\text{VOC12}}$	$\mathcal{D}_{\text{train}}^{\text{COCO17}}$	$\mathcal{D}_{\text{val}}^{\text{COCO17}}$	$\mathcal{D}_{\text{train}}^{\text{KITTI}}$	$\mathcal{D}_{\text{eval}}^{\text{KITTI}}$
Size	14,805	4,952	118,287	5,002	5,481	2,000

images. From the annotated training set, we take 2,000 randomly chosen images for evaluation. An overview of the sizes of the respective training and evaluation datasets can be found in Table 1.

The experiments we present here have been carried out based on a `YOLOv3` [RF18] re-implementation in `PyTorch`. We have trained our model from scratch on each of the three training splits under dropout with a probability 0.5 between the last and the second-to-last convolution layer in each of the regression heads. As meta classification and meta regression models, we employ the gradient boosting models in [CG16] with standard settings.

For the evaluation of our meta classification models, we use cross-validated area-under-curve metrics (*AuROC* and *AuPR* [DG06]) instead of accuracy, since the former measure the classification quality independently of any threshold, whereas accuracy only reflects the decisions made for the classification probability 0.5. For cross-validation, we use 10 individual image-wise splits of uncertainty metrics for the training of meta classifiers and the respective complementary split for evaluation. This particular procedure is employed whenever cross-validation is used in the following experiments. Especially in the VOC and KITTI dataset, *AuROC* values are mostly over 0.9, so we regard *AuPR* as a second classification metric. Object detection quality in meta fusion experiments will on the one hand be evaluated in terms of cross-validated mean average precision (*mAP*) [EVGW+15]. However, as the authors of [RF18] recognized *mAP* as computed in the VOC challenge is insensitive to FPs, which is why, on the other hand, we compute the cross-validated mean F_1 score (mF_1) as well. For the precision-recall curves of each class, we compute individual F_1 scores which are sensitive to FPs and we average them over all classes as a complementary metric to *mAP*. Confidence calibration is usually evaluated in terms of expected or maximum calibration error (*MCE*) [NCH15], computed over bins of width 0.1. As the expected calibration error is sensitive to the box count per bin, we instead regard in addition to the *MCE* the average calibration error (*ACE*) [NZV18]. Meta regression quality is evaluated by the usual coefficient of determination R^2.

4.2 Implementation Details

Our object detection models receive a dropout layer between the second-to-last and the last layers in each of the three `YOLOv3` detection heads. Dropout is active during training with a rate of 0.5 and also during MC inference, where we take standard deviations over 30 dropout samples for each of the $4 + 1 + N_C$ instance features of

all output boxes. The MetaDetect metrics introduced in Sect. 3.3 are computed from a score threshold of $\varepsilon_s = 0.0$, as it has been found in [SKR20] that lower thresholds lead to better performance in meta classification and meta regression.

Gradient uncertainty metrics were computed for the weights in the second-to-last ("$T-1$") and the last ("T") convolutional layers in the `YOLOv3` architecture at the same candidate score threshold as for the MetaDetect metrics. As there are three detection heads, corresponding to a 76×76 ("S"), a 38×38 ("M"), and a 19×19 ("L") cell grid, we also compute gradient uncertainty metrics individually for each detection head. We use this distinction in our notation and, for example, indicate the set of parameters from the last layer (T) of the detection head producing the 76×76 cell grid (S) by $\boldsymbol{\theta}(T, \mathrm{S})$. Moreover, as indicated in Sect. 3.4, we also exploit the split of the loss function in (5). Each of the computed $2 \times 3 \times 3 = 18$ gradients per box results in the 6 uncertainty metrics presented in (9) giving a total of 108 gradient uncertainty metrics per bounding box. Due to the resulting computational expense, we only compute gradient metrics for output boxes with score values $\hat{s} \geq 10^{-4}$ and regard only those boxes for all of the following experiments.

4.3 Experimental Setup and Results

Starting from pre-trained models, we compute the DNN output, as well as MC dropout, MetaDetect, and gradient uncertainty metrics on each evaluation dataset (see Table 1). For each output box, we also compute the maximum IoU with the respective ground truth such that the underlying relation can be leveraged. Before fitting any model, we perform NMS on the output boxes which constitutes the relevant case for our meta fusion investigation. In the latter, we are primarily interested in finding true boxes which are usually discarded due to their low assigned score. We compare the performance of different sets of uncertainty metrics in different disciplines. In the following, "Score" denotes the network's confidence score, "MC" stands for MC dropout, and "MD" the MetaDetect uncertainty metrics as discussed in Sect. 3.3. Moreover, "G" denotes the full set of gradient uncertainty metrics from Sect. 3.4 and the labels "MD+MC", "G+MC", "MD+G", and "MD+G+MC" stand for the respective unions of MC dropout, MetaDetect, and gradient uncertainty metrics.

Meta classification: We create 10 splits of the dataset of uncertainty metrics and computed IoU by randomly dividing it 10 times in half. We assign the TP label to those examples which have $IoU > 0.5$ and the FP label otherwise. On each split, we fit a gradient boosting classifier to the respective set of uncertainty metrics (see Fig. 5) on one half of the data. The resulting model is used to predict a classification probability on the uncertainty metrics of the second half of the data and vice-versa. This results in meta classification predictions on all data points. We evaluate these in terms of the previously introduced area-under-curve metrics under usage of the TP/FP labels generated from the ground truth, where we obtain averages and standard deviations from the 10 splits shown in Fig. 5. Performance measured in either of the two area-

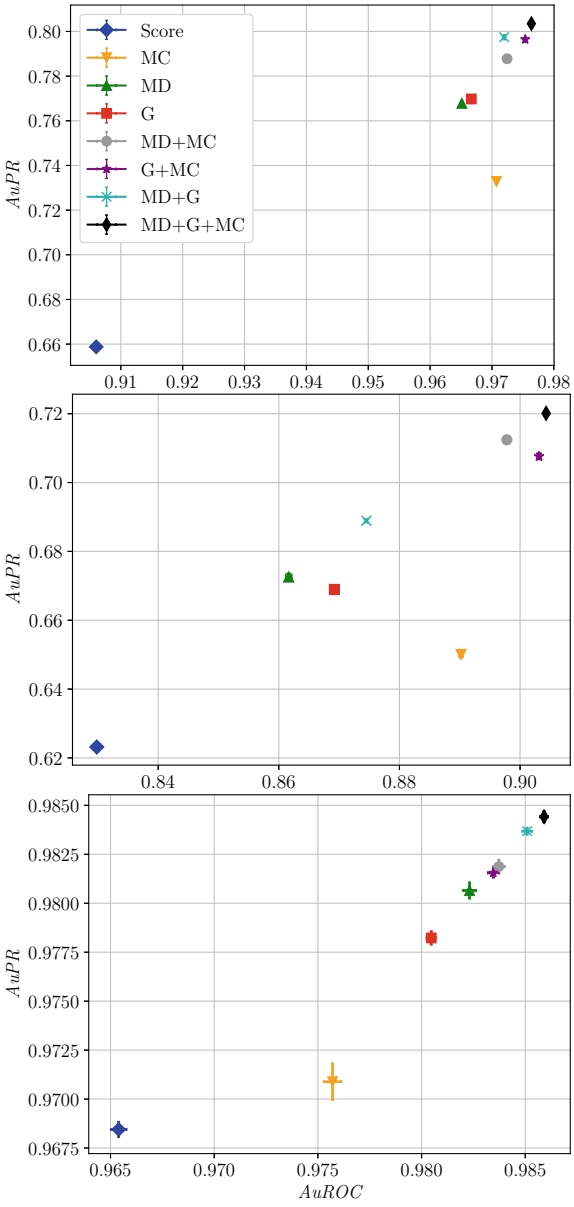


Fig. 5 Score and meta classification *AuPR* over *AuROC* from 10-fold cross-validation with respective standard deviation error bars for $\mathcal{D}_{\text{test}}^{\text{VOC12}}$, $\mathcal{D}_{\text{val}}^{\text{COCO17}}$, and $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$ (from top to bottom)

under-curve metrics suggests that the models situated in the top-right separate TP from FP best. In all datasets, the meta classifiers MD, G, and combinations thereof outperform the network confidence by a large margin (upwards of 6 *AuROC* percent points (ppts) on $\mathcal{D}_{\text{test}}^{\text{VOC12}}$, 2.5 on $\mathcal{D}_{\text{val}}^{\text{COCO17}}$, and 1.5 on $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$), with MC dropout being among the weakest meta classifiers. Note that for both $\mathcal{D}_{\text{test}}^{\text{VOC12}}$ (top) and $\mathcal{D}_{\text{val}}^{\text{COCO17}}$ (center), the standard deviation bars are occluded by the markers due to the size of the range. On $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$, we can see some of the error bars and the models forming a strict hierarchy across the two metrics with overlapping MD+MC and G+MC. Overall, the three sources of uncertainty MC, MD, and G show a significant degree of non-redundancy across the three datasets with mutual boosts of up to 7 *AuPR* ppts on $\mathcal{D}_{\text{test}}^{\text{VOC12}}$, 4.5 on $\mathcal{D}_{\text{val}}^{\text{COCO17}}$, and 1.1 on $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$.

Meta fusion: We use the 10 datasets of uncertainty metrics and resulting confidences as in meta classification in combination with the bounding box information as new post-NMS output of the network. Thresholding on the respective confidence on decision thresholds $\varepsilon_{\text{dec}} \in (0, 1)$ with a fixed step width of 0.01 reproduces the baseline object detection pipeline in the case of the score. The network prediction with assigned confidence can then be evaluated in terms of the aforementioned class-averaged performance measures for which we obtain averages and standard deviations from the 10 splits, except for the deterministic score baseline. In our comparison, we focus on the score baseline, MC dropout baseline, output- and gradient-based uncertainty metrics (MD and G), and the full model based on all available uncertainty metrics (G+MD+MC). We show the resulting curves of *mAP* over mF_1 in Fig. 6 on the right, where we see that the maximum mF_1 value is attained by the confidence score. In addition, we observe a sharp bend in the score curve resulting from a large count of samples with a score between 0 and 0.01, where the interpolation to the threshold $\varepsilon_s = 0$ reaches to 0.912 *mAP* at a low 0.61 mF_1 (outside the plotting range). In terms of *mAP*, however, the score is surpassed by the meta fusion models G and in particular the meta classification models involving MD (maximum *mAP* close to 0.92 at around 0.81 mF_1). The raw MetaDetect confidences seem

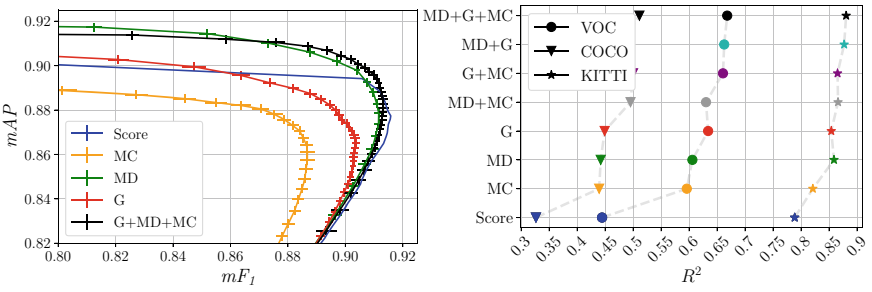


Fig. 6 *Left: mAP* over mF_1 score for five meta fusion models computed from 10-fold cross-validation. We show standard deviation error bars obtained from cross-validation. *Right:* meta regression R^2 for uncertainty models on $\mathcal{D}_{\text{test}}^{\text{VOC12}}$, $\mathcal{D}_{\text{val}}^{\text{COCO17}}$, and $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$. The plot shows the averages over 10-fold cross-validation with standard deviation error bars hidden by the symbols

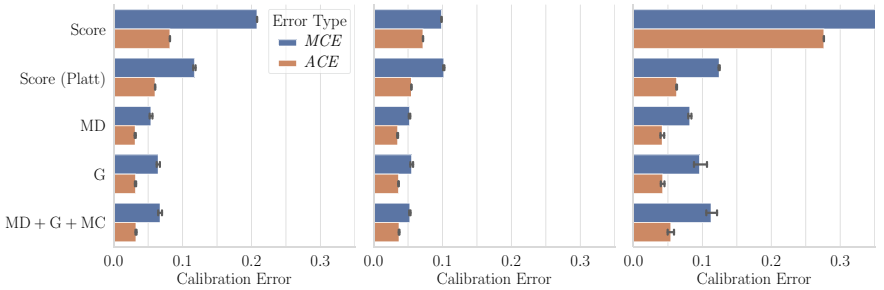


Fig. 7 Average *ACE* and mean calibration error *MCE* of the score and meta classification confidences for $\mathcal{D}_{\text{test}}^{\text{VOC12}}$, $\mathcal{D}_{\text{val}}^{\text{COCO17}}$, and $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$ (from left to right)

to slightly improve over G+MD+MC in terms of *mAP* by about 0.3 ppts., which may be due to overfitting of the gradient boosting model, but could also result from the altered confidence ranking which is initially performed when computing mean average precision. Meta fusion based on MC dropout uncertainty shows the worst performance of all four methods considered in this test.

Calibration: From the generated meta classification probabilities and the score, calibration errors can be computed indicating their calibration as binary classifiers and compared; see Fig. 7 with standard deviations indicated by error bars. In addition, we show the Platt-scaled confidence score as a reference for a post-processing calibration method. Naturally, *ACE* is smaller than *MCE*. The meta classification models show consistently far-smaller calibration errors (maximum *MCE* of 0.11 on $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$) than the confidence score (minimum *MCE* of 0.1 on $\mathcal{D}_{\text{val}}^{\text{COCO17}}$) with comparatively small miscalibration on the COCO validation dataset (center). We also observe that meta classifiers are also better calibrated than the Platt-scaled confidence score with the smallest margin on $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$. Note from the *MCE* that meta classification models show generally good calibration with a maximum *ACE* around 0.05 as indicated in Sect. 3 and by the example of Fig. 3.

Meta regression: By an analogous approach to meta classification, we can fit gradient boosting regression models in a cross-validated fashion to sets of uncertainty metrics and the true maximum *IoU* with the ground truth. The resulting predicted *IoU* values can be evaluated in terms of R^2 which is illustrated in Fig. 6 on the left. Note once again that error bars for the standard deviation are covered by the markers. We observe an overall similar trend of the metrics to the meta classification performance in that combined models tend to perform better (boosts of up to 6 R^2 ppts.) indicating mutual non-redundancy among the different sources of uncertainty. The models based on MD and G individually show significant improvements over the confidence score (over 5 R^2 ppts. on all datasets) and a slight improvement over MC (up to 3 R^2 ppts.) for all investigated datasets.

Parameter importance: We investigate which parameters are most important for meta classification in order to find the best performance with as few metrics as

Table 2 Cumulative parameter ranking in terms of their importance for *AuROC* and *AuPR* obtained from a greedy search for the first 9 chosen parameters. Metrics indicated are averages obtained from cross-validation. We show the performance of the models with all metrics as co-variables for reference on each of the datasets $\mathcal{D}_{\text{test}}^{\text{VOC12}}$, $\mathcal{D}_{\text{val}}^{\text{COCO17}}$, and $\mathcal{D}_{\text{eval}}^{\text{KITTI}}$

VOC				COCO				KITTI			
AuROC		*AuPR*		*AuROC*		*AuPR*		*AuROC*		*AuPR*	
0.917	$\text{std}_{\text{MC}}(\hat{s})$	0.658	$\hat{s}$	0.830	$\hat{s}$	0.623	$\hat{s}$	0.966	$\hat{s}$	0.969	$\hat{s}$
0.959	$m_{\text{mean}}^{\boldsymbol{\theta}(T,\text{S})}(J_s)$	0.740	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{S})}(J_p)$	0.875	$\text{std}_{\text{MC}}(\hat{s})$	0.664	$\text{std}_{\text{MC}}(\hat{s})$	0.978	$\hat{c}_{\text{min}}$	0.976	$\hat{c}_{\text{min}}$
0.966	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{S})}(J_p)$	0.769	$\text{std}_{\text{MC}}(\hat{s})$	0.886	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{L})}(J_p)$	0.685	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{S})}(J_p)$	0.981	$\text{std}_{\text{MC}}(\hat{s})$	0.980	$m_{1}^{\boldsymbol{\theta}(T-1,\text{L})}(J_p)$
0.971	$m_{\text{std}}^{\boldsymbol{\theta}(T,\text{M})}(J_p)$	0.784	$m_{\text{mean}}^{\boldsymbol{\theta}(T,\text{M})}(J_p)$	0.893	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{S})}(J_p)$	0.704	$m_{\text{max}}^{\boldsymbol{\theta}(T-1,\text{M})}(J_p)$	0.983	$m_{\text{std}}^{\boldsymbol{\theta}(T-1,\text{L})}(J_p)$	0.981	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{M})}(J_p)$
0.972	$\hat{c}_{\text{min}}$	0.790	$m_{\text{mean}}^{\boldsymbol{\theta}(T-1,\text{S})}(J_\xi)$	0.899	$m_{\text{max}}^{\boldsymbol{\theta}(T-1,\text{M})}(J_p)$	0.710	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{L})}(J_p)$	0.984	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{M})}(J_\xi)$	0.982	$\hat{r}_{\text{max}}$
0.974	$\hat{s}$	0.795	$\hat{c}_{\text{min}}$	0.900	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{M})}(J_p)$	0.713	$\text{std}_{\text{MC}}(\hat{c}_{\text{min}})$	0.985	$\hat{r}_{\text{max}}$	0.983	$\sum_i \hat{p}_i$
0.975	$m_{\text{mean}}^{\boldsymbol{\theta}(T,\text{S})}(J_\xi)$	0.799	$\text{std}_{\text{MC}}(\hat{c}_{\text{min}})$	0.901	$m_{\text{max}}^{\boldsymbol{\theta}(T-1,\text{L})}(J_s)$	0.715	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{M})}(J_p)$	0.985	$m_{\text{max}}^{\boldsymbol{\theta}(T,\text{S})}(J_p)$	0.984	$\text{std}_{\text{MC}}(\hat{s})$
0.975	$\hat{r}_{\text{min}}$	0.802	$m_{\text{mean}}^{\boldsymbol{\theta}(T,\text{S})}(J_\xi)$	0.902	$\text{std}_{\text{MC}}(\hat{c}_{\text{min}})$	0.716	$\text{std}_{\text{MC}}(\hat{r}_{\text{max}})$	0.986	$\hat{p}_1$	0.984	$m_{2}^{\boldsymbol{\theta}(T,\text{S})}(J_\xi)$
0.976	$\hat{c}_{\text{max}}$	0.803	$m_{\text{max}}^{\boldsymbol{\theta}(T-1,\text{M})}(J_p)$	0.902	$m_{\text{mean}}^{\boldsymbol{\theta}(T,\text{S})}(J_s)$	0.717	$m_{2}^{\boldsymbol{\theta}(T,\text{S})}(J_p)$	0.986	$m_{\text{mean}}^{\boldsymbol{\theta}(T-1,\text{L})}(J_\xi)$	0.984	$m_{\text{mean}}^{\boldsymbol{\theta}(T-1,\text{L})}(J_\xi)$
0.976	MD+G+MC	0.803	MD+G+MC	0.904	MD+G+MC	0.720	MD+G+MC	0.986	MD+G+MC	0.984	MD+G+MC

possible. We pursue a greedy approach and evaluate single-metric models at first in terms of *AuROC* and *AuPR* in the cross-validated manner used before. We then fix the best-performing single metric and investigate two-metric combinations and gather the resulting metrics ranking with their area-under-curve score in Table 2. In addition to the nested models, we also show the model using all metrics which is also depicted in Fig. 5. Note that the numbers represent cross-validated averages, where we choose not to show standard deviations as we are mainly interested in seeing which metrics combine to yield well-separating meta classifiers. On all datasets, we find that a total of 9 metrics suffices to come close to or reach the performance of the best model in question. The network confidence appears as the most informative single metric 5 out of 6 times. Another metric that leads to early good improvements is the related MC dropout standard deviation of the confidence score. Other noteworthy metrics are the left border coordinate $\hat{c}_{\text{min}}$ and gradient metrics for the class probability contribution J_p. Overall, gradient metrics contribute a large amount of information to $\hat{s}$ and $\text{std}_{\text{MC}}(\hat{s})$ which is indicative to meta classification performance.

Conclusions and Outlook

In this chapter, we have outlined two largely orthogonal approaches to quantifying epistemic uncertainty in deep object detection networks, the output-based MetaDetect approach, and gradient-based uncertainty metrics. Both compare favorably to the network confidence score and MC dropout in terms of meta classification and meta regression with the additional benefit that meta classifiers give rise to well-calibrated confidences irrespective of the set of metrics they are based on. Moreover, different sources of uncertainty information lead to additional boosts when combined, indicating mutual non-redundancy. We have seen that the meta fusion approach enables us to trade uncertainty information for network performance in terms of mean average precision with the best meta fusion models involving the MetaDetect metrics. In terms of raw meta classification capabilities, well-performing models can be achieved by only regarding a small fraction of all metrics, with gradient-based uncertainty metrics contributing highly informative components. The design of well-performing meta classifiers and meta regression models opens up possibilities for further applications. On the one hand, the implementation of meta classification models into an active learning cycle could potentially lead to a drastic decrease in data annotation costs. On the other hand, meta classification models may be applicable beneficially in determining the labeling quality of datasets and detecting labeling mistakes.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI-Absicherung)". The authors would like to thank the consortium for the successful cooperation. Furthermore, we gratefully acknowledge financial support from the state Ministry of Economy, Innovation and Energy of Northrhine Westphalia (MWIDE) and the European Fund for Regional Development via the FIS.NRW project BIT-KI, grant no. EFRE-0400216. The authors gratefully acknowledge the Gauss

Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

References

[BCR+20] D. Brüggemann, R. Chan, M. Rottmann, H. Gottschalk, S. Bracke, Detecting out of distribution objects in semantic segmentation of street scenes, in *Proceedings of the European Safety and Reliability Conference (ESREL)*, virtual conference (November 2020), pp. 3023–3030

[CCKL19] J. Choi, D. Chun, H. Kim, H.-J. Lee, Gaussian YOLOv3: an accurate and fast object detector using localization uncertainty for autonomous driving, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, June 2019), pp. 502–511

[CCLK20] J. Choi, D. Chun, H.-J. Lee, H. Kim, Uncertainty-based object detector for autonomous driving embedded platforms, in *Proceedings of the IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, virtual conference (August 2020), pp. 16–20

[CG16] T. Chen, C. Guestrin, XGBoost: a scalable tree boosting system, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (August 2016), pp. 785–794

[CRG21] R. Chan, M. Rottmann, H. Gottschalk, Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, virtual conference (October 2021), pp. 5128–5137

[CRGR21] P. Colling, L. Roese-Koerner, H. Gottschalk, M. Rottmann, MetaBox+: a new region based active learning method for semantic segmentation using priority maps, in *Proceedings of the International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, virtual conference (February 2021), pp. 51–62

[CRH+19] R. Chan, M. Rottmann, F. Hüger, P. Schlicht, H. Gottschalk, Controlled false negative reduction of minority classes in semantic segmentation, in *2020 International Joint Conference on Neural Networks (IJCNN)* (IEEE, July 2020), pp. 1–8

[DG06] J. Davis, M. Goadrich, The relationship between precision-recall and ROC curves, in *Proceedings of the International Conference on Machine Learning (ICML)* (Pittsburgh, PA ,USA, June 2006), pp. 233–240

[DL90] J.S. Denker, Y. LeCun, Transforming neural-net output levels to probability distributions, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Denver, CO, USA, November 1990), pp. 853–859

[EVGW+15] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The Pascal visual object classes challenge: a retrospective. Int. J. Comput. Vis. (IJCV) **111**(1), 98–136 (2015)

[Gal17] Y. Gal, *Uncertainty in Deep Learning*. Dissertation (University of Cambridge, 2017)

[GG16] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, in *Proceedings of the International Conference on Machine Learning (ICML)* (New York, NY, USA, June 2016), pp. 1050–1059

[GLSU15] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, The KITTI Vision Benchmark Suite (2015). [Online; accessed 2021-11-18]

[GPSW17] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, in *Proceedings of the International Conference on Machine Learning (ICML)* (Sydney, NSW, Australia, August 2017), pp. 1321–1330

[GSS15] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in *Proceedings of the International Conference on Learning Representations (ICLR)* (San Diego, CA, USA, 2015), pp. 1–11

[HG17] D. Hendrycks, K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Toulon, France, April 2017), pp. 1–12

[HSW20] A. Harakeh, M. Smart, S.L. Waslander, BayesOD: a Bayesian approach for uncertainty estimation in deep object detectors, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, virtual conference (May 2020), pp. 87–93

[HVD14] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops* (Montréal, QC, Canada, December 2014), pp. 1–9

[HW21] Eyke Hüllermeier, Willem Waegeman, Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. Mach. Learn. **110**, 457–506 (2021)

[HZW+19] Y. He, C. Zhu, J. Wang, M. Savvides, X. Zhang, Bounding box regression with uncertainty for accurate object detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, May 2019), pp. 2888–2897

[JLM+18] B. Jiang, R. Luo, J. Mao, T. Xiao, Y. Jiang, Acquisition of localization confidence for accurate object detection, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Munich, Germany, September 2018), pp. 784–799

[KD19] F. Kraus, K. Dietmayer, Uncertainty estimation in one-stage object detection, in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)* (Auckland, New Zealand, October 2019), pp. 53–60

[KG17] A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision? in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Long Beach, CA, USA, December 2017), pp. 5574–5584

[KKSH20] F. Küppers, J. Kronenberger, A. Shantia, A. Haselhoff, Multivariate confidence calibration for object detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, virtual conference (June 2020), pp. 1322–1330

[KP20] A. Kumar, B. Poole, On implicit regularization in β-VAEs, in *Proceedings of the International Conference on Machine Learning (ICML)* (July 2020), pp. 5480–5490

[KRBG21] K. Kowol, M. Rottmann, S. Bracke, H. Gottschalk, YOdar: uncertainty-based sensor fusion for vehicle detection with camera and radar sensors, in *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, virtual conference (February 2021), pp. 177–186

[LAE+16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Amsterdam, The Netherlands, October 2016), pp. 21–37

[LDBK18] M. Truong Le, F. Diehl, T. Brunner, A. Knol,, Uncertainty estimation for deep neural object detectors in safety-critical applications, in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)* (Maui, HI, USA, November 2018), pp. 3873–3878

[LGG+17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Venice, Italy, October 2017), pp. 2980–2988,

[LGRB20] Z. Lyu, N. Gutierrez, A. Rajguru, W.J. Beksi, Probabilistic object detection via deep ensembles, in *Proceedings of the European Conference on Computer Vision (ECCV)*, virtual conference (August 2020), pp. 67–75

[LHK+21] Y. Lee, J.-w. Hwang, H.-I. Kim, K. Yun, J. Park, S. Ju Hwang, Localization Uncertainty Estimation for Anchor-Free Object Detectors (September 2021), pp. 1–10. arXiv:2006.15607

[LMB+14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. Lawrence Zitnick, Microsoft COCO: common objects in context, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Zurich, Switzerland, September 2014), pp. 740–755

[LPB17] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Long Beach, CA, USA, December 2017), pp. 6402–6413

[Mac92] D.J.C. MacKay, A practical Bayesian framework for backpropagation networks. Neural Comput. **4**(3), 448–472 (1992)

[MNDS18] D. Miller, L. Nicholson, F. Dayoub, N. Sünderhauf, Dropout sampling for robust object detection in open-set conditions, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, QLD, Australia, May 2018), pp. 3243–3249

[MRG20] K. Maag, M. Rottmann, H. Gottschalk, Time-dynamic estimates of the reliability of deep semantic segmentation networks, in *Proceedings of the IEEE International Conference on Tools With Artificial Intelligence (ICTAI)* virtual conference (November 2020), pp. 502–509

[MRV+21] K. Maag, M. Rottmann, S. Varghese, F. Hüger, P. Schlicht, H. Gottschalk, Improving video instance segmentation by light-weight temporal uncertainty estimates, in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (July 2021), pp. 1–8

[NCH15] M. Pakdaman Naeini, G. Cooper, M. Hauskrecht, Obtaining well calibrated probabilities using Bayesian binning, in *Proceedings of the AAAI Conference on Artificial Intelligence* (Austin, TX, USA, January 2015), pp. 2901–2907

[NMC05] A. Niculescu-Mizil, R. Caruana, Predicting good probabilities with supervised learning, in *Proceedings of the International Conference on Machine Learning (ICML)* (Bonn, Germany, August 2005), pp. 625–632

[NZV18] L. Neumann, A. Zisserman, A. Vedaldi, Relaxed softmax: efficient confidence auto-calibration for safe pedestrian detection, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops* (Montréal, QC, Canada, December 2018), pp. 1–8

[ORF20] P. Oberdiek, M. Rottmann, G.A. Fink, Detection and retrieval of out-of-distribution objects in semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (June 2020), pp. 1331–1340

[ORG18] P. Oberdiek, M. Rottmann, H. Gottschalk, Classification uncertainty of deep neural networks based on gradient information, in *Proceedings of the IAPR TC3 Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR)* (Siena, Italy, September 2018), pp. 113–125

[Pla99] J. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in *Advances in Large Margin Classifiers*. ed. by A.J. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (MIT Press, 1999), pp. 61–74

[RCH+20] M. Rottmann, P. Colling, T.-P. Hack, R. Chan, F. Hüger, P. Schlicht, H. Gottschalk, Prediction error meta classification in semantic segmentation: detection via aggregated dispersion measures of softmax probabilities, in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, virtual conference (July 2020), pp. 1–9

[RF18] J. Redmon, A. Farhadi, Yolov3: An Incremental Improvement (April 2018), pp. 1–6. arXiv:1804.02767

[RHGS15] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, December 2015), pp. 91–99

[RMC+20] M. Rottmann, K. Maag, R. Chan, F. Hüger, P. Schlicht, H. Gottschalk, Detection of false positive and false negative samples in semantic segmentation, in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)* (Grenoble, France, March 2020), pp. 1351–1356

[RRSG21] T. Riedlinger, M. Rottmann, M. Schubert, H. Gottschalk, Gradient-Based Quantification of Epistemic Uncertainty for Deep Object Detectors (July 2021), pp. 1–20. arXiv:2107.04517

[RS19] M. Rottmann, M. Schubert, Uncertainty measures and prediction quality rating for the semantic segmentation of nested multi resolution street scene images, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (Long Beach, CA, USA, June 2019), pp. 1361–1369

[SFKM20] N. Ståhl, G. Falkman, A. Karlsson, G. Mathiason, Evaluation of uncertainty quantification in deep learning, in *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, virtual conference (June 2020), pp. 556–568

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(56), 1929–1958 (2014)

[SKR20] M. Schubert, K. Kahl, M. Rottmann, Metadetect: uncertainty quantification and prediction quality estimates for object detection, in *2021 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2021), pp. 1–10

[SZS+14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Banff, AB, Canada, December 2014), pp. 1–10

[VSG19] V. Thanvantri Vasudevan, A. Sethy, A. Roshan Ghias, Towards better confidence estimation for neural models, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Brighton, UK, May 2019), pp. 7335–7339

[ZE01] B. Zadrozny, C. Elkan, Obtaining calibrated probability estimates from decision trees and Naive Bayesian classifiers, in *Proceedings of the International Conference on Machine Learning (ICML)* (Williamstown, MA, USA, June 2001), pp. 609–616

[ZE02] B. Zadrozny, C. Elkan, Transforming classifier scores into accurate multiclass probability estimates, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (Edmonton, AB, Canada, July 2002), pp. 694–699

Detecting and Learning the Unknown in Semantic Segmentation

Robin Chan, Svenja Uhlemeyer, Matthias Rottmann, and Hanno Gottschalk

Abstract Semantic segmentation is a crucial component for perception in automated driving. Deep neural networks (DNNs) are commonly used for this task, and they are usually trained on a closed set of object classes appearing in a closed operational domain. However, this is in contrast to the open world assumption in automated driving that DNNs are deployed to. Therefore, DNNs necessarily face data that they have never encountered previously, also known as *anomalies*, which are extremely safety-critical to properly cope with. In this chapter, we first give an overview about anomalies from an information-theoretic perspective. Next, we review research in detecting unknown objects in semantic segmentation. We present a method outperforming recent approaches by training for high entropy responses on anomalous objects, which is in line with our theoretical findings. Finally, we propose a method to assess the occurrence frequency of anomalies in order to select anomaly types to include into a model's set of semantic categories. We demonstrate that those anomalies can then be learned in an unsupervised fashion which is particularly suitable in online applications.

1 Introduction

Recent developments in deep learning have enabled scientists and practitioners to advance in a broad field of applications that were intractable before. To this end, deep neural networks (DNNs) are mostly employed which are usually trained in a

R. Chan (✉) · S. Uhlemeyer · M. Rottmann · H. Gottschalk
University of Wuppertal, Gaußstr. 20, 42119 Wuppertal, Germany
e-mail: rchan@uni-wuppertal.de

S. Uhlemeyer
e-mail: suhlemeyer@uni-wuppertal.de

M. Rottmann
e-mail: rottmann@uni-wuppertal.de

H. Gottschalk
e-mail: hanno.gottschalk@uni-wuppertal.de

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_10

supervised fashion with closed-world assumption. However, when those algorithms are deployed to real-world applications, e.g., artificial intelligence (AI) systems used for perception in automated driving, they often operate in an open-world setting where they have to face diversity of the real world. Consequently, DNNs are likely exposed to data which is "unknown" to them and therefore possibly beyond their capabilities to process. For this reason, having methods at hand, that indicate when a DNN is operating outside of its learned domain to seek for human intervention, is of utmost importance in safety-critical applications.

A generic term for such a task is *anomaly* detection, which is generally defined as recognizing when something departs from what is regarded as normal or common. More precisely, identifying anomalous examples during inference, i.e., new examples that are "extreme" in some sense as they lie in low density regimes or even outside of the training data distribution, is commonly referred to as *out-of-distribution (OoD)* or *novelty* detection in deep learning terminology. The latter is in close connection to the task of identifying anomalous examples in training data, which is contrarily known as *outlier* detection; a term originating from classical statistics to determine whether observational data is polluted. Those outlined notions are often used interchangeably in deep learning literature. Throughout this chapter, we will stick to the general term anomaly and only specify when distinguishing is relevant.

For the purpose of anomaly detection, plenty of methods, ranging from classical statistical ones (see Sect. 2) to deep-learning-specific ones (see Sect. 4) have been developed in the past. Nowadays for the most challenging computer vision tasks tackled by deep learning, where both the model weights and output are of high dimension (in the millions), specific approaches to anomaly detection are mandatory.

Classical methods such as density estimation fail due to the curse of dimensionality. Early approaches identify outliers based on the distance to their neighbors [KNT00, RRS00], i.e., they are looking for sparse neighborhoods. Other methods consider relative densities to handle clusters of different densities, e.g., by comparing one instance either to its k-nearest neighbors [BKNS00] or using an ε-neighborhood as reference set [PKGF03]. However, the concept of neighborhoods becomes meaningless in high dimensions [AHK01]. More advanced approaches for high-dimensional data compute outlier degrees based on angles instead of distances [KSZ08] or even identify lower-dimensional subspaces [AY01, KKSZ09].

In deep-learning-driven computer vision applications, novelties are typically regarded as more relevant than outliers. In semantic segmentation, i.e., pixel-level image classification, novelty detection may even refer to a number of sub-tasks. On the one hand, we might be concerned with the detection of semantically anomalous objects. This is also known as anomaly segmentation in the case of semantic segmentation. On the other hand, we also might be concerned with the detection of changed environmental conditions that are novel. The latter may be effects of a domain shift and include change in weather, time of day, seasonality, location and time. In this chapter, we focus only on semantically novel objects as anomalies.

In general, an important capability of AI systems is to identify the unknown. However, when striving for improved self-reflection capabilities, anomaly detection is not sufficient. Another important capability for real-world deployment of AI systems is

to realize that some specific concept appears over and over again and potentially constitutes a new (or novel) object class. Incremental learning refers to the task of learning new classes, however, especially in semantic segmentation, mostly in a strictly supervised or semi-supervised fashion where data for the new class is labeled with ground truth [MZ19, CMB+20]. This is accompanied by an enormous data collection and annotation effort. In contrast to supervised incremental learning, humans may recognize a novelty of a given class that appears over and over again very well, such that in the end a single feedback might be sufficient to assign a name to a novel class. For the task of image classification, [HZ21] provides an unsupervised extension of the semantic space, while for segmentation there exist only approaches for supervised extension of the semantic space via incremental learning.

In this chapter, we first introduce anomaly detection from an information-based perspective in Sect. 2. We provide theoretical evidence that the entropy is a suitable quantity for anomaly detection, particularly in semantic segmentation. In Sect. 3, we review recent developments in the fields of anomaly detection and unsupervised learning of new classes. We give an overview on existing methods, both in the context of image classification and semantic segmentation. In this setting, we present an approach to train semantic segmentation DNNs for high entropy on anomaly data in Sect. 4. We compare our proposed approach against other established and recent state-of-the-art anomaly segmentation methods and empirically show the effectiveness of entropy maximization in identifying unknown objects. Lastly, we propose an unsupervised learning technique for novel object classes in Sect. 5. Further, we provide an outlook how the latter approach can be combined with entropy maximization to handle the unknown at run time in automated driving.

2 Anomaly Detection Using Information and Entropy

Anomaly detection is a common routine in any data analysis task. Before training a statistical model on data, the data should be investigated whether the underlying distribution generating the data is polluted by anomalies. In this context, anomalies can generally be understood as samples that do not fit into a distribution. Such anomalous samples can, e.g., be generated in the data recording process either by extreme observations, by errors in recording and transmission, or by the fusion of datasets that use different systems of units. Most common for the detection of anomalies in statistics is the inspection of maximum and minimum values for each feature, or simple univariate visualization via box-whisker plots or histograms.

More sophisticated techniques are applied in multivariate anomaly detection. Here, anomalous samples do not necessarily have to contain extreme values for single features, but rather an untypical combination of them. One of the application areas for multivariate anomaly detection is, e.g., fraud detection.

In both outlined cases, an anomaly $\mathbf{z} \in \mathbb{R}^d$ can be qualified as an observation that occurs at a location of extremely low density of the underlying distribution $\mathrm{p}(\mathbf{z})$ or, equivalently, has an exceptionally high value of the information

$$I(\mathbf{z}) = -\log \mathrm{p}(\mathbf{z}) \,. \tag{1}$$

Here, two problems occur: First, it is generally not specified what is considered as exceptionally high. Second, $\mathrm{p}(\mathbf{z})$ and thereby $I(\mathbf{z})$ are generally unknown. Regarding the latter issue, however, the estimate $\hat{I}(\mathbf{z}) = -\log \hat{\mathrm{p}}(\mathbf{z})$ can be used which in turn relies on estimating $\hat{\mathrm{p}}(\mathbf{z})$ from data associated to the probability density function $\mathrm{p}(\mathbf{z})$. Estimation approaches for $\hat{\mathrm{p}}(\mathbf{z})$ can be distinguished between parametric and non-parametric ones.

The Mahalanobis distance [Mah36] is the best known parametric method for anomaly detection which is based on information of the multivariate normal distribution N. In fact, if $\mathbf{z} \sim N(\boldsymbol{\mu}, \Sigma)$ with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and positive definite covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$, then

$$I(\mathbf{z}) = -\log\left(\frac{1}{(2\pi)^{d/2}(\det\Sigma)^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^{\mathsf{T}}\Sigma^{-1}(\mathbf{z}-\boldsymbol{\mu})\right)\right) \tag{2}$$

$$= \frac{d}{2}\log(2\pi) + \frac{1}{2}\log(\det\Sigma) + \frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^{\mathsf{T}}\Sigma^{-1}(\mathbf{z}-\boldsymbol{\mu}) = \frac{1}{2}\mathsf{d}_\Sigma(\mathbf{z},\boldsymbol{\mu})^2 + c \,, \tag{3}$$

where

$$\mathsf{d}_\Sigma := \sqrt{(\mathbf{z}-\boldsymbol{\mu})^{\mathsf{T}}\Sigma^{-1}(\mathbf{z}-\boldsymbol{\mu})} \tag{4}$$

denotes the Mahalanobis distance. The estimation $\hat{I}(\mathbf{z})$ is obtained by replacing $\boldsymbol{\mu}$ and Σ by the arithmetic mean $\hat{\boldsymbol{\mu}}$ and the empirical covariance matrix $\hat{\Sigma}$, respectively, and likewise $\mathsf{d}_\Sigma(\mathbf{z},\boldsymbol{\mu})$ by the empirical Mahalanobis distance $\mathsf{d}_{\hat{\Sigma}}(\mathbf{z},\hat{\boldsymbol{\mu}})$.

In contrast, non-parametric techniques of anomaly detection rely on non-parametric techniques to estimate $\mathrm{p}(\mathbf{z})$. Here, a large variety of methods from histograms, kernel estimators and many others exist [Kle09]. We note, however, that the non-parametric estimation of densities and information generally suffers from the curse of dimensionality. To alleviate the latter issue in anomaly detection, estimation of information is often combined with techniques of dimensionality reduction, such as, e.g., principal component analysis [HTF07] or autoencoders [SY14].

When using non-linear dimensionality reduction with autoencoders, densities obtained in the latent space depend on the encoder and not only on the data itself. This points towards a general problem in anomaly detection. If $\mathrm{p}(\mathbf{z})$ is the density of a random quantity $\mathbf{z}$ and $\mathbf{z}' = \boldsymbol{\phi}(\mathbf{z})$ is an equivalent encoding of the data $\mathbf{z}$ using a bijective and differentiable mapping $\boldsymbol{\phi} : \mathbb{R}^d \mapsto \mathbb{R}^d$, the *change of variables* formula [Rud87, AGLR21]

$$\mathrm{p}(\mathbf{z}') = \mathrm{p}(\mathbf{z}) \cdot |\det(\nabla_{\mathbf{z}'}\mathbf{z})| = \mathrm{p}(\mathbf{z}) \cdot |\det\left(\nabla_{\mathbf{z}}\boldsymbol{\phi}^{-1}(\mathbf{z})\right)| \tag{5}$$

implies that the information of $\mathbf{z}'$ is

$$I(\mathbf{z}') = -\log(\mathrm{p}(\mathbf{z})) - \log\left(\left|\det\left(\nabla_{\mathbf{z}}\boldsymbol{\phi}^{-1}(\mathbf{z})\right)\right|\right) \,, \tag{6}$$

where $\nabla_{\mathbf{z}}\boldsymbol{\phi}^{-1}(\mathbf{z})$ denotes the Jacobian matrix of the inverse function $\boldsymbol{\phi}^{-1}$. Thus, whenever a high value of $I(\mathbf{z})$ indicates an anomaly, there always exists another equivalent representation of the data $\mathbf{z}'$, where the information $I(\mathbf{z}')$ is low. In other words, if $\mathbf{z}$ is remote from other instances $\mathbf{z}_j$ of a dataset and therefore considered an anomaly, there will be a transformation $\mathbf{z}' = \boldsymbol{\phi}(\mathbf{z})$ that brings $\mathbf{z}'$ right into the center of the data $\mathbf{z}'_j = \boldsymbol{\phi}(\mathbf{z})_j$. In fact, via the Rosenblatt transformation [Ros52] any representation $\mathbf{z}$ of the data can be expressed via a representation $\mathbf{z}' = \boldsymbol{\phi}(\mathbf{z})$ where $I(\mathbf{z}')$ is constant over all data points. This stresses the importance to understand that an anomaly always refers to probability *and* encoding of the data $\mathbf{z}$. This is true for both the original data and its approximated lower-dimensional representation.

As a side remark, autoencoders designed from neural networks have been very successfully applied in anomaly detection. Encoder and decoder networks possess the universal approximation property [Cyb89]. Furthermore, common training losses like the reconstruction error are invariant under a change of the representation on latent spaces. Therefore, additional insights seem to be required to explain the empirical success of anomaly detection with autoencoders which is, however, not the scope of this chapter.

Another way of looking at the issue of anomaly detection in the context of different representations of same data is an explicit choice of a reference measure. This reference measure represents to which extent, or how likely, data is contaminated by potential anomalies. Suppose we can associate the probability density $\mathrm{p}^{\mathrm{anom}}(\mathbf{z})$ to the reference measure, then we can base anomaly detection on the quotient of densities, i.e., the odds $\frac{\mathrm{p}(\mathbf{z})}{\mathrm{p}^{\mathrm{anom}}(\mathbf{z})}$, and apply a threshold whenever this ratio is low or, equivalently, when the relative information

$$I^{\mathrm{rel}}(\mathbf{z}) := -\log\left(\frac{\mathrm{p}(\mathbf{z})}{\mathrm{p}^{\mathrm{anom}}(\mathbf{z})}\right) = I(\mathbf{z}) - I^{\mathrm{anom}}(\mathbf{z}) \tag{7}$$

is high. We note that the relative information is independent under changes of the representation $\mathbf{z}' = \boldsymbol{\phi}(\mathbf{z})$ as the $-\log|\det(\nabla_{\mathbf{z}}\boldsymbol{\phi}^{-1}(\mathbf{z}))|$ term from (6) occurs once with positive sign in $I(\mathbf{z}')$ and once with negative sign in $-I^{\mathrm{anom}}(\mathbf{z}')$ and therefore cancels. Thus, the choice of a reference measure and the choice of a representation for the data is largely equivalent.

In practical situations, $\mathrm{p}^{\mathrm{anom}}(\mathbf{z})$ is often represented by some data $\{\mathbf{z}_i^{\mathrm{anom}}\}_{i\in\mathcal{T}'}$ that are either simulated or drawn from some data source of known anomalies. A binary classifier $\hat{\mathrm{p}}(\mathrm{anom}|\mathbf{z})$ can then be trained on basis of proper data $\{\mathbf{z}_i\}_{i\in\mathcal{T}}$ and anomalous data $\{\mathbf{z}_i^{\mathrm{anom}}\}_{i\in\mathcal{T}'}$. The assumed prior probability $\mathrm{p}(\mathrm{anom})$ for anomalies, i.e., the degree of contamination, acts as a threshold for the estimated odds. Equivalently, the estimate of the relative information

$$\hat{I}^{\mathrm{rel}}(\mathbf{z}) = -\log\left(\frac{\hat{\mathrm{p}}(\mathbf{z})}{\hat{\mathrm{p}}^{\mathrm{anom}}(\mathbf{z})}\right) \overset{\text{Bayes' Theorem}}{=} -\log\left(\frac{\hat{\mathrm{p}}(\text{non-anom}|\mathbf{z})\mathrm{p}(\mathbf{z})}{\mathrm{p}(\text{non-anom})} \cdot \frac{\mathrm{p}(\mathrm{anom})}{\hat{\mathrm{p}}(\mathrm{anom}|\mathbf{z})\mathrm{p}(\mathbf{z})}\right) \tag{8}$$

$$= -\log\left(\frac{1-\hat{\mathrm{p}}(\mathrm{anom}|\mathbf{z})}{\hat{\mathrm{p}}(\mathrm{anom}|\mathbf{z})} \cdot \frac{\mathrm{p}(\mathrm{anom})}{1-\mathrm{p}(\mathrm{anom})}\right) \tag{9}$$

$$= -\log\left(\frac{1-\hat{\mathrm{p}}(\mathrm{anom}|\mathbf{z})}{\hat{\mathrm{p}}(\mathrm{anom}|\mathbf{z})}\right) - \log\left(\frac{\mathrm{p}(\mathrm{anom})}{1-\mathrm{p}(\mathrm{anom})}\right) \tag{10}$$

$$= -\log\left(\frac{1-\hat{\mathrm{p}}(\mathrm{anom}|\mathbf{z})}{\hat{\mathrm{p}}(\mathrm{anom}|\mathbf{z})}\right) + c \tag{11}$$

with the prior log-odds $c = -\log\left(\frac{\mathrm{p}(\mathrm{anom})}{1-\mathrm{p}(\mathrm{anom})}\right)$ being a parameter controlling the threshold for the binary classifier $\hat{\mathrm{p}}(\mathrm{anom}|\mathbf{z})$.

If specifying what is an exceptionally high value for the information $I(\mathbf{z})$ or relative information $I^{\mathrm{rel}}(\mathbf{z})$, the distinction between the detection of outliers in the training data and the detection of novelties during inference has to be taken into account. In outlier detection, observations, which have high (relative) information but which are in agreement with the extreme value of the (relative) information

$$I^{\max} = \max_{i\in\mathcal{T}} I(\mathbf{z}_i) \quad \text{or} \quad I^{\mathrm{rel}\,\max} = \max_{i\in\mathcal{T}} I^{\mathrm{rel}}(\mathbf{z}_i), \tag{12}$$

are usually intentionally not eliminated. An outlier $\mathbf{z}$ for the level of significance $0 < \alpha < 1$ can then be detected using the condition

$$\mathrm{P}_{\{\mathbf{z}_i\}_{i\in\mathcal{T}}}(I^{\max} > I(\mathbf{z})) \leq \alpha \quad \text{or} \quad \mathrm{P}_{\{\mathbf{z}_i\}_{i\in\mathcal{T}}}(I^{\mathrm{rel}\,\max} > I^{\mathrm{rel}}(\mathbf{z})) \leq \alpha. \tag{13}$$

Note again that the distribution of $I^{\mathrm{rel}}(\mathbf{z}_j)$ has to be estimated to derive the associated distribution for the extreme values, see, e.g., [DHF07], and also $I^{\mathrm{rel}}(\mathbf{z})$ requires the estimation $\hat{\mathrm{p}}(\mathbf{z})$ or $\hat{\mathrm{p}}^{\mathrm{anom}}(\mathbf{z})$. Therefore, a quantification of the epistemic uncertainty is essential for a proper outlier detection. Given the already mentioned problems of density estimation in high dimension, epistemic uncertainties may play a major role, unless a massive amount of data is available.

For the case of novelty detection taking place at inference, a comparison of the information of a single instance $I^{\mathrm{rel}}(\mathbf{z})$ with the usual distribution of information $\mathrm{P}_{\mathbf{z}_i}$ seems to be in order, which leads to the novelty criterion for level of significance $0 < \alpha < 1$

$$\mathrm{P}_{\mathbf{z}_i}(I(\mathbf{z}_i) > I(\mathbf{z})) \leq \alpha \quad \text{or} \quad \mathrm{P}_{\mathbf{z}_i}(I^{\mathrm{rel}}(\mathbf{z}_i) > I^{\mathrm{rel}}(\mathbf{z})) \leq \alpha. \tag{14}$$

As a variant to this criterion, $I^{\mathrm{rel}}(\mathbf{z}_i)$ could also be replaced by the extreme value statistics over the number of inferences alleviating the problem of generating false novelties by multiple testing. What has been stated on the necessity to quantify the epistemic uncertainty for the case of outlier detection equally applies for novelty detection.

While anomaly detection is generally seen as a sub-field of unsupervised learning, some specific effects occur in the case of novelty detection in supervised learning. During the phase of inference, the data $\mathbf{z} = (y, \mathbf{x})$ contain an unobserved component $y \in \mathcal{S}$, which, e.g., represent the instance's label in a classification problem for the classes contained in $\mathcal{S}$. Using the decomposition $\mathrm{p}(\mathbf{z}) = \mathrm{p}(y, \mathbf{x}) = \mathrm{p}(y|\mathbf{x})\mathrm{p}(\mathbf{x})$, one obtains the (relative) information from

$$I(\mathbf{z}) = I(y|\mathbf{x}) + I(\mathbf{x}), \quad \text{or} \quad I^{\text{rel}}(\mathbf{z}) = I(y|\mathbf{x}) + I^{\text{rel}}(\mathbf{x}) - I^{\text{anom}}(y|\mathbf{x}), \tag{15}$$

where $I(y|\mathbf{x}) = -\log(\mathrm{p}(y|\mathbf{x}))$, $I^{\text{anom}}(y|\mathbf{x}) = -\log(\mathrm{p}^{\text{anom}}(y|\mathbf{x}))$ is the conditional information on the right hand side. Often, for the data of the reference measure $\mathrm{p}^{\text{anom}}(\mathbf{z})$, the labels are not contained in $\mathcal{S}$. In this case, one uses a non-informative conditional distribution $\mathrm{p}^{\text{anom}}(y|\mathbf{x}) = \frac{1}{|\mathcal{S}|}$. If this is done, the last term of (15) becomes a constant that can be integrated into a threshold parameter.

The (relative) information cannot be computed without knowing y. Therefore, the conditional expectation is used as an unbiased estimate, yielding the expected information

$$EI(\mathbf{x}) = \mathbb{E}_{y\sim\mathrm{p}(y|\mathbf{x})}(I^{\text{rel}}(\mathbf{z})) = E(\mathbf{x}) + I^{\text{rel}}(\mathbf{x}) + b^{\text{rel}}, \tag{16}$$

where $E(\mathbf{x}) = \sum_{y\in\mathcal{S}} \mathrm{p}(y|\mathbf{x}) I(y|\mathbf{x})$ is the expected information, or entropy, of the conditional distribution $\mathrm{p}(y|\mathbf{x})$ and b^{rel} is zero for the information and equal to $-\log(|\mathcal{S}|)$ for the relative information with non-informative conditional distribution $\mathrm{p}^{\text{anom}}(y|\mathbf{x})$. Note that $E(\mathbf{x})$ is bounded by $\log(|\mathcal{S}|)$. Therefore, under normal circumstances, the term $I^{\text{rel}}(\mathbf{x})$ will outweigh $E(\mathbf{x})$ by far. However, in problems like semantic segmentation, each component of $\mathbf{x}$ is assigned a label from $\mathcal{S}$. This implies solving $|\mathcal{I}|$ classification problems, where $\mathcal{I}$ denotes the pixel space of $\mathbf{x}$, thus the maximum value for $E(\mathbf{x})$ yields $|\mathcal{I}|\log(|\mathcal{S}|)$.

Therefore, the first term in (16) contains significant contributions, especially in situations where $|\mathcal{I}|$ is large. The second term, $I^{\text{rel}}(\mathbf{x})$ loses importance under the hypothesis that the probability of the inputs $\mathbf{x}$ does not vary greatly. Despite this hypothesis could be supported by fair sampling strategies, it requires further critical evaluation. But at least to a significant part, the expected information as an anomaly measure with regard to instance $\mathbf{x}$ is given by a dispersion measure, namely the entropy of the conditional probability. As the entropy can be well estimated using a supervised machine learning approach to estimate $\hat{\mathrm{p}}(y|\mathbf{x})$ from the data $\{\mathbf{z}_j\}_{j\in\mathcal{T}}$, this part of the information is well accessible in contrast to $I^{\text{rel}}(\mathbf{x})$, which requires density estimation in high dimension.

Lastly in this section, let us give a remark on the role of anomaly data $\{\mathbf{z}_j^{\text{anom}}\}_{j\in\mathcal{T}'} = \{\mathbf{x}_j^{\text{anom}}\}_{j\in\mathcal{T}'}$. If such data is available, it is desirable to train the machine learning model $\hat{\mathrm{p}}(y|\mathbf{x})$ to produce high values for $E(\mathbf{x}_j^{\text{anom}})$ so that the tractable part of the expected information $EI(\mathbf{x})$ shows good separation properties. This requirement can be inserted to the loss function, as it has been proposed in [HAB19, HMKS19] for classification. In fact, as the entropy $E(\mathbf{x})$ is maximized by the uniform (non-informative) label distribution $\mathrm{p}(y|\mathbf{x}_j^{\text{anom}}) = \frac{1}{|\mathcal{S}|}$, the aforementioned loss will favor this prediction on anomalous inputs $\{\mathbf{x}_j^{\text{anom}}\}_{j\in\mathcal{T}'}$. In this chapter, in Sect. 4, we will extend this approach to the computer vision task of semantic segmentation, after having reviewed related works based on deep learning in the following section.

3 Related Works

After the introduction to anomaly detection from a theoretical point of view, we now turn to anomaly detection in deep learning. In this section, we review research in the direction of detecting and learning unknown objects in semantic segmentation.

3.1 Anomaly Detection in Semantic Segmentation

An emerging body of work explores the detection of anomalous inputs on image data, where the task is more commonly referred to as *anomaly* or *out-of-distribution (OoD) detection*. Anomaly detection was first tackled in the context of image classification by introducing post-processing techniques applied to softmax probabilities to adjust the confidence values produced by a classification model [HG17, LLLS18, LLS18, HAB19, MH20]. These methods have proven to successfully lower confidence scores for anomalous inputs at image-level, which is why they were also adapted to anomaly detection in semantic segmentation [ACS19, BSN+19], i.e., to *anomaly segmentation* by treating each single pixel in an image as a potential anomaly. Although those methods represent good baselines, they usually do not generalize well to segmentation, e.g., due to the high prediction uncertainties at object boundaries. The latter problem can, however, be mitigated by using segment-wise prediction quality estimates [RCH+20], an approach which has also demonstrated to indicate anomalous regions within an image [ORF20].

Recent works have proposed more dedicated solutions to anomaly segmentation. Among the resulting methods, many originate from uncertainty quantification. The intuition is that anomalous regions in an image correlate with high uncertainty. In this regard, early approaches estimate uncertainty using Bayesian deep learning, treating model parameters as distributions instead of point estimates [Mac92, Nea96]. Due to the computational complexity, approximations are mostly preferred in practice, which comprise, e.g., Monte-Carlo dropout [GG16], stochastic batch normalization [AAM+19], or an ensemble of neural networks [LPB17, GDS20]; with some of them also being extended to semantic segmentation in [BKC17, KG17, MG19]. Even when using approximations, Bayesian models still tend to be computationally expensive. Thus, they are not well suited to real time semantic segmentation which is required for safe automated driving.

This is why tackling anomaly segmentation with non-Bayesian methods are more favorable from a practitioner's point of view. Some approaches therefore include tuning a previously trained model to the task of anomaly detection, by either modifying its architecture or exploiting additional data. In [DT18], anomaly scores are learned by adding a separate branch to the neural network. In [HMD19, MH20] the network architecture is not changed but auxiliary outlier data, which is disjoint from the actual training data, is induced into the training process to learn anomalies. The latter idea motivated several works in anomaly segmentation [BSN+19, BKOŠ19,

JRF20, CRG21]. Nonetheless, such models have to cope with multiple tasks, hence possibly leading to a performance loss with respect to the original semantic segmentation task [VGV+21]. Moreover, when including outlier datasets in the training process, it cannot be guaranteed that the chosen outlier data is a good proxy for all possible anomalies.

Another recent line of works performs anomaly segmentation via generative models that reconstruct original input images. These methods assume that reconstructed images will better preserve the appearance of known image regions than that of unknown ones. Anomalous regions are then identified by means of pixel-wise discrepancies between the original and reconstructed image. Thus, such an approach is specifically designed to anomaly segmentation and has been extensively studied in [CM15, MVD17, LNFS19, XZL+20, LHFS21, BBSC21]. The main benefit of these approaches is that they do not require any OoD training data, allowing them to generalize to all possible anomalous objects. However, all these methods are limited by the integrated discrepancy module, i.e., the module that identifies relevant differences between the original and reconstructed image. In complex scenes, such as street scene images for automated driving, this might be a challenging task due to the open world setting.

Regarding the dataset landscape, only few anomaly segmentation datasets exist. The LostAndFound dataset [PRG+16] is a prominent example which contains anomalous objects in various streets in Germany while sharing the same setup as Cityscapes [COR+16]. LostAndFound, however, considers children and bicycles as anomalies, even though they are part of the Cityscapes training set. This was filtered and refined in Fishyscapes [BSN+19]. Another anomaly segmentation dataset accompanies the CAOS benchmark [HBM+20], which considers three object classes from BDD100k [YCW+20] as anomalies. Both, Fishyscapes and CAOS, try to mitigate low diversity by complementing their real images with synthetic data.

Efforts to provide anomalies in real images have been made in [LNFS19] by sourcing and annotating street scene images from the web and in [LHFS21, SKGK20] by capturing and annotating images with small objects placed on the road. Just recently, the datasets published alongside the SegmentMeIfYouCan benchmark [CLU+21] build upon those works, particularly contributing to broad diversity of anomalous street scenes as well as objects.

3.2 Incremental Learning in Semantic Segmentation

Building upon the detection of anomalies, training data can be enriched in order to learn novel classes. To avoid training from scratch, several approaches tackle the task of *incremental* or even *continuous learning*, which can be understood as adapting to continuously evolving environments. Besides learning novel classes, incremental learning also encompasses adapting to alternative tasks or other domains. A comprehensive framework to compare these different learning scenarios is provided in [vdVT19].

When learning novel classes, the primary issue incremental learning approaches face is a loss of the original performance on previously learned classes, that is commonly known as *catastrophic forgetting* [MC89]. To overcome this problem, a model needs to be both, "stable" and "plastic", i.e., the model needs to retain its original knowledge while being able to adapt to new environments. The complexity of meeting these requirements at the same time is called the stability-plasticity-dilemma [AR05]. In this regard, proposed solution strategies can be separated into three categories, which are either based on architecture, regularization, or rehearsal. Most of these methods have been applied to image classification first.

Architecture strategies employ separate models for each sequential incremental learning task, combined with a selector to determine which model will be used for inference [PUUH01, CSMDB19, ARC20]. However, these approaches suffer from data imbalances, consequently standard classification algorithms tend to favor the majority class. Approaches to mitigate skewed data distributions are usually based on over- or undersampling. Another line of works, such as [RRD+16, RPR20], employ "growing" models, i.e., enlarging the model capacity by increasing the number of model parameters for more complex tasks. In [ACT17], the authors propose an automated approach to select the proper task-specific model at test time. More efficient approaches were introduced in [GK16, YYLH18], that restrict the adaptation of parameters to relevant parts of the model in terms of the new task. The `Self-Net` [MCE20] is made up of an autoencoder that learns low-dimensional representations of the models belonging to previously learned tasks. By that, retaining existing knowledge via approximating the old weights instead of saving them directly is accompanied with an implicit storage compression. The incremental adaptive deep model developed in [YZZ+19] enables capacity scalability and sustainability by exploiting the fast convergence of shallow models at the initial stage and afterwards utilizing the power of deep representations gradually. Other procedures perform continuous learning, e.g., using a random-forest [HCP+19], an incrementally growing DNN, retaining a basic backbone [SAR20], or nerve pruning and synapse consolidation [PTJ+21].

Regularization strategies can be further distinguished between weight regularization, i.e., measuring the importance of weights, and distillation, i.e., transferring a model's knowledge into another. The former identifies parameters with great impact on the original tasks that are suppressed to be updated. Elastic weight consolidation (EWC) [KPR+17] is one representative method, evaluating weight importance based on the Fisher information matrix, while the synaptic intelligence (SI) method [ZPG17] calculates the cumulative change of Euclidean distance after retraining the model. Both regularization methods were further enhanced, e.g., by combining them [CDAT18, AM19], or by including unlabeled data [ABE+18]. Another idea to maintain model stability was adapted in [ZCCY21, FAML19], updating gradients based on orthogonal constraints. Bayesian neural networks are applied in [LKJ+17] to approximate a Gaussian distribution of the parameters from a single to a combined task.

Distillation is a regularization method, where the knowledge of an old model can be drawn into a new model to partly overcome catastrophic forgetting. Knowl-

edge distillation, proposed in [HVD14], was originally invented to transfer knowledge from a complex into a simple model. The earliest approach, which applies knowledge distillation to incremental learning, is called learning without forgetting (LwF) [LH18]. A combination of knowledge distillation and EWC was proposed in [SCL+18]. Further approaches based on distillation loss are e.g., [JJJK18, YHW+19, KBJC19, LLSL19].

Rehearsal or pseudo-rehearsal-based methods, which were already proposed in [Rob95], mitigate catastrophic forgetting by allowing the model to review old knowledge whenever it learns new tasks. While rehearsal-based methods retain a subset of the old training data, pseudo-rehearsal strategies construct a generator during retraining, which learns to produce pseudo-data as similar to the old training data as possible. Hence, they provide the advantages of rehearsal even if the previously learned information is unavailable. Methods reusing old data are, e.g., incremental classifier and representation learning (iCaRL) [RKSL17], which simultaneously learns classifiers and feature representation, or the method presented in [CMJM+18], which proposes a representative memory. The bias correction (BiC) method [WCW+19] keeps old data in a similar manner, but handles the data imbalance differently. Most pseudo-rehearsal approaches include generative adversarial networks (GANs) [OOS17, WCW+18, MSC+19, OPK+19] or a variational autoencoder (VAE) [SLKK17]. The method presented in [HPL+18] combines distillation and retrospective (DR), whereby baseline approaches such as LwF are outperformed by a large margin.

Only few works exist, such as [KBDFs20, MZ21, TTA19], that adapt incremental learning techniques to semantic segmentation. They adjust knowledge distillation, using no or only a small portion of old data, respectively. One challenge of continuous learning for semantic segmentation is that images may contain unseen as well as known classes. Hence, annotations that are restricted to some task assign a great amount of pixels to a background class, exhibiting a semantic distribution shift. The authors of [CMB+20] provide a framework that mitigates biased predictions towards this background class. While existing approaches require supervision, we employ incremental learning in a semi-supervised fashion, as we do not have access to any ground truth including novelties.

4 Anomaly Segmentation

The task of anomaly detection in the context of semantic segmentation, i.e., identifying anomalies at pixel-level, is commonly known as *anomaly segmentation*. For this task several approaches have been proposed that are either based on uncertainty quantification, generative models, or training strategies specifically tailored to anomaly detection. In this chapter, we will first review some of those well-established methods and, subsequently, report a performance comparison with respect to their capability of identifying anomalies. In particular, we will demonstrate empirically that entropy maximization yields great performance on this segmentation task, which

is in accordance to the statement of the entropy's importance from the information-based perspective as presented in Sect. 2.

4.1 Methods

Let $\mathbf{x} \in \mathbb{I}^{H\times W\times 3}$, $\mathbb{I} = [0, 1]$, denote (normalized) color images of resolution $H \times W$. Feeding those images to a semantic segmentation network $\mathbf{F} : \mathbb{I}^{H\times W\times 3} \to \mathbb{R}^{H\times W\times S}$, the model produces pixel-wise class scores $\mathbf{y} = (y_{i,s})_{i\in\mathcal{I},s\in\mathcal{S}} = \mathbf{F}(\mathbf{x}) \in \mathbb{R}^{H\times W\times S}$, with the set of pixel locations denoted by $\mathcal{I} = \{1, \ldots, H\} \times \{1, \ldots, W\}$ and the set of trained (hence known) classes denoted by $\mathcal{S} = \{1, \ldots, S\}$. The corresponding predicted segmentation mask is given by $\mathbf{m} = (m_i)_{i\in\mathcal{I}} \in \{1, \ldots, S\}^{H\times W}$, where for $m_i = \arg\max_{s\in\mathcal{S}} y_{i,s}$ $\forall\, i \in \mathcal{I}$ the maximum a-posteriori probability principle is applied. Regarding the task of anomaly segmentation, the ultimate goal is then to obtain a score map $\mathbf{a} = (a_i)_{i\in\mathcal{I}} \in \mathbb{R}^{H\times W}$ that indicates the presence of an anomaly at each pixel location $i \in \mathcal{I}$ within image $\mathbf{x}$, i.e., the higher the score the more likely there should be an anomaly.

Each of the methods employed in this section provides such score maps. Their underlying segmentation networks (`DeepLabV3+`, [CZP+18]) are all trained on Cityscapes [COR+16], i.e., objects not included in the set of Cityscapes object classes are considered as anomalies since they have not been seen during training and thus are unknown. The anomaly detection methods, however, differ in the way how the scores are obtained, which is why we briefly introduce the different techniques in the following.

Maximum softmax probability: The most commonly-used baseline for anomaly detection at image level is thresholding at the maximum softmax probability (MSP) [HG17]. Therefore, this method assumes that anomalies are attached a low confidence or, equivalently, high uncertainty. Using MSP in anomaly segmentation, the score map is computed via

$$a_i = 1 - \max_{s\in\mathcal{S}} \mathbf{softmax}(\mathbf{y}_i) \quad \forall\, i \in \mathcal{I}\,. \tag{17}$$

ODIN: A simple extension to improve MSP is applying temperature scaling as well as adding perturbations, which is known as out-of-distribution detector for Neural networks (ODIN) [LLS18]. In more detail, let $t \in \mathbb{R}_{>0}$ be a hyperparameter for temperature scaling and let $\varepsilon \in \mathbb{R}_{\geq 0}$ be a hyperparameter for the perturbation magnitude. Then, the input $\mathbf{x}$ is modified as

$$\tilde{\mathbf{x}} = (\tilde{x}_i)_{i\in\mathcal{I}} \quad \text{with} \quad \tilde{x}_i = x_i - \varepsilon\ \mathrm{sign}\left(-\frac{\partial}{\partial x_i} \log \max_{s\in\mathcal{S}} \mathbf{softmax}\left(\frac{\mathbf{y}_i}{t}\right)\right) \quad \forall\, i \in \mathcal{I}\,, \tag{18}$$

yielding prediction $\tilde{\mathbf{y}} = F(\tilde{\mathbf{x}})$ for which thresholding is applied at the MSP, i.e., the anomaly score map is given by

$$a_i = 1 - \max_{s \in \mathcal{S}} \mathbf{softmax}(\tilde{\mathbf{y}}_i) \quad \forall\, i \in \mathcal{I}\,. \tag{19}$$

Mahalanobis distance: This anomaly detection approach estimates how well latent features fit to those observed in the training data. Let $(L-1)$ denote the penultimate layer of a network $\mathbf{F}$ with L layers. In [LLLS18] the authors have shown that training a softmax classifier fits a class-conditional Gaussian distribution for the output features $\mathbf{f}_{L-1}$. Hence, under that assumption

$$\mathrm{P}\left(\mathbf{y}_i^{(L-1)} \,\middle|\, \overline{y}_{i,s} = 1\right) = N\left(\mathbf{y}_i^{(L-1)} \,\middle|\, \boldsymbol{\mu}_s, \Sigma_s\right) \quad \forall\, i \in \mathcal{I}\,, \tag{20}$$

where $\mathbf{y}^{(L-1)} = \mathbf{f}_{L-1}(\mathbf{x}) \in \mathbb{R}^{H \times W \times C_{L-1}}$ denotes the feature map of the penultimate layer given input $\mathbf{x}$, and $\overline{\mathbf{y}}$ the corresponding one-hot encoded final target. The minimal Mahalanobis distance $d_{\Sigma_s}(\mathbf{x}, \boldsymbol{\mu}_s)$ is then an obvious choice for an anomaly score map

$$a_i = \min_{s \in \mathcal{S}} d_{\Sigma_s}(\mathbf{x}, \boldsymbol{\mu}_s) = \min_{s \in \mathcal{S}} (\mathbf{y}_i^{(L-1)} - \boldsymbol{\mu}_s)^{\mathsf{T}} \Sigma_s^{-1} (\mathbf{y}_i^{(L-1)} - \boldsymbol{\mu}_s) \quad \forall\, i \in \mathcal{I}\,, \tag{21}$$

cf. (2). Note that the class means $\boldsymbol{\mu}_s \in \mathbb{R}^{C_{L-1}}$ and class covariances $\Sigma_s \in \mathbb{R}^{C_{L-1} \times C_{L-1}}$ are generally unknown, but can be estimated by means of the training dataset.

Monte-Carlo dropout: In semantic segmentation, Monte Carlo dropout represents the most prominent technique to approximate Bayesian neural networks. According to [MG19], (epistemic) uncertainty is measured as the mutual information which might serve as anomaly score map, i.e.,

$$a_i = -\sum_{s \in \mathcal{S}} \left(\frac{1}{R} \sum_{r \in \mathcal{R}} \mathrm{p}_{i,s}^{(r)}\right) \log \left(\frac{1}{R} \sum_{r \in \mathcal{R}} \mathrm{p}_{i,s}^{(r)}\right) - \frac{1}{R} \sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} \mathrm{p}_{i,s}^{(r)} \log \mathrm{p}_{i,s}^{(r)} \quad \forall\, i \in \mathcal{I}, \tag{22}$$

with $\mathbf{p}_i^{(r)} = (p_{i,s}^{(r)})_{s \in \mathcal{S}} = \mathbf{softmax}(\mathbf{y}_i^{(r)})$ in the sampling round $r \in \mathcal{R} = \{1, \ldots R\}$. Typically, $8 \leq R \leq 12$.

Void classifier: Neural networks can be trained to output confidences for the presence of anomalies [DT18]. One approach in this context is adding an extra class to the set $\mathcal{S}$ of previously trained classes of a semantic segmentation network, which then also requires annotated anomaly data to learn from. To this end, the void class in Cityscapes is a popular choice as proxy for all possible anomaly data [BSN+19], in particular if the segmentation model was originally trained on Cityscapes. Thus, the softmax output of the additional class $s = S + 1$ represents the anomaly score map, i.e.,

$$a_i = \mathrm{softmax}_{s=S+1}(\mathbf{y}'_i) \quad \forall\, i \in \mathcal{I}\,, \tag{23}$$

where $\mathbf{y}' = (\mathbf{y}'_i)_{i \in \mathcal{I}} = (y'_{i,s})_{i \in \mathcal{I}, s \in \{1,\ldots,S+1\}} = \mathbf{F}'(\mathbf{x})$, $\mathbf{F}' : \mathbb{I}^{H \times W \times 3} \to \mathbb{R}^{H \times W \times (S+1)}$.

Learned embedding density: Let $\mathbf{f}_\ell(\mathbf{x}) \in \mathbb{R}^{H_\ell \times W_\ell \times C_\ell}$ denote the feature map, or equivalently feature embedding, at layer $\ell \in \mathcal{L} = \{1, \ldots, L\}$ of a semantic segmentation network. By employing normalizing flows, the true distribution of features $\mathbf{p}(\mathbf{f}_\ell(\mathbf{x})) \in \mathbb{I}^{H_\ell \times W_\ell}$, where $\mathbf{x} \in \mathcal{X}^{\text{train}}$ is drawn from the training dataset, can be trained via maximum likelihood, i.e., normalizing flows learn to produce the approximation $\hat{\mathbf{p}}(\mathbf{f}_\ell(\mathbf{x})) \approx \mathbf{p}(\mathbf{f}_\ell(\mathbf{x}))$ [BSN+19]. At test time, the negative log-likelihood measures how well features of a test sample fit to the feature distribution observed in the training data, yielding the anomaly score map

$$\mathbf{a} = \mathbf{up}^{\text{lin}}\left(-\mathbf{log}\,\hat{\mathbf{p}}(\mathbf{f}_\ell(\mathbf{x}))\right) \quad (\mathbf{log} \text{ applies log element-wise}) \tag{24}$$

with $\mathbf{up}^{\text{lin}} : \mathbb{R}^{H_\ell \times W_\ell} \to \mathbb{R}^{H \times W}$ denoting (bi-)linear upsampling.

Image resynthesis: After obtaining the predicted segmentation mask $\mathbf{m} \in \{1, \ldots, S\}^{H \times W}$, $\mathbf{m} = (m_i)_{i \in \mathcal{I}}$, this output can be further processed by a generative model $\mathbf{G} : \{1, \ldots, S\}^{H \times W} \to \mathbb{I}^{H \times W \times 3}$ aiming to reconstruct the original input image, i.e., $\mathbf{x}' = \mathbf{G}(\mathbf{m}) \approx \mathbf{x}$. This process is also called image resynthesis, and the intuition is that reconstruction quality for anomalous objects is worse than for those on which the generative model is trained on. To determine pixel-wise anomalies, a discrepancy network [LNFS19] $\mathbf{D} : \{1, \ldots, S\}^{H \times W} \times \mathbb{I}^{H \times W \times 3} \times \mathbb{I}^{H \times W \times 3} \to \mathbb{R}^{H \times W}$ can then be employed, which classifies whether one pixel is anomalous or not, based on information provided by $\mathbf{m}$, $\mathbf{x}'$, and $\mathbf{x}$. Here, $\mathbf{D}$ is trained on intentionally triggered classification mistakes that are produced by flipping classes on predicted segmentation masks. The anomaly score map is given by the output of the discrepancy network, i.e.,

$$\mathbf{a} = \mathbf{D}(\mathbf{m}, \mathbf{x}', \mathbf{x}) = \mathbf{D}(\mathbf{m}, \mathbf{G}(\mathbf{m}), \mathbf{x})\,. \tag{25}$$

SynBoost: The image resynthesis approach is limited by the employed discrepancy module $\mathbf{D}$. In [BBSC21], the authors proposed to extend the discrepancy network by incorporating further inputs based on uncertainty, such as the pixel-wise softmax entropy

$$H_i(\mathbf{x}) = -\sum_{s \in \mathcal{S}} \text{softmax}_s(\mathbf{y}_i) \log(\text{softmax}_s(\mathbf{y}_i)) \quad \forall\, i \in \mathcal{I}, \tag{26}$$

and the pixel-wise softmax probability margin

$$M_i(\mathbf{x}) = 1 - \max_{s \in \mathcal{S}} (\mathbf{softmax}(\mathbf{y}_i)) + \max_{s \in \mathcal{S} \setminus \{m_i\}} (\mathbf{softmax}(\mathbf{y}_i)) \quad \forall\, i \in \mathcal{I}\,. \tag{27}$$

Furthermore, $\mathbf{D}$ is trained on anomaly data provided by the Cityscapes void class. Thus, the anomaly score map is given by

$$\mathbf{a} = \mathbf{D}(\mathbf{m}, \mathbf{x}', \mathbf{x}, \mathbf{H}(\mathbf{x}), \mathbf{M}(\mathbf{x})) = \mathbf{D}(\mathbf{m}, \mathbf{G}(\mathbf{m}), \mathbf{x}, \mathbf{H}(\mathbf{x}), \mathbf{M}(\mathbf{x}))\,. \tag{28}$$

with $\mathbf{H}(\mathbf{x}) = (H_i(\mathbf{x}))_{i \in \mathcal{I}}$ and $\mathbf{M}(\mathbf{x}) = (M_i(\mathbf{x}))_{i \in \mathcal{I}}$.

Entropy maximization: A desirable property of semantic segmentation networks is that they attach high prediction uncertainty to novel objects. To this end, the softmax entropy, see (26), is one intuitive uncertainty measure. The segmentation network can be trained for high entropy on anomalous inputs via the multi-criteria training objective [CRG21]

$$J^{\text{total}} = (1-\lambda)\mathbb{E}_{(\overline{\mathbf{y}},\mathbf{x})\sim\mathcal{D}}[J^{\text{CE}}(\mathbf{F}(\mathbf{x}),\overline{\mathbf{y}})] + \lambda\mathbb{E}_{\mathbf{x}\sim\mathcal{D}^{\text{anom}}}[J^{\text{anom}}(\mathbf{F}(\mathbf{x}))]\,, \tag{29}$$

where $\mathcal{D}$ denotes non-anomaly training data (labels available) and $\mathcal{D}^{\text{anom}}$ denotes anomaly training data (no labels available). In this approach, the COCO dataset [LMB+14] represents a set of so-called *known unknowns*, which is used as proxy for $\mathcal{D}^{\text{anom}}$ with the aim to represent all possible anomaly data. Moreover, $\lambda \in \mathbb{I}$ is a hyperparameter controlling the impact of each single loss function on the overall loss J^{total}. For non-anomaly data, the loss function is chosen to be the commonly-used cross-entropy J^{CE}, while for anomaly data, i.e., for known unknowns, we have

$$J^{\text{anom}}(\mathbf{F}(\mathbf{x})) = -\frac{1}{H\cdot W}\sum_{i\in\mathcal{I}}\frac{1}{S}\sum_{s\in\mathcal{S}}\log \text{softmax}_s(\mathbf{y}_i)\,, \quad \mathbf{x}\sim\mathcal{D}^{\text{anom}}\,. \tag{30}$$

Therefore, minimizing J^{anom} is equivalent to maximizing the softmax entropy since both reach their optimum when the softmax probabilities are uniformly distributed, i.e., $\text{softmax}_s(\mathbf{y}_i) = \frac{1}{S}\ \forall\, s\in\mathcal{S}, i\in\mathcal{I}$. After training, the anomaly score map is then given by the (normalized) softmax entropy

$$a_i = \frac{1}{\log S}H_i(\mathbf{x}) = -\frac{1}{\log S}\sum_{s\in\mathcal{S}}\text{softmax}_s(\mathbf{y}_i)\log(\text{softmax}_s(\mathbf{y}_i)) \quad \forall\, i\in\mathcal{I}\,. \tag{31}$$

From an information-based point of view, the entropy contains significant contribution to the expected information. This particularly applies for instance predictions in semantic segmentation, which motivates the entropy maximization approach for the detection of unknown objects, cf. Sect. 2.

4.2 Evaluation and Comparison of Anomaly Segmentation Methods

Discriminating between anomaly and non-anomaly is essentially a binary classification problem. In order to evaluate the pixel-wise anomaly detection capability, we use the receiver operating characteristic (ROC) curve as well as the precision recall (PR) curve. While for the ROC curve the true positive rate is plotted against the false positive rate at varying thresholds, in the PR curve precision is plotted against recall at varying thresholds. Note that we consider anomalies as the positive class, i.e., correctly identified anomaly pixels are considered as true positive. In both curves,

the degree of separability is then measured by the area under the curve (AUC), where better separability corresponds to a higher AUC.

The main difference between these two performance metrics is how they cope with class imbalance. While the ROC curve incorporates the number of true negatives (for the computation of the false positive rate), in PR curves true negatives are ignored and, consequently, more emphasis is put on finding the positive class. With the anomaly score maps as defined in Sect. 4.1, in our case, finding the positive class corresponds to identifying anomalies.

As evaluation datasets, we use LostAndFoundNoKnown [BSN+19] and RoadObstacle21 [CLU+21], which are both part of the public SegmentMeIfYouCan anomaly segmentation benchmark.[1] LostAndFoundNoKnown consists of 1043 road scene images where obstacles are placed on the road. This dataset is a subset of the prominent LostAndFound dataset [PRG+16] but considers only obstacles from object classes which are disjoint to those in the Cityscapes labels [COR+16]. More precisely, images with humans and bicycles are removed such that the remaining obstacles in the dataset also represent anomalies to models trained on Cityscapes. Similar scenes can be found in RoadObstacle21. That dataset was published alongside the SegmentMeIfYouCan benchmark and contains 327 road obstacle scene images with diverse road surfaces as well as diverse types of anomalous objects. Both datasets restrict the region of interest to the road where anomalies appear. This task is extremely safety-critical as it is mandatory in automated driving to make sure that the drivable area is free of any hazard. All anomaly segmentation methods introduced in the preceding Sect. 4.1 are suited to be evaluated on these datasets. We provide a visual comparison of anomaly scores produced by the tested methods in Fig. 1. We report numerical results in Fig. 2 and in the corresponding Table 1.

In general, we observe that anomaly detection methods originally designed for image classification, including MSP, ODIN and Mahalanobis, do not generalize well to anomaly segmentation. As the Mahalanobis distance is based on statistics of the Cityscapes dataset, the anomaly detection is likely to suffer from performance loss under domain shift. The same holds for Monte Carlo dropout and learned embedding density, particularly resulting in poor performance in RoadObstacle21, where various road surfaces are available. Therefore, those methods potentially act as domain shift classifier rather than as detector of unknown objects.

The detection methods based on autoencoders, namely image resynthesis and SynBoost, show to be better suited for the task of anomaly segmentation, clearly being superior to all the approaches that already have been discussed. Autoencoders are limited by their discrepancy module, and we observe that anomaly detection performance significantly benefits from incorporating uncertainty measures, as done by SynBoost. Only entropy maximization reaches similar anomaly segmentation performance, even outperforming SynBoost in RoadObstacle21. This again can be explained by the diversity of road surfaces, which detrimentally affects the discrepancy module.

[1] www.segmentmeifyoucan.com.

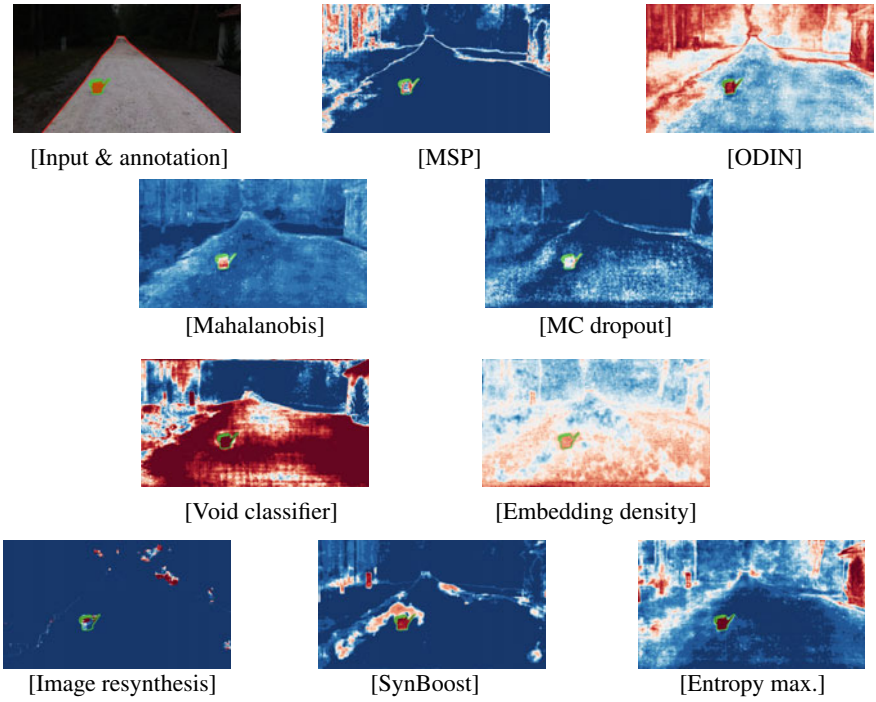


Fig. 1 Qualitative comparison of anomaly score maps for one example image of RoadAnomaly21. Here, red indicates high anomaly scores while blue indicates low ones. The ground truth anomaly instance is highlighted by green contours. Note that the region of interest is restricted to the road, highlighted by red contours in the annotation

As a final remark, we draw attention to the use of anomaly data. The void classifier follows the same intuition as entropy maximization by including known unknowns, but cannot reach nearly as good anomaly segmentation performance. We conclude that the COCO dataset is better suited as proxy for anomalous objects than the Cityscapes unlabeled objects. Moreover, the results of that method empirically demonstrate the impact of the entropy in anomaly segmentation, which is in accordance to the statement of the entropy's importance from the information perceptive described in Sect. 2.

4.3 Combining Entropy Maximization and Meta Classification

Meta classification is the task of discriminating between a true positive prediction and a false positive prediction. For semantic segmentation, this idea was originally

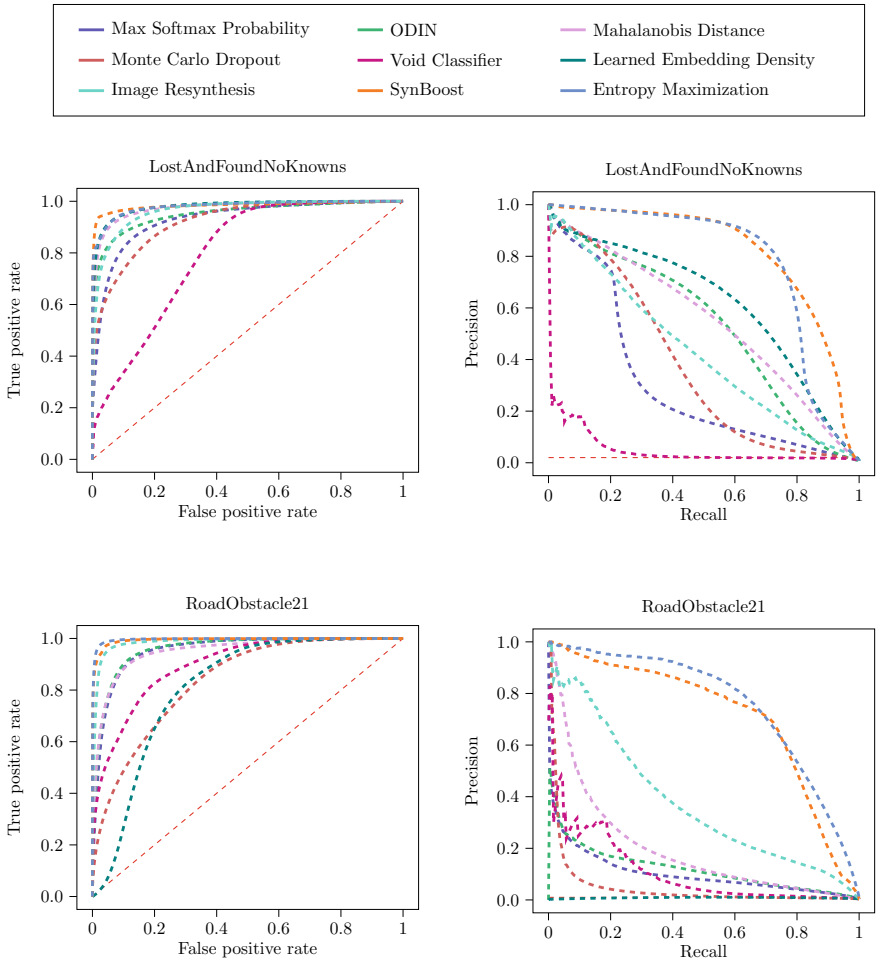


Fig. 2 Receiver operating characteristic (left column) and precision recall (right column) curves for LostAndFoundNoKnowns (top row) and RoadObstacle21 (bottom row), respectively. Dashed red lines indicate the performance of random guessing, i.e., the "no-skill" baseline. The degree of separability between anomaly and non-anomaly is measured by the area under the curve

proposed in [RCH+20]. By means of hand-crafted metrics, which are based on dispersion measures, geometry features, or location information, all derived from softmax probabilities, meta classifiers have shown to reliably identify incorrect predictions at segment level. More precisely, connected components of pixels sharing the same class label are considered as segments in this context, and a false positive segment then corresponds to a segment-wise intersection-over-union (IoU) of 0.

The meta classification approach can straightforwardly be adapted to post-process anomaly segmentation masks. This seems particularly reasonable in combination

Table 1 Pixel-wise anomaly detection performance on the datasets LostAndFoundNoKnown and RoadObstacle21, respectively. The main evaluation metric represents the area under precision-recall curve (AuPRC). Moreover, the area under receiver operating characteristic (AuROC) and the false positive rate at a true positive rate of 95% (FPR_{95TPR}) are reported for further insights

Method	LostAndFoundNoKnown			RoadObstacle21		
	AuPRC ↑	AuROC ↑	FPR_{95TPR} ↓	AuPRC ↑	AuROC ↑	FPR_{95TPR} ↓
Maximum Softmax	30.1	93.0	33.2	10.0	95.5	17.9
ODIN	52.9	95.1	30.0	11.9	96.0	16.4
Mahalanobis	55.0	97.5	12.9	19.5	95.1	21.7
Monte Carlo Dropout	36.8	92.2	35.5	4.9	83.5	50.3
Void classifier	4.8	79.5	47.0	10.4	89.7	41.5
Embedding density	61.7	98.0	10.4	0.8	81.0	46.4
Image resynthesis	42.7	96.4	17.4	37.5	98.6	4.7
SynBoost	**81.7**	**98.3**	**4.6**	71.3	99.4	3.2
Entropy maximization	77.9	98.0	9.7	**76.0**	**99.7**	**1.3**

with entropy maximization. Since entropy maximization generally increases the sensitivity towards predicting anomalies, it is possible that the entropy is also increased at pixels belonging to non-anomalous objects. In the latter case, this would yield false positive anomaly instance predictions, which, however, can be identified and discarded afterwards by meta classification. The concept of trading false-positive detection for anomaly detection performance is motivated by [CRH+20]. Moreover, meta classifiers are expected to considerably benefit from entropy maximization, since in the original work [RCH+20] the entropy as metric has already been observed to be well correlated to the segment-wise IoU.

In our experiments on LostAndFound [PRG+16], we employ a logistic regression as meta classifier that is applied as a post-processing step on top of softmax probabilities. We observe that the meta classifier is capable of reliably removing false-positive anomaly instance predictions, which in turn significantly improves detection performance of anomalous objects. The meta classification performance is reported in Table 2, a visual example is given in Fig. 3. We note that meta classification is applied to segmentation masks as input. Therefore, the output of the combination of entropy maximization and meta classification does not yield pixel-wise anomaly scores to compare against the methods presented in Sect. 4.1.

The idea of meta classification can even be used to directly identify potential anomalous objects in the semantic segmentation mask, see [ORG18], which will

Table 2 Detection errors at object level for LostAndFound anomalies at different anomaly score / entropy thresholds τ (to generate anomaly segmentation masks). The quantities false-positives (FP) and false-negatives (FN) are reported at segment level, with anomalies as positive class. The F_1 summarizes these quantities into an overall measure. By δ we denote the performance loss on the original task, which is the semantic segmentation of Cityscapes. In this context, we consider a performance loss of 1% as acceptable, particularly in regard of a significantly improved anomaly detection performance

Anomaly score/entropy threshold	Entropy maximization + thresholding				Entropy maximization + thresholding + meta classifier			
$a_i \geq \tau, i \in \mathcal{I}$	FP ↓	FN ↓	F_1 ↑	δ in % ↓	FP ↓	FN ↓	F_1 ↑	δ in % ↓
$\tau = 0.30$	8,068	191	0.26	0.30	290	308	0.82	0.06
$\tau = 0.40$	4,035	289	0.39	0.11	251	359	0.81	0.03
$\tau = 0.50$	1,215	415	0.60	0.04	145	447	0.80	0.02
$\tau = 0.60$	327	613	0.69	0.02	49	619	0.76	0.02
$\tau = 0.70$	135	879	0.61	0.01	21	881	0.63	0.01

[without meta classifier]

[prediction quality rating]

[with meta classifier]

Fig. 3 Meta classification as quality rating of anomaly instance predictions. Before applying the meta classifier (top), the anomaly segmentation mask contains anomaly instance predictions (orange segments), with some false-positives on the road. Based on softmax probabilities, the meta classifier performs a prediction quality rating (bottom left, red corresponds to poor quality), which is then used to remove false positive anomaly instance predictions (bottom right). Note that the region of interest is restricted to the road, where ground truth anomalous objects (or obstacles) are indicated by green contours

be subject to discussion in the following section about unsupervised learning of unknown objects.

5 Discovering and Learning Novel Classes

If certain types of anomalies appear frequently, it might be reasonable to include them as additional learnable classes of the segmentation model. In this section, we propose an unsupervised method to further process anomaly predictions, all with the goal to produce labels corresponding to novel classes. Afterwards, we will introduce an incremental learning approach to train a model on novel classes by means of the retrieved unsupervised labels.

5.1 Unsupervised Identification and Segmentation of a Novel Class

Consider the dataset $\mathcal{D}^{\text{test}} \subseteq \mathcal{X}$ of unlabeled images $\mathbf{x} = (x_i)_{i \in \mathcal{I}} \in \mathbb{I}^{H \times W \times 3}$, along with a semantic segmentation network $\mathbf{F} : \mathbb{I}^{H \times W \times 3} \to \mathbb{R}^{H \times W \times S}$ trained on the set of classes $\mathcal{S} = \{1, \ldots, S\}$. Moreover, let $\mathbf{a} = (a_i)_{i \in \mathcal{I}} \in \mathbb{R}^{H \times W}$ denote a score map, as

introduced in Sect. 4, which assigns the degree of anomaly to each pixel $i \in \mathcal{I}$ in $\mathbf{x}$. Our unsupervised anomaly segmentation technique is a three-step procedure:

1. Image embedding: Image retrieval methods are commonly applied to construct a database of images that are visually related to a given image. On that account, such methods must quantify visual similarities, i.e., to measure the discrepancy or "distance" between images. A simple idea is averaging over the pixel-wise differences. However, this approach is extremely sensitive towards data transformation such as rotation, variation in light, or different resolutions. More advanced approaches make use of visual descriptors that extract the elementary characteristics of the visual contents, e.g., color, shape, or texture. These methods are invariant to data transformation, i.e., they perform well in identifying images representing the same item. If we want to detect different instances of the same category, deep learning methods represent the state-of-the-art. In this regard, convolutional neural networks (CNNs) achieve very high accuracy in image classification tasks. These networks extract features of the images, that are stable regarding transformations as well as the represented object itself, i.e., objects of the same category result in similar feature vectors. We now adapt this idea to identify anomalies that belong to the same class.

Let $\mathcal{K}_{\mathbf{a}|\mathbf{x}}$ denote the set of connected components within $(a_i^{(\tau)})_{i\in\mathcal{I}}$, $a_i^{(\tau)} := \mathbf{1}_{\{a_i \geq \tau\}}$ $\forall\, i \in \mathcal{I}$ for a given threshold $\tau \in \mathbb{R}$, after processing image $\mathbf{x}$. Furthermore, let $\mathcal{K} := \bigcup_{x\in\mathcal{X}} \mathcal{K}_{\mathbf{a}|\mathbf{x}}$ denote the set of all predicted anomaly components in $\mathcal{D}^{\text{test}}$. For each component $k \in \mathcal{K}_{\mathbf{a}|\mathbf{x}}$, we tailor the input $\mathbf{x}$ to the image crop $\mathbf{x}^{(k)} = (x_i)_{i\in\mathcal{I}'}$, $\mathcal{I}' \subseteq \mathcal{I}$ by means of the bounding box around $k \in \mathcal{K}_{\mathbf{a}|\mathbf{x}}$. By feeding the crop $\mathbf{x}^{(k)}$ to an image classification network $\mathbf{G}$, we map $\mathbf{x}^{(k)}$ onto its feature vector $\mathbf{g}^{(k)} := \mathbf{G}_{L-1}(\mathbf{x}^{(k)}) \in \mathbb{R}^n$, $n \in \mathbb{N}$ for all $k \in \mathcal{K}$. Here, $\mathbf{G}_{L-1}$ denotes the output of the penultimate layer of $\mathbf{G}$.

2. Dimensionality reduction: Feature vectors extracted by CNNs are usually very high-dimensional. This evokes several problems regarding the clustering of such data. The first issue is known as *curse of dimensionality*, i.e., the amount of required data explodes with increasing dimensionality. Furthermore, distance metrics become less precise. Dimensionality reduction approaches project the feature vectors onto a low-dimensional representation, either by feature elimination, selection, or extraction. The latter creates new independent features as a combination of the original vectors and can be further distinguished between linear and non-linear techniques. A linear feature extraction approach, named principal component analysis (PCA) [Pea01], aims at decorrelating the components of the vectors by a change of basis, such that they are mostly aligned along the first axes. Thereby, not much information is lost if we drop the last components. A more recent non-linear method is t-distributed stochastic neighbor embedding (t-SNE) [vdMH08], which uses conditional probabilities representing pairwise similarities. Let us consider two feature vectors $\mathbf{g}^{(k)}, \mathbf{g}^{(k')}$ with $k, k' \in \mathcal{K}$ and let $\mathrm{p}_{k|k'} \in \mathbb{I}$ denote their similarity under a Gaussian distribution. Employing a Student t-distribution with one degree of freedom in the low-dimensional space then provides a second probability $\mathrm{q}_{k|k'} \in \mathbb{I}$. Hence, t-SNE aims at minimizing the following sum (or Kullback-Leibler divergence) [vdMH08]

$$\sum_{k\in\mathcal{K}}\sum_{k'\in\mathcal{K}} \mathrm{p}_{k|k'} \log\left(\frac{\mathrm{p}_{k|k'}}{\mathrm{q}_{k|k'}}\right) \tag{32}$$

using gradient descent. We first perform dimensionality reduction via PCA, which is then followed by t-SNE. In our experiments, we observed that this combination of methods improves the effectiveness of mapping anomaly predictions onto a two-dimensional embedding space. Here, the embedding ideally creates neighborhoods of visually related anomalies.

3. Novelty segmentation: If anomalies of the same category are detected more frequently, they are expected to form a bigger cluster in the embedding space. Those clusters can be identified by employing algorithms such as density-based spatial clustering of applications with noise (DBSCAN) [EKSX96]. This algorithm supports the idea of non-supervision since it does not require any information of the potential anomaly data, such as e.g., the number of clusters. Moreover, DBSCAN divides data points into core points, border points, and noise, depending on the size of the neighborhood $\varepsilon \in [0, \infty)$ and the minimal number of a core point's neighbors $\delta \in \mathbb{N}$.

More precisely, let $\tilde{\mathbf{g}}^{(k)} \in \mathbb{R}^2$ denote the two-dimensional representation of $\mathbf{x}^{(k)}$. Then, $\tilde{\mathbf{g}}^{(k)}$ is considered as a core point, if the corresponding point-wise density $\rho(\tilde{\mathbf{g}}^{(k)}) := |\{\tilde{\mathbf{g}}^{(k')} : \|\tilde{\mathbf{g}}^{(k)} - \tilde{\mathbf{g}}^{(k')}\| < \varepsilon, k' \in \mathcal{K}\}| \geq \delta$, i.e., the ε-neighborhood of $\tilde{\mathbf{g}}^{(k)}$ contains at least δ points including itself. We denote the neighborhood of a core point $\tilde{\mathbf{g}}^{(\mathring{k})}$, which corresponds to a component $\mathring{k} \in \mathcal{K}$, as $B_{\mathring{k}} := \{\tilde{\mathbf{g}}^{(k')} : \|\tilde{\mathbf{g}}^{(\mathring{k})} - \tilde{\mathbf{g}}^{(k')}\| < \varepsilon, k' \in \mathcal{K}\}$. If $\tilde{\mathbf{g}}^{(k)}$ is not a core point but belongs to a core point's neighborhood, we call it a border point. Otherwise, i.e., if $\tilde{\mathbf{g}}^{(k)}$ is neither a core point nor within a core point's neighborhood, we call it noise.

Finally, a cluster $\mathcal{C}_j \subset \mathcal{K}$, $j \in \mathcal{J} := \{1, \ldots, J\}$ of components is formed by merging overlapping neighborhoods $B_{\mathring{k}}$, yielding $J \in \mathbb{N}$ clusters in total. In other words, clusters are formed from connected core points and their neighborhoods' border points. Given $\rho(\tilde{\mathbf{g}}^{(k)})$, we can determine the cluster density of $\mathcal{C}_j$, e.g.,

$$\text{as the maximum } \max_{k\in\mathcal{C}_j} \rho(\tilde{\mathbf{g}}^{(k)}) \text{ or as the average } \frac{1}{|\mathcal{C}_j|}\sum_{k\in\mathcal{C}_j} \rho(\tilde{\mathbf{g}}^{(k)}) \,.$$

The cluster $\mathcal{C}^* \subset \mathcal{K}$, which is the cluster of highest density given a sufficient cluster size, is then selected to be further processed. To this end, let us consider the predicted segmentation mask $\mathbf{F}(\mathbf{x}) = \mathbf{m} = (m_i)_{i\in\mathcal{I}}$, where $m_i = \arg\max_{s\in\mathcal{S}} y_{i,s}$, $i \in \mathcal{I}$. The pseudo labels $\tilde{\mathbf{y}} = (\tilde{y}_i)_{i\in\mathcal{I}}$ for the originally unlabeled $\mathbf{x}$ are then obtained by setting $\tilde{y}_i = S + 1$ if pixel location i belongs to a component $k \in \mathcal{C}^*$, and $\tilde{y}_i = m_i$ otherwise.

5.2 Class-Incremental Learning

Let $\tilde{\mathcal{Y}}$ denote the set of pseudo labels, then the training data for some novel class $S + 1$ can be represented by $\mathcal{D}^{S+1} \subseteq \mathcal{D}^{\text{novel}} \times \tilde{\mathcal{Y}}$, where $\mathcal{D}^{\text{novel}}$ denotes the set of previously-unseen images containing novel classes. By extending the semantic segmentation network $\mathbf{F}$ to $\mathbf{F}^+ : \mathbb{I}^{H\times W\times 3} \to \mathbb{R}^{H\times W\times(S+1)}$ and retraining $\mathbf{F}^+$ on $\mathcal{D}^{S+1}$, we perform

incremental learning to add a novel and previously unknown class to the semantic space of $\mathbf{F}$.

Regularization: Knowledge distillation is a subcategory of regularization strategies aiming to mitigate a catastrophic forgetting, i.e., these strategies try to mitigate performance loss on the previously-learned classes $\mathcal{S} = \{1, \ldots, S\}$ while learning the additional class $S + 1$. In [MZ19], the authors adapted incremental learning techniques to the task of semantic segmentation. Among others, they introduced the overall objective

$$J^{\text{total}}(\mathbf{x}, \tilde{\mathbf{y}}) = (1 - \lambda) J^{\text{CE}}(\mathbf{F}^+(\mathbf{x}), \tilde{\mathbf{y}}) + \lambda J^{\text{D}}(\mathbf{F}^+(\mathbf{x}), \mathbf{F}(\mathbf{x})), \quad \lambda \in \mathbb{I}\,, \tag{33}$$

where $(\mathbf{x}, \tilde{\mathbf{y}}) \in \mathcal{D}^{S+1}$. Here, J^{CE} denotes the common cross-entropy loss over the enlarged set of class indices $\mathcal{S}^+ := \{1, \ldots, S + 1\}$ and J^{D} the distillation loss. The latter loss is defined as

$$J^{\text{D}}(\mathbf{F}^+(\mathbf{x}), \mathbf{F}(\mathbf{x})) := -\frac{1}{H \cdot W} \sum_{i \in \mathcal{I}} \sum_{s \in \mathcal{S}} \text{softmax}_s(\mathbf{y}_i) \log(\text{softmax}_s(\mathbf{y}_i^+)) \tag{34}$$

with $\mathbf{y} = \mathbf{F}(\mathbf{x})$ and $\mathbf{y}^+ = \mathbf{F}^+(\mathbf{x})$. Knowledge distillation can be further improved by freezing the weights of the encoder part of $\mathbf{F}^+$ during the training procedure [MZ19].

Rehearsal: If the original training data $\mathcal{D}^{\text{train}} \subseteq \mathcal{X} \times \mathcal{Y}$ of network $\mathbf{F}$ is available, in incremental learning such data is usually re-integrated into the training set of the extended network $\mathbf{F}^+$, i.e., the training samples are drawn from $\mathcal{D}^{\text{train}} \cup \mathcal{D}^{S+1}$. To save computational costs of training and to balance the amount of old and new training data, established methods, e.g., [Rob95], only use a subset of $\mathcal{D}^{\text{train}}$. This subset is typically obtained by randomly sampling a set from $\mathcal{D}^{\text{train}}$ that matches the size of $|\mathcal{D}^{S+1}|$.

In combination with knowledge distillation, rehearsal strategies can be employed to mitigate a loss of performance on classes that are related to the novel class. This issue may arise e.g., through visual similarity such as between classes like *bus* and *train*, or due to class affiliation as in the case of *bicycle* and *rider*. Relevant classes can be identified by their frequency of being predicted on the relabeled pixels, i.e.,

$$\nu_s^{\text{tot}} := \sum_{(\mathbf{x}, \tilde{\mathbf{y}}) \in \mathcal{D}^{S+1}} |\{i \in \mathcal{I} \mid m_i = s \ \wedge \ \tilde{y}_i = S + 1\}| \ \ \forall s \in \mathcal{S}\,, \tag{35}$$

and hence

$$\nu_s^{\text{rel}} := \frac{\nu_s^{\text{tot}}}{\sum_{s' \in \mathcal{S}} \nu_{s'}^{\text{tot}}} \ \ \forall s \in \mathcal{S}\,. \tag{36}$$

The subset of $\mathcal{D}^{\text{train}}$ is then randomly sampled under the constraint that there are at least $\nu_s^{\text{rel}} |\mathcal{D}^{S+1}|$ images containing the class s for all $s \in \mathcal{S}$.

Fig. 4 Predicted semantic segmentation mask of the initial model's prediction (top) and corresponding segment-wise quality estimation (bottom) for one example from the Cityscapes test split. Green color indicates a high segment-wise IoU, red color indicates a low one

5.3 Experiments and Evaluation

In the following experiments, we will employ a `DeepLabV3+` [CZP+18] model with an underlying `WiderResNet38` [ZSR+19] backbone for semantic segmentation. This network is initially trained on a set of 17 classes, which we will extend by a novel class. The already trained classes are the Cityscapes training classes except *pedestrian* and *rider*, i.e., we exclude any *human* in the training process of our initial semantic segmentation network **F**.

The initial model was trained on the Cityscapes [COR+16] training data. For the incremental learning process, we use a portion of those data and combine them with our generated disjoint training set $\mathcal{D}^{S+1}$ containing previously unseen images and pseudo labels on novel objects. Here, the images from $\mathcal{D}^{S+1}$ are drawn from the Cityscapes test data. For evaluation purposes, we use the Cityscapes validation data. Hence, during the incremental learning process only known objects are presented to the model except humans and a few instances, such as the ego-car or mountains in an image background, belonging to the Cityscapes void category.

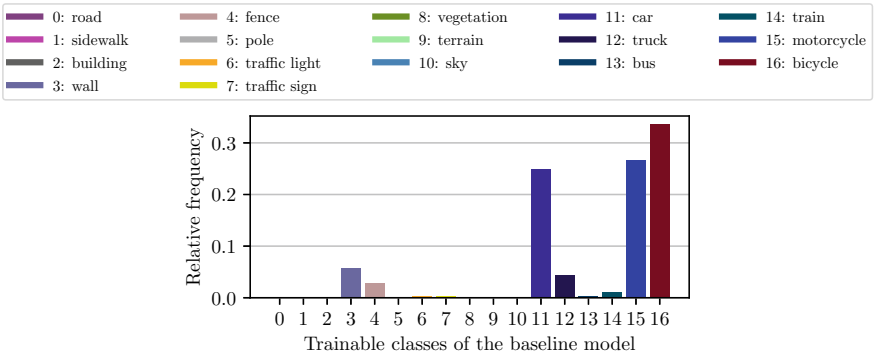


Fig. 5 Relative frequency of old classes being predicted by the initial model on pixels that are assigned to the novel class. Thus, the subset of $\mathcal{D}_{\text{CS}}^{\text{train}}$ included in the retraining should mainly involve bicycles, motorcycles, and cars

We use the idea of meta classification, similarly as introduced in Sect. 4.3, to rate the prediction quality of predicted semantic segmentation masks. Here, the meta task is to estimate the segment-wise IoU first, see Fig. 4, on which we apply thresholding (at $\tau = 0.5$) to determine potential anomalies, cf. [ORF20]. We employ gradient boosting as meta model, which achieves a coefficient of determination of $R^2 = 82.51\%$ in estimating the segment-wise IoU on the Cityscapes validation split.

In accordance to Sect. 2 and as already observed in Sect. 4.3, the softmax entropy is again one of the main metrics included in the meta model to identify anomalous predictions. Thus, the entropy shows to have great impact on meta classification performance, which, similarly, has also been observed in [CRH+20, CRG21].

Given anomaly segmentation masks, we perform image embedding using the encoder of the image classification network `DenseNet201` [HLMW17], that is pre-trained on ImageNet [DDS+09]. Next, we reduce the dimensionality of the resulting feature vectors to 50 via PCA and further to 2 by applying t-SNE. In [ORF20], a qualitative and quantitative evaluation of different embedding approaches is provided. Note that t-SNE is non-deterministic, i.e., we obtain slightly different embedding spaces for different runs. In our experiment, employing DBSCAN with parameters $\varepsilon = 2.5$ and $\delta = 15$ produces a *human*-cluster including 91 components from 76 different images. The most frequently predicted class of these components are *car*, *motorcycle*, and *bicycle* with $\nu_{11}^{\text{rel}} = 24.84\%$, $\nu_{15}^{\text{rel}} = 26.69\%$ and $\nu_{16}^{\text{rel}} = 33.53\%$, respectively, see Fig. 5.

We train the extended model $\mathbf{F}^+$ as described in Sect. 5.2 for 70 epochs, weighting the loss functions in (33) equally, i.e., $\lambda = 0.5$. The extended model shows the ability to retain its initial knowledge by achieving an mIoU score of 68.24% on the old classes when evaluating on the Cityscapes validation data. This yields a marginal loss of only 0.39% compared to the initial model $\mathbf{F}$. At the same time, $\mathbf{F}^+$ predicts the novel human class with a class IoU of 41.42%, without a single annotated human instance in the training data $\mathcal{D}^{S+1}$. A visual example of our unsupervised novelty

[Prediction of initial model]

[Novel class in validation image]

[Prediction of extended model]

Fig. 6 Comparison of the predicted semantic segmentation masks before (top) and after (bottom) adapting the model to the novel *human* class (orange) for one example of the Cityscapes validation split (middle). Here, the novel components are highlighted in orange, green contours indicate the ground truth annotation of the novelty

Fig. 7 A comparison of anomaly scores obtained by meta classification (left) and entropy (right) on an image from RoadObstacle21. The dog on the image is the anomaly of interest (indicated by green contours), which would have been overlooked by meta classification but entirely detected by the entropy

segmentation approach is provided in Fig. 6, more details on the numerical evaluation is given in Table 3.

Table 3 Evaluation of the Cityscapes validation split before and after incremental learning the novelty *human* (highlighted in gray) with knowledge distillation and rehearsal. The classes *pedestrian* and *rider* are aggregated to the novel class *human*. All other classes are treated as background, i.e., they are ignored during training, regarding the data from $\mathcal{D}^{S+1}$

	metric	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	human	car	truck	bus	train	motorcycle	bicycle	mean excl. human	mean incl. human
initial	IoU	97.34	80.63	88.91	47.24	51.03	52.90	55.44	66.66	89.95	56.29	93.76	00.00	90.61	69.66	76.90	70.35	24.45	54.57	68.63	64.82
	Precision	98.35	89.39	92.80	74.57	66.76	72.68	75.04	86.22	93.60	77.66	96.38	00.00	92.97	80.23	88.59	83.33	28.57	59.30	79.79	75.36
	Recall	98.96	89.16	95.50	56.32	68.41	66.02	67.98	74.61	95.85	67.17	97.18	00.00	97.27	84.09	85.35	81.87	62.92	87.24	80.94	76.44
ours	IoU	97.46	80.78	89.30	47.48	49.31	53.25	55.28	65.72	90.17	54.53	93.47	41.42	91.21	69.30	72.52	62.06	30.45	57.72	68.24	66.75
	Precision	98.68	89.31	93.11	78.33	69.48	73.74	76.02	88.99	94.21	75.66	95.69	59.73	95.26	84.88	87.26	91.68	64.38	76.01	84.28	82.91
	Recall	98.75	89.43	95.62	54.67	62.95	65.70	66.96	71.54	95.46	66.13	97.57	57.48	95.45	79.06	81.11	65.76	36.61	70.57	76.08	75.05

5.4 Outlook on Improving Unsupervised Learning of Novel Classes

In the preliminary experiments presented in this section, we demonstrated that a semantic segmentation network can be extended by a novel class in an unsupervised fashion. As a basis to start, our unsupervised learning approach requires anomaly segmentation masks. Currently, these are obtained by meta classification [RCH+20], which is, however, not a method specifically tailored for the task of anomaly segmentation. In other words, the obtained masks are possibly inaccurate. To be even more precise on the limitation of plain meta classification, this method is only able to find anomalies when the segmentation model produces a (false positive) object prediction on those anomalies. By design, meta classifiers cannot find overlooked instances, e.g., obstacles on the road which also have been classified as road. As an illustration of this issue, we refer to Fig. 7.

Having now several methods at hand, that we, e.g., introduced in Sect. 4.1, it seems obvious to replace the underlying anomaly segmentation method by a more sophisticated one as future work. In particular, given the decent performance of our unsupervised learning approach relying only on meta classification and the entropy measure as highly beneficial metric for meta classification, combining entropy maximization and meta classification is a promising approach to improve the presented novelty training approach.

Conclusions

Semantic segmentation as a supervised learning task is typically performed by models that operate on a given set containing a fixed number of classes. This is in clear contrast to the open world scenarios to which practitioners contemplate the usage of segmentation models. There are important capabilities that standard segmentation models do not exhibit. Among them is the capability to know when they face an object of a class they have not learned – i.e., to perform anomaly segmentation – as well as the capability to realize that similar objects, presumably of the same (yet unknown) class, appear frequently and should be learned either as a new class or be attributed to an existing one. In this chapter, we have seen first promising results for two tasks, for anomaly segmentation as well as for the detection and unsupervised learning of new classes.

For anomaly segmentation, we considered a number of generic baseline methods stemming from image classification as well as some recent anomaly segmentation methods. Since the latter clearly outperforms the former, this stresses the need for the development of methods specifically designed for anomaly segmentation. We have demonstrated with our entropy maximization method empirically as well as theoretically that good proxies in combination with training on anomaly examples for high entropy are key to equip standard semantic segmentation models with anomaly

segmentation capabilities. Particularly on the challenging RoadObstacle21 dataset with diverse street scenarios, entropy maximization yields great performance which is not reached by any other method so far. While there exists a moderate number of datasets for anomaly segmentation, there is clearly still the need of additional datasets. The number of possible unknown object classes not covered by these datasets is evidently enormous. Furthermore, also the vast variety of possible environmental conditions and further domain shifts that may occur, possibly also in combination with unknown objects, continuously demand their exploration.

For detection and unsupervised learning of new classes, we demonstrated in preliminary experiments that a combination of well-established dimensionality reduction and clustering methods along with the advanced uncertainty quantification method for semantic segmentation called MetaSeg is well able to detect unknown classes of which objects appear relatively frequently in a given test set. Indeed, MetaSeg can also be used to define segmentation proposals for pseudo ground-truth of new classes, which can also be learned incrementally by the segmentation model. For the considered scenario of subsequently learning humans within the Cityscapes dataset, this approach yields an IoU of 41.42% on the novel class without losing performance on the original classes. The proposed methodology may help to incorporate new classes into existing models with low human labeling effort. The necessity for this will occur repeatedly in future. An example are the electric scooters that recently arose in several metropolitan areas across the globe. This is an example for a global phenomenon. However, also local phenomena, such as boat trailers at the coast, could be of interest. Such classes can be initially incorporated into an existing model using our methodology. Afterwards, the initial performance could be further improved with active learning approaches, such as [CRGR21], still requiring only a small amount of human labeling effort. It is also an open question, to which extent the proposed method can be used iteratively to improve the performance on a new class. Also for this track of research, the lack of data for pursuing that task is a limiting factor as of now.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the projects "KI Absicherung - Safe AI for Automated Driving", grant no. 19A19005R, and "KI Delta Learning - Scalable AI for Automated Driving", grant no. 19A19013Q, respectively. The authors would like to thank the consortia for the successful cooperation. The authors gratefully also acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

References

[AAM+19] A. Atanov, A. Ashukha, D. Molchanov, K. Neklyudov, D. Vetrov, Uncertainty estimation via stochastic batch normalization, in *Proceedings of the International Symposium on Neural Networks (ISNN)* (Moscow, Russia, 2019), pp. 261–269

[ABE+18] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, T. Tuytelaars, Memory Aware Synapses: Learning what (not) to forget, in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 139–154

[ACS19] M. Angus, K. Czarnecki, R. Salay, Efficacy of Pixel-Level OOD Detection for Semantic Segmentation (November 2019), pp. 1–13. arXiv:1911.02897

[ACT17] R. Aljundi, P. Chakravarty, T. Tuytelaars, Expert gate: lifelong learning with a network of experts, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI, USA, July 2017), pp. 7120–7129

[AGLR21] H. Asatryan, H. Gottschalk, M. Lippert, M. Rottmann, A Convenient Infinite Dimensional Framework for Generative Adversarial Learning (January 2021), pp. 1–29. arXiv:2011.12087

[AHK01] C. Aggarwal, A. Hinneburg, D. Keim, On the surprising behavior of distance metrics in high dimensional spaces, in *Proceedings of the International Conference on Database Theory* (London, UK, January 2001), pp. 420–434

[AM19] M. Amer, T. Maul, Reducing Catastrophic Forgetting in Modular Neural Networks by Dynamic Information Balancing (December 2019), pp. 1–15. arXiv:1912.04508

[AR05] W. Abraham, A. Robins, Memory retention – the synaptic stability versus plasticity dilemma. Trends Neurosci. **28**(2), 73–78 (2005)

[ARC20] S. Agarwal, A. Rattani, C.R. Chowdary, A-iLearn: An adaptive incremental learning model for spoof fingerprint detection. Machine Learning with Applications, 7, 100210 (2020)

[AY01] C.C. Aggarwal, P.S. Yu, Outlier detection for high dimensional data, in *Proceedings of the ACM SIGMOD International Conference on Management of Data (MOD)* (May 2001), pp. 37–46

[BBSC21] G. Di Biase, H. Blum, R. Siegwart, C. Cadena, Pixel-Wise Anomaly Detection in Complex Driving Scenes, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 16918–16927

[BKC17] V. Badrinarayanan, A. Kendall, R. Cipolla, Bayesian SegNet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding, in *Proceedings of the British Machine Vision Conference (BMVC)* (2017), pp. 1–12

[BKNS00] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers. ACM SIGMOD Rec. **29**(2), 93–104 (2000)

[BKOŠ19] P. Bevandić, I. Krešo, M. Oršić, S. Šegvić, Simultaneous semantic segmentation and outlier detection in presence of domain shift, in *Proceedings of the German Conference on Pattern Recognition (GCPR)* (Dortmund, Germany, 2019), pp. 33–47

[BSN+19] H. Blum, P.-E. Sarlin, J. Nieto, R. Siegwart, C. Cadena, Fishyscapes: a benchmark for safe semantic segmentation in autonomous driving, in *Proceedings of the IEEE Inter-*

national Conference on Computer Vision (ICCV) Workshops (Seoul, Korea, October 2019), pp. 2403–2412

[CDAT18] A. Chaudhry, P.K. Dokania, T. Ajanthan, P.H.S. Torr, Riemannian walk for incremental learning: understanding forgetting and intransigence, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Munich, Germany, September 2018), pp. 556–572

[CLU+21] R. Chan, K. Lis, S. Uhlemeyer, H. Blum, S. Honari, R. Siegwart, P. Fua, M. Salzmann, M. Rottmann, SegmentMeIfYouCan: a benchmark for anomaly segmentation, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) – Datasets and Benchmarks Track*, virtual conference (December 2021), pp. 1–35

[CM15] C. Creusot, A. Munawar, Real-time small obstacle detection on highways using compressive RBM road reconstruction, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)* (Seoul, Korea, June 2015), pp. 162–167

[CMB+20] F. Cermelli, M. Mancini, S. Rota Bulo, E. Ricci, B. Caputo, Modeling the background for incremental learning in semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (June 2020), pp. 9233–9242

[CMJM+18] F.M. Castro, M.J. Marín-Jiménez, N. Guil, C. Schmid, K. Alahari, End-to-End Incremental Learning, in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 233–248

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The Cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, June 2016), pp. 3213–3223

[CRG21] R. Chan, M. Rottmann, H. Gottschalk, Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, virtual conference (October 2021), pp. 5128–5137

[CRGR21] P. Colling, L. Roese-Koerner, H. Gottschalk, M. Rottmann, MetaBox+: a new region based active learning method for semantic segmentation using priority maps, in *Proceedings of the International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, virtual conference (February 2021), pp. 51–62

[CRH+20] R. Chan, M. Rottmann, F. Hüger, P. Schlicht, H. Gottschalk, Controlled false negative reduction of minority classes in semantic segmentation, in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, virtual conference (July 2020), pp. 1–8

[CSMDB19] A. Chefrour, L. Souici-Meslati, I. Difi, N. Bakkouche, A novel incremental learning algorithm based on incremental vector support machina and incremental neural network learn++. Revue d'Intelligence Artificielle **33**(3), 181–188 (2019)

[Cyb89] G. Cybenko, Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. **2**(4), 303–314 (1989)

[CZP+18] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Munich, Germany, September 2018), pp. 833–851

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, F.-F. Li, ImageNet: a large-scale hierarchical image database, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Miami, FL, USA, June 2009), pp. 248–255

[DHF07] L. De Haan, A. Ferreira, *Extreme Value Theory: An Introduction* (Springer, 2007)

[DT18] T. DeVries, G.W. Taylor, Learning Confidence for Out-of-Distribution Detection in Neural Networks (February 2018), pp. 1–12. arXiv:1802.04865

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in *Proceedings of the ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining (KDD) (August 1996), pp. 226–231

[FAML19] M. Farajtabar, N. Azizan, A. Mott, A. Li, Orthogonal gradient descent for continual learning, in *International Conference on Artificial Intelligence and Statistics* (2020, June), pp. 3762–3773. PMLR

[GDS20] F.K. Gustafsson, M. Danelljan, T. Bo Schön, Evaluating scalable Bayesian deep learning methods for robust computer vision, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, virtual conference (June 2020), pp. 1289–1298

[GG16] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, in *Proceedings of the International Conference on Machine Learning (ICML)* (New York, NY, USA, June 2016), pp. 1050–1059

[GK16] A. Gepperth, C. Karaoguz, A bio-inspired incremental learning architecture for applied perceptual problems. Cogn. Comput. **8**(5), 924–934 (2016)

[HAB19] M. Hein, M. Andriushchenko, J. Bitterwolf, Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, June 2019), pp. 41–50

[HBM+20] D. Hendrycks, S. Basart, M. Mazeika, M. Mostajabi, J. Steinhardt, D. Song, Scaling Out-of-Distribution Detection for Real-World Settings (December 2020), pp. 1–12. arXiv:1911.11132

[HCP+19] C. Hu, Y. Chen, X. Peng, H. Yu, C. Gao, L. Hu, A novel feature incremental learning method for sensor-based activity recognition. IEEE Trans. Knowl. Data Eng. **31**(6), 1038–1050 (2019)

[HG17] D. Hendrycks, K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Toulon, France, April 2017), pp. 1–12

[HLMW17] G. Huang, Z. Liu, L. van der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI, USA, July 2017), pp. 4700–4708

[HMD19] D. Hendrycks, M. Mazeika, T. Dietterich, Deep anomaly detection with outlier exposure, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, May 2019), pp. 1–18

[HMKS19] D. Hendrycks, M. Mazeika, S. Kadavath, D. Song, Using self-supervised learning can improve model robustness and uncertainty, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Vancouver, BC, Canada, December 2019), pp. 15637–15648

[HPL+18] S. Hou, X. Pan, C. Change Loy, Z. Wang, D. Lin, Lifelong learning via progressive distillation and retrospection, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Munich, Germany, August 2018), pp. 452–467

[HTF07] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning* (Springer, 2007)

[HVD14] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops* (Montréal, QC, Canada, December 2014), pp. 1–9

[HZ21] J. He, F. Zhu, Unsupervised Continual Learning via Pseudo Labels (July 2021), pp. 1–9. arXiv:2104.07164

[JJJK18] H. Jung, J. Ju, M. Jung, J. Kim, Less-forgetful learning for domain expansion in deep neural networks, in *Proceedings of the AAAI Conference on Artificial Intelligence* (New Orleans, LA, USA, February 2018), pp. 3358–3365

[JRF20] N. Jourdan, E. Rehder, U. Franke, Identification of uncertainty in artificial neural networks, in *Proceedings of the Uni-DAS e.V. Fahrerassistenz und automatisiertes Fahren (FAS) Workshop*, virtual conference (July 2020), pp. 1–11

[KBDFs20] M. Klingner, A. Bär, P. Donn, T. Fingscheidt, Class-incremental learning for semantic segmentation re-using neither old data nor old labels, in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)*, virtual conference (September 2020), pp. 1–8

[KBJC19] D. Kim, J. Bae, Y. Jo, J. Choi, Incremental Learning With Maximum Entropy Regularization: Rethinking Forgetting and Intransigence (February 2019), pp. 1–10. arXiv:1902.00829

[KG17] A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision? in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Long Beach, CA, USA, December 2017), pp. 5574–5584

[KKSZ09] H.-P. Kriegel, P. Kröger, E. Schubert, A. Zimek, Outlier detection in axis-parallel subspaces of high dimensional data, in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (Bangkok, Thailand, April 2009), pp. 831–838

[Kle09] J. Sakari Klemelä, *Smoothing of Multivariate Data: Density Estimation and Visualization* (Wiley, 2009)

[KNT00] E.M. Knorr, R.T. Ng, V. Tucakov, Distance-based outliers: algorithms and applications. Int. J. Very Larg. Data Bases **8**(3–4), 237–253 (2000)

[KPR+17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, R. Hadsell, Overcoming Catastrophic Forgetting in Neural Networks, in *Proceedings of the national academy of sciences*, 114(13), 3521–3526 (2017)

[KSZ08] H.-P. Kriegel, M. Schubert, A. Zimek, Angle-based outlier detection in high-dimensional data, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (Las Vegas, NV, USA, August 2008), pp. 444–452

[LH18] Z. Li, D. Hoiem, Learning without forgetting. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **40**, 2935–2947 (2018)

[LHFS21] K. Lis, S. Honari, P. Fua, M. Salzmann, Detecting road obstacles by erasing them (April 2021), pp. 1–18. arXiv:2012.13633

[LKJ+17] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, B.-T. Zhang, Overcoming Catastrophic Forgetting by Incremental Moment matching. Advances in neural information processing systems, 30 (2017)

[LLLS18] K. Lee, K. Lee, H. Lee, J. Shin, A simple unified framework for detecting out-of-distribution samples and adversarial attacks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, December 2018), pp. 7167–7177

[LLS18] S. Liang, Y. Li, R. Srikant, Enhancing the reliability of out-of-distribution image detection in neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Vancouver, BC, Canada, April 2018), pp. 1–15

[LLSL19] K. Lee, K. Lee, J. Shin, H. Lee, Overcoming catastrophic forgetting with unlabeled data in the wild, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, October 2019), pp. 312–321

[LMB+14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. Lawrence Zitnick, Microsoft COCO: Common Objects in Context, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Zurich, Switzerland, September 2014), pp. 740–755

[LNFS19] K. Lis, K. Nakka, P. Fua, M. Salzmann, Detecting the Unexpected via Image Resynthesis, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, October 2019), pp. 2152–2161

[LPB17] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Long Beach, CA, USA, December 2017), pp. 6402–6413

[Mac92] D.J.C. MacKay, A practical Bayesian framework for backpropagation networks. Neural Comput. **4**(3), 448–472 (1992)

[Mah36] P. Chandra Mahalanobis, On the generalized distance in statistics. Proc. Natl. Inst. India **2**(1), 49–55 (1936)

[MCE20] J.K. Mandivarapu, B. Camp, R. Estrada, Self-net: Lifelong learning via continual self-modeling. Frontiers in artificial intelligence, 3, 19

[MC89] M. McCloskey, N. Cohen, Catastrophic interference in connectionist networks: the sequential learning problem. Psychol. Learn. Motiv. **24**, 109–165 (1989)

[MG19] J. Mukhoti, Y. Gal, Evaluating Bayesian deep learning methods for semantic segmentation (March 2019), pp. 1–13. arXiv:1811.12709

[MH20] A. Meinke, M. Hein, Towards neural networks that provably know when they don't know, in *Proceedings of the International Conference on Learning Representations (ICLR)*, virtual conference (April 2020), pp. 1–18

[MSC+19] D. Mellado, C. Saavedra, S. Chabert, R. Torres, R.F. Salas, Self-improving generative artificial neural network for pseudorehearsal incremental class learning. Algorithms **12**(10), 1–17 (2019)

[MVD17] A. Munawar, P. Vinayavekhin, G. De Magistris, Limiting the reconstruction capability of generative neural network using negative learning, in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)* (Tokyo, Japan, September 2017), pp. 1–6

[MZ19] U. Michieli, P. Zanuttigh, Incremental learning techniques for semantic segmentation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops* (Seoul, Korea, October 2019), pp. 3205–3212

[MZ21] U. Michieli, P. Zanuttigh, Knowledge distillation for incremental learning in semantic segmentation. Comput. Vis. Image Underst. **205**, 103167 (2021)

[Nea96] R.M. Neal, *Bayesian Learning for Neural Networks* (Springer, 1996)

[OOS17] A. Odena, C. Olah, J. Shlens, Conditional image synthesis with auxiliary classifier GANs, in *Proceedings of the International Conference on Machine Learning (ICML)* (Sydney, NSW, Australia, August 2017), pp. 2642–2651

[OPK+19] O. Ostapenko, M. Puscas, T. Klein, P. Jähnichen, M. Nabi, Learning to remember: a synaptic plasticity driven framework for continual learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, June 2019), pp. 11313–11321

[ORF20] P. Oberdiek, M. Rottmann, G.A. Fink, Detection and retrieval of out-of-distribution objects in semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (June 2020), pp. 1331–1340

[ORG18] P. Oberdiek, M. Rottmann, H. Gottschalk, Classification uncertainty of deep neural networks based on gradient information, in *Proceedings of the IAPR TC3 Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR)* (Siena, Italy, September 2018), pp. 113–125

[Pea01] K. Pearson, On lines and planes of closest fit to systems of points in space. Lond. Edinb. Dublin Philos. Mag. J. Sci. **2**(11), 559–572 (1901)

[PKGF03] S. Papadimitriou, H. Kitagawa, P.B. Gibbons, C. Faloutsos, LOCI: fast outlier detection using the local correlation integral, in *Proceedings of the International Conference on Data Engineering* (Bangalore, India, March 2003), pp. 315–326

[PRG+16] P. Pinggera, S. Ramos, S. Gehrig, U. Franke, C. Rother, R. Mester, Lost and found: detecting small road hazards for self-driving vehicles, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Daejeon, Korea, October 2016), pp. 1099–1106

[PTJ+21] J. Peng, B. Tang, H. Jiang, Z. Li, Y. Lei, T. Lin, H. Li, Overcoming long-term catastrophic forgetting through adversarial neural pruning and synaptic consolidation, in *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* (February 2021) early access, pp. 1–14

[PUUH01] R. Polikar, L. Upda, S.S. Upda, V. Honavar, Learn++: an incremental learning algorithm for supervised neural networks. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **31**(4), 497–508 (2001)

[RCH+20] M. Rottmann, P. Colling, T.-P. Hack, R. Chan, F. Hüger, P. Schlicht, H. Gottschalk, Prediction error meta classification in semantic segmentation: detection via aggregated dispersion measures of softmax probabilities, in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, virtual conference (July 2020), pp. 1–9

[RKSL17] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert, iCaRL: incremental classifier and representation learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI, USA, July 2017), pp. 5533–5542

[Rob95] A.V. Robins, Catastrophic forgetting, rehearsal and pseudorehearsal. Connect. Sci. **7**(2), 123–146 (1995)

[Ros52] M. Rosenblatt, Remarks on a multivariate transformation. Ann. Math. Stat. **23**(3), 470–472 (1952)

[RPR20] D. Roy, P. Panda, K. Roy, Tree-CNN: A Hierarchical Deep Convolutional Neural Network for Incremental Learning. *Neural Networks*, 121, 148–160 (2020)

[RRD+16] A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive Neural Networks (September 2016), pp. 1–14. arXiv:1606.04671

[RRS00] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in *Proceedings of the ACM SIGMOD International Conference on Management of Data (MOD)* (May 2000), pp. 427–438

[Rud87] W. Rudin, *Real and Complex Analysis* (McGraw-Hill Inc., 1987)

[SAR20] S. Shakib Sarwar, A. Ankit, K. Roy, Incremental learning in deep convolutional neural networks using partial network sharing. IEEE Access **8**, 4615–4628 (2020)

[SCL+18] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W.Teh, R. Pascanu, R. Hadsell, Progress & Compress: A Scalable Framework for Continual Learning, in *International Conference on Machine Learning* (July, 2018), pp. 4528–4537. PMLR

[SKGK20] A. Singh, A. Kamireddypalli, V. Gandhi, K.M. Krishna, LiDAR Guided Small Obstacle Segmentation, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), pp. 8513–8520. IEEE

[SLKK17] H. Shin, J.K. Lee, J. Kim, J. Kim, Continual learning with deep generative replay, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Long Beach, CA, USA, December 2017), pp. 2990–2999

[SY14] M. Sakurada, T. Yairi, Anomaly detection using autoencoders with nonlinear dimensionality reduction, in *Proceedings of the Workshop on Machine Learning for Sensory Data Analysis (MLSDA)* (Gold Coast, QLD, Australia, December 2014), pp. 4–11

[TTA19] O. Tasar, Y. Tarabalka, P. Alliez, Incremental learning for semantic segmentation of large-scale remote sensing data. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **12**(9), 3524–3537 (2019)

[vdMH08] L. van der Maaten, G. Hinton, Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(86), 2579–2605 (2008)

[vdVT19] G.M. van de Ven, A.S. Tolias, Three Scenarios for Continual Learning (April 2019), pp. 1–18. arXiv:1904.07734

[VGV+21] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, L. Van Gool, Multi-task learning for dense prediction tasks: a survey, in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (January 2021), early access, pp. 1–20

[WCW+18] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, Y. Fu, Incremental Classifier Learning With Generative Adversarial Networks (February 2018), pp. 1–10. arXiv:1802.00853

[WCW+19] Y. Wu, Y.-J. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, June 2019), pp. 374–382

[XZL+20] Y. Xia, Y. Zhang, F. Liu, W. Shen, A. Yuille, Synthesize then compare: detecting failures and anomalies for semantic segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)*, virtual conference (August 2020), pp. 145–161

[YCW+20] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, T. Darrell, BDD100K: a diverse driving dataset for heterogeneous multitask learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* virtual conference (June 2020), pp. 2636–2645

[YHW+19] X. Yao, T. Huang, W. Chenglei, R. Zhang, L. Sun, Adversarial feature alignment: avoid catastrophic forgetting in incremental task lifelong learning. Neural Comput. **31**(11), 2266–2291 (2019)

[YYLH18] J. Yoon, E. Yang, J. Lee, S. Ju Hwang, Lifelong Learning With Dynamically Expandable Networks (June 2018), pp. 1–11. arXiv:1708.01547

[YZZ+19] Y. Yang, D. Wei Zhou, D. Chuan Zhan, H. Xiong, Y. Jiang, Adaptive deep models for incremental learning: considering capacity scalability and sustainability, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (Anchorage, AK, USA, July 2019), pp. 74–82

[ZCCY21] G. Zeng, Y. Chen, B. Cui, S. Yu, Continuous Learning of Context-dependent Processing in Neural Networks (June 2021), pp. 1–18. arXiv:1810.01256

[ZPG17] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, in *Proceedings of the International Conference on Machine Learning (ICML)* (Sydney, NSW, Australia, August 2017), pp. 3987–3995

[ZSR+19] Y. Zhu, K. Sapra, F.A. Reda, K.J. Shih, S. Newsam, A. Tao, B. Catanzaro, Improving semantic segmentation via video propagation and label relaxation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, June 2019), pp. 8856–8865

Evaluating Mixture-of-Experts Architectures for Network Aggregation

Svetlana Pavlitskaya, Christian Hubschneider, and Michael Weber

Abstract The mixture-of-experts (MoE) architecture is an approach to aggregate several expert components via an additional gating module, which learns to predict the most suitable distribution of the expert's outputs for each input. An MoE thus not only relies on redundancy for increased robustness—we also demonstrate how this architecture can provide additional interpretability, while retaining performance similar to a standalone network. As an example, we train expert networks to perform semantic segmentation of the traffic scenes and combine them into an MoE with an additional gating network. Our experiments with two different expert model architectures (`FRRN` and `DeepLabv3+`) reveal that the MoE is able to reach, and for certain data subsets even surpass, the baseline performance and also outperforms a simple aggregation via ensembling. A further advantage of an MoE is the increased interpretability—a comparison of pixel-wise predictions of the whole MoE model and the participating experts' help to identify regions of high uncertainty in an input.

1 Introduction

An intelligent combination of redundant, complementary functional modules is one of the strategies to enhance the accuracy of an overall system as well as its robustness to unseen data. Existing aggregation methods either combine neural network outputs, as is the case with *ensembles*, or include combinations at the structural level, generally referred to as *fusion*, including multi-stream and multi-head neural networks.

Ensembles of identical models are a typical approach to gain superior accuracy by combining several distinct modules. Combination strategies include (un-)weighted

S. Pavlitskaya (✉) · C. Hubschneider · M. Weber
FZI Research Center for Information Technology, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany
e-mail: pavlitskaya@fzi.de

C. Hubschneider
e-mail: hubschneider@fzi.de

M. Weber
e-mail: weber@fzi.de

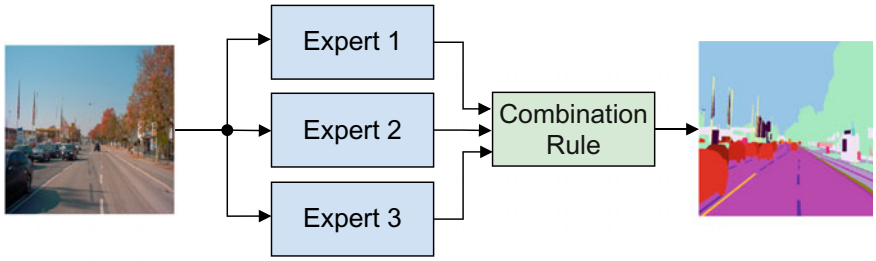
T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_11

averaging, majority voting, stacking, or creating a committee with a combination rule. Ensembles are conceptually simple, deterministic combination rules that are easy to implement and interpret. The approach also easily scales to a multitude of modules. Deep ensembles, i.e., ensembles of deep neural networks, trained with different random initialization parameters, have become one of the de facto standards for achieving better test accuracy and model robustness [LPB17].

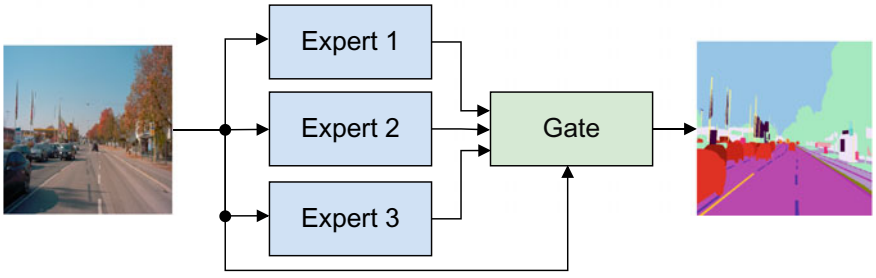
However, since combination rules in ensembling are static, no sophisticated combinations are possible. In the case of deep learning models, each model usually needs to be trained separately. In addition, the overall results can only be calculated after the outputs of all experts are calculated.

Fusion is on the opposite side in the spectrum of model combination approaches. Information from several models is combined either starting from early layers (early fusion), over multiple network layers (slow fusion), or by merging only the last layers (late fusion) [KTS+14]. Regardless of the fusion type, these methods usually involve a complex implementation and allow for no or only little insights into the decision-making process itself.

The *mixture-of-experts* architecture, first proposed by Jacobs et al. [JJNH91], takes a middle path and combines the simplicity and interpretability of the result with the possibility to form sophisticated network architectures. Similar to an ensemble, a mixture of experts combines several separate modules named experts. However, an



(a) Aggregation via ensembling



(b) Aggregation via a mixture of experts

Fig. 1 In an ensemble (**a**), the combination rule is deterministic, so that the distribution of expert weights is fixed for all inputs. A mixture of experts (MoE) (**b**) contains an additional gate module, which predicts the distribution of the expert weights for each input

additional, trainable gating component allows to perform weighting of the expert outputs on a per-input basis (see Fig. 1).

Here, we want to introduce the general concept more thoroughly and extend the evaluation from our previous work [PHW+20] to demonstrate the abilities and possibilities of MoE architectures to perform the task at hand (semantic segmentation of traffic scenes) and to provide post-hoc explainability via a comparison of internal decisions.

The contributions on top of our previous work [PHW+20] are as follows: The experiments are extended to include and evaluate a second expert architecture (`DeepLabv3+`). We also provide new results for the complete label set of the A2D2 dataset. Moreover, a deeper analysis of the impact of the chosen feature extraction layer for the `FRRN` architecture is performed, whereas in previous work only two last pooling layers were considered. Then, expert weights as predicted by different gate architectures and the benefit of an additional convolutional layer is analyzed and discussed. A more advanced MoE architecture is considered and evaluated, in which the encoder layers of the experts are shared. Finally, an additional comparison of the proposed MoE architecture to an ensemble of experts is made.

2 Related Works

Mixtures of experts have primarily been applied for the divide-and-conquer tasks, where the input space is split into disjoint subsets, so that each expert is specialized on a separate subtask and a gating component learns a distribution of these experts.

One of the application areas of MoEs studied in current publications is the fine-grained image classification, where the task is to discriminate between classes in a sub-category of objects, such as the bird classification task [GBM+16]. Ahmed et al. [ABT16] propose a tree-like structure of experts, where the first expert (a generalist) learns to discriminate between coarse groups of classes, whereas further experts (specialists) learn to categorize within a specific group of objects. This way, at inference time, the generalist model acts as a gate by selecting the correct specialist for each input.

The mixture-of-experts approach has also been extensively applied to tackle multi-sensor fusion. To achieve this, each expert is trained on a certain sensor modality and a gating component combines outputs from experts for different input types. Mees et al. [MEB16] apply a mixture of three experts for different modalities (RGB, depth, and optical flow input) to the task of object recognition in mixed indoor and outdoor environments. An analysis of the weights predicted by a gate shows that the MoE adapts to the changing environment by choosing the most appropriate combination of expert weights—e.g., a depth expert gets a higher weight for darker image frames. A similar setup is used in [VDB16], where it is applied to the task of semantic segmentation. Similar to the previous setting, the MoE demonstrates the ability to compensate for a failed sensor via an appropriate weight combination.

Furthermore, in [FC19] and [FC20] the MoE approach is used to handle the multi-modal end-to-end learning for autonomous driving.

A stacked MoE model, consisting of two MoEs, each containing its own experts and corresponding gates, is proposed in [ERS14]. Although evaluated only on MNIST, this work was an important step toward studying large models, which apply conditional execution of a certain part of the whole architecture for each image. The idea was further developed in an influential work on sparse MoEs for language modeling and machine translation [SMM+17], where a generic MoE layer is proposed. Similar to a general MoE, the embedded MoE layer contains several expert networks as well as a gating component, but it forms only a part of the overall architecture trained in an end-to-end manner. During the inference, the experts with non-zero weights are activated in each MoE layer, thus enabling conditional computation on a per-input basis.

The mixture-of-experts approach is also tightly connected to the *selective execution* or *conditional computation* paradigm. Normally, a straightforward way to boost the performance of a deep learning model, given a sufficiently large dataset, is to increase its capacity, i.e., the number of parameters. This, however, leads to the quadratic increase in the computational costs for training [SMM+17]. One of the approaches to overcome this is to execute only a certain part of the network in an input-dependent manner. In this area, primarily approaches such as dynamic deep networks [LD18] have been used. A large model with hundreds of MoE layers with dynamic routing is explored in [WYD+19]. Recent work [LLX+21] continues this line of research and applies conditional computation and automatic sharding to a large, sparsely gated MoE combined with transformers for neural machine translation.

In contrast to existing work, we treat the MoE not only as a method to achieve higher accuracy but also as a post-hoc interpretability approach. The described architecture is closer to earlier MoE methods [MEB16] and less complex than the stacked and embedded MoE.

3 Methods

In this section, we introduce the mixture-of-experts approach formally and show how discrepancy masks may arise from a comparison of the intermediate predictions within an MoE and how they can serve as a means to allocate regions of high uncertainty in an input.

3.1 MoE Architecture

The mixture-of-experts architecture presented here originates from our previous work [PHW+20] and is partially inspired by Valada et al. [VDB16]. In this set-

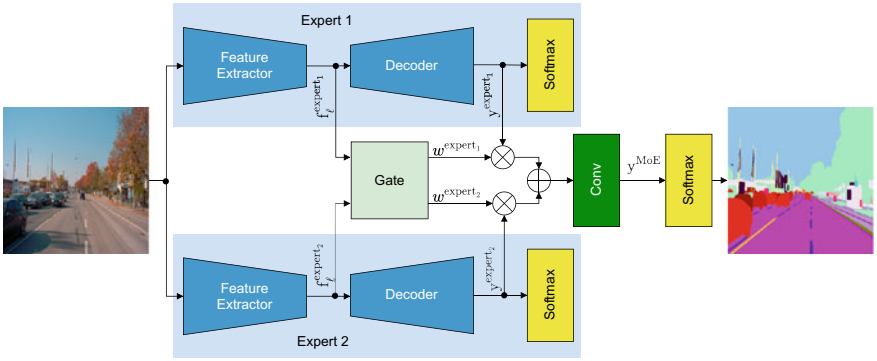


Fig. 2 Architecture of an MoE with separate feature extractors. The gate module receives a concatenation of feature maps $\mathbf{f}_\ell^{\text{expert}_1}$ and $\mathbf{f}_\ell^{\text{expert}_2}$, extracted from the layer ℓ in an encoder part of each of the experts. The gate then predicts the expert weights w^{expert_1} and w^{expert_2}

ting, the MoE consists of two experts, each trained on a separate subset of data, as well as a small gating network. Deviating from the standard MoE approach, we reuse existing feature maps, extracted from the encoder part of the experts, as inputs to a gate. These feature maps are much smaller and can thus speed up both training and inference (see Fig. 2).

Formally, an MoE $\mathbf{F}^{\text{MoE}}$ consists of n experts $\{\mathbf{F}^{\text{expert}_\nu}\}_{\nu=1\ldots n}$ and a gate $\mathbf{F}^{\text{gate}}$. Assume each expert $\mathbf{F}^{\text{expert}_\nu}$ follows an encoder-decoder architecture. We select a certain feature extraction layer ℓ from the encoder part of each expert. The gate $\mathbf{F}^{\text{gate}}$ receives a concatenation of feature maps $\mathbf{f}_\ell^{\text{expert}_\nu}$ from layer ℓ of each expert and computes expert weights for an input $\mathbf{x} \in \mathcal{I}^{H\times W\times C}$ with height H, width W, number of channels $C = 3$, and $\mathcal{I} = [0, 1]$ via

$$(w^{\text{expert}_1}, \ldots, w^{\text{expert}_n}) = \mathbf{F}^{\text{gate}}\left(\mathbf{f}_\ell^{\text{expert}_\nu}\Big|_{\nu=1,\ldots,n}\right). \tag{1}$$

To calculate the overall MoE prediction, the weighted sum of logits $\mathbf{y}^{\text{expert}_\nu}$ of all experts is computed as

$$\mathbf{y} = \mathbf{F}^{\text{MoE}}(\mathbf{x}) = \sum_{\nu=1}^{n} w^{\text{expert}_\nu} \cdot \mathbf{y}^{\text{expert}_\nu}. \tag{2}$$

The resulting approach can thus be interpreted as an extension of ensembles, where weighting of the outputs of individual models is not predefined, but chosen according to the outputs of a trainable gate.

To further increase the capacity of the model, this architecture can be enhanced in a late fusion manner. This can be achieved by appending an additional convolutional layer which processes the weighted sum of the expert outputs before the final prediction $\mathbf{F}^{\text{MoE}}$ is computed (c.f. Fig. 2).

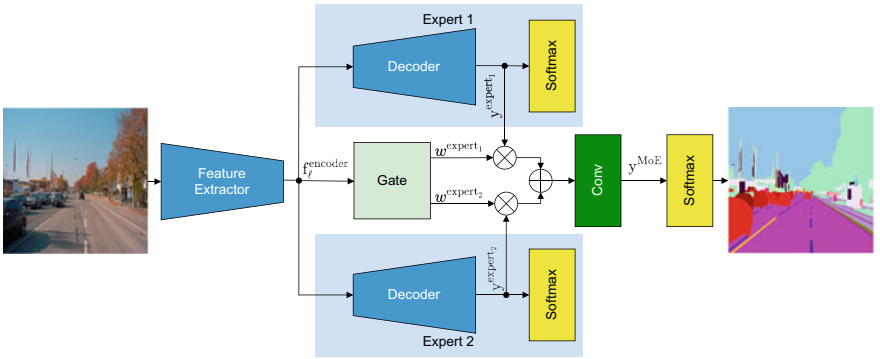


Fig. 3 Architecture of an MoE with a shared feature extractor. In contrast to the architecture in Fig. 2, the gate module receives a single feature map $\mathbf{f}_{\ell}^{\text{encoder}}$ from the shared feature extractor as an input

Since convolutional neural networks tend to learn similar generic features in the lower layers, these layers can be shared across experts to reduce the overall number of parameters (see Fig. 3). In this case, each of the expert decoders and the gate receive the same pre-processed input $\mathbf{f}_{\ell}^{\text{encoder}}$.

3.2 *Disagreements Within an MoE*

The multi-component nature of an MoE allows analyzing the final predictions via comparison of several intermediate predictions, which are either taken into consideration or declined by the overall model. The final decision of an MoE is performed on a per-pixel basis and consists of weighting individual decisions of the participating experts.

By comparison of the predictions of single experts $\mathbf{y}^{\text{expert}_\nu}$ and prediction of the MoE $\mathbf{y}^{\text{MoE}}$, we are able to decide for each pixel whether its classification presents a difficulty to the overall model.

We consider the following three cases that could arise. The discrepancy mask $\mathbf{y} = (y_{h,w}) \in \mathcal{B}^{H \times W}$ has the same height H and width W as the MoE prediction $\mathbf{y}^{\text{MoE}}$, with h and w being row and column index, and $\mathcal{B} = \{0, 1\}$. For each case, a corresponding discrepancy mask for an input image is calculated as follows (note that the symbol $\bigwedge$ represents the logical conjunction operator):

- Perfect case: Here, the prediction by all experts and the MoE is the same. The elements of the discrepancy mask $\mathbf{y}^{\text{perfect}}$ for the perfect case are computed as follows:

$$y_{h,w}^{\text{perfect}} = 1 \iff \bigwedge_{\nu=1}^{n} \left(\left(\arg\max(\mathbf{y}^{\text{expert}_\nu}) \right)_{h,w} = \left(\arg\max(\mathbf{y}^{\text{MoE}}) \right)_{h,w} \right). \quad (3)$$

- Normal case: Here, the prediction by one of the experts and the MoE is the same. This case can further be split into normal case 1, where the same prediction is output by expert 1 and the MoE, normal case 2 for expert 2 and the MoE, etc. The elements of the discrepancy mask $\mathbf{y}^{\text{normal}_1}$ for the normal case 1 are computed as follows:

$$y_{h,w}^{\text{normal}_1} = 1 \iff \left(\left(\arg\max(\mathbf{y}^{\text{expert}_1}) \right)_{h,w} = \left(\arg\max(\mathbf{y}^{\text{MoE}}) \right)_{h,w} \right) \wedge \bigwedge_{v=2}^{n} \left(\left(\arg\max(\mathbf{y}^{\text{expert}_v}) \right)_{h,w} \neq \left(\arg\max(\mathbf{y}^{\text{MoE}}) \right)_{h,w} \right). \tag{4}$$

- Critical case: The MoE and experts output different predictions. The elements of the discrepancy mask $\mathbf{y}^{\text{critical}}$ for the critical case are computed as follows:

$$y_{h,w}^{\text{critical}} = 1 \iff \bigwedge_{v=1}^{n} \left(\left(\arg\max(\mathbf{y}^{\text{expert}_v}) \right)_{h,w} \neq \left(\arg\max(\mathbf{y}^{\text{MoE}}) \right)_{h,w} \right). \tag{5}$$

The critical case, where the MoE outputs a prediction, which was not foreseen by any of the experts, helps to identify image regions, which are challenging for the current model and could, for example, require more relevant examples in the training data.

4 Experiment Setup

To demonstrate the mixture-of-experts architecture, we chose the application of semantic image segmentation of traffic scenes. The experiments build upon and extend our previous work [PHW+20].

4.1 Datasets and Metrics

Dataset: All experiments were performed using the A2D2 dataset [GKM+20], consisting of RGB images with detailed semantic segmentation masks. In addition to the semantic segmentation labels, we manually labeled front camera images by road type (highway vs. urban vs. ambiguous). The disjoint subsets of highway and urban images were then used to train two expert networks, one for each data subset. Even though the urban images outnumber those depicting highway scenes, the same number of training (6132) and validation (876) samples are used for each expert. Additionally, 1421 test samples of each of the three road classes (including ambiguous) are available. All images are resized to 640×480 pixels for training and inference.

In contrast to previous work [PHW+20], we use the complete label set of the A2D2 dataset here, comprising 38 classes.

Metrics: The distribution of labels in the A2D2 dataset is highly imbalanced. In particular, objects of nine out of 38 classes occur in less than 5% of images. To help alleviate this challenge, in addition to the mean intersection-over-union (mIoU), the frequency-weighted IoU (fwIoU) was used for evaluation purposes. In the case of the mIoU, the class-wise average is taken. To calculate the fwIoU, each class-wise IoU is weighted by class frequency, pre-computed as a ratio of pixels labeled as belonging to a certain class in the dataset.

4.2 Topologies

Expert architectures: To lay the focus on the overall mixture-of-experts architecture, two different expert base architectures are evaluated: the Full-Resolution Residual Network (`FRRN`) [PHML17] and `DeepLabv3+` [CZP+18].

An `FRRN` network is comprised of two interconnected streams: a pooling and a residual one. While the residual stream carries the information at the original image resolution, the pooling stream follows the encoder-decoder approach with a series of pooling and unpooling operations. Furthermore, the whole `FRRN` is constructed as a series of the full-resolution residual units (FRRUs). Each FRRU takes information from both streams as input and also outputs information for both streams, thus having two inputs and two outputs. An FRRU contains two convolutional layers, each followed by a ReLU activation function.

For the following experiments, we use a shallower version of `FRRN`, namely `FRRN-A`. The encoder part of the `FRRN-A` pooling stream consists of ten FRRU blocks, which are distributed into four groups. Each group contains FRRU blocks with the same number of channels in their convolution layers; the group of FRRU blocks is correspondingly denoted by the number of channels (e.g., FRRU_{96} is a group of FRRUs containing convolutions with 96 channels). Each group is then followed by a pooling layer (max-pooling in our case).

The second expert architecture, `DeepLabv3+`, uses a previous network version (`DeepLabv3`) as an encoder and enhances it with a decoder module. This encoder-decoder architecture of `DeepLabv3+` suggests to use the feature maps of the final encoder layer, namely the atrous spatial pyramid pooling (ASPP) layer, as input to the gate. For the experts based on `DeepLabv3+`, we evaluate architectures with the `ResNet-101` backbone [HZRS16], pre-trained on ImageNet [RDS+15].

Gating: The gate is a core MoE module. It predicts expert probabilities, which can then be directly used to weight expert outputs. Figure 4 demonstrates the gate architecture. We consider two possibilities to incorporate a gate into the MoE architecture: a simple and a class-wise gate. A simple gate predicts a distribution of expert weights for the whole input image. The output of a simple gate is thus a list of n scalars, one for each expert. A class-wise gate, originally proposed in [VVDB17], is a list of

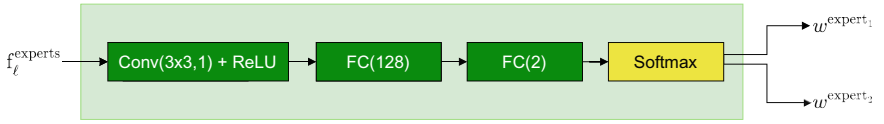


Fig. 4 Gate architecture. Gate input $\mathbf{f}_\ell^{\text{experts}}$ is either the concatenation of feature maps $\mathbf{f}_\ell^{\text{expert}_1}$ and $\mathbf{f}_\ell^{\text{expert}_2}$ (in case of MoE architecture from Fig. 2) or a single feature map $\mathbf{f}_\ell^{\text{encoder}}$ from the shared encoder (in case of the MoE architecture from Fig. 3). Gate outputs w^{expert_1} and w^{expert_2} are the corresponding expert weights

simple gates, one for each label. Each simple gate follows the same architecture as shown in Fig. 4. The output of a class-wise gate is thus a tensor of dimension $n \times m$, where n is the number of experts and m is the number of classes. Via multiplication with the two-dimensional gate outputs, the expert outputs are therefore weighted in a class-wise manner. Overall, a simple gate learns to predict how much each expert can be trusted for a specific input, whereas a class-wise gate learns to decide which expert is most suitable for each label given a specific input.

Moreover, we evaluate the addition of a further convolutional layer $\mathbf{f}_\ell^{\text{conv}}$, followed by a ReLU activation function. $\mathbf{f}_\ell^{\text{conv}}$ is additionally inserted (see Figs. 2 and 3), such that the overall MoE output is then computed as follows:

$$\mathbf{y} = \mathbf{F}^{\text{MoE}}(\mathbf{x}) = \sigma\left(\mathbf{f}_\ell^{\text{conv}}(\mathbf{y}^{\text{expert}_1} \cdot w^{\text{expert}_1} + \mathbf{y}^{\text{expert}_2} \cdot w^{\text{expert}_2})\right). \tag{6}$$

4.3 Training

Pipelines for training and evaluation are implemented in `PyTorch` [PGM+19]. `NVIDIA GeForce GTX 1080 Ti` graphics cards are used for training and inference. The MoE with separate feature extractors (as shown in Fig. 2) is trained in two stages. First, each expert is trained as a separate neural network on the corresponding data subset. Then, the expert weights are frozen and the whole MoE architecture is trained end-to-end. In the case of a shared encoder architecture (as shown in Fig. 3), the expert networks are trained jointly at the first stage, whereas their encoder layers share the parameters. Afterwards, the weights of the shared encoder and of the expert decoders are frozen and the overall MoE is trained end-to-end. Each expert is trained for 50 epochs with a batch size of two. The MoE is trained for 20 epochs with a batch size of six. A smaller batch size is used for larger feature maps. SGD is used as an optimizer for all models. The polynomial learning rate decay with an initial value set to 0.01 is used.

5 Experimental Results and Discussion

In the following sections, we discuss our findings from the experiments regarding architectural choices, analyze the performance of the MoE model, and discuss its potential as an interpretability approach.

5.1 Architectural Choices

Architectural choices reported in this section refer to Tables 1, and 2, 3 and have been obtained on the test dataset. Note that a good generalization on unseen test data can only be ensured if the respective ablation study is repeated on a separate validation dataset.

Expert feature extraction layer: For the `FRRN`-based architecture with separate encoders, which is shown in Fig. 2, we evaluate feature maps, extracted from the last max-pooling layer in each FRRU block of the encoder. We additionally evaluate the usage of input images directly as gate input as in the original MoE approach. Table 1 demonstrates the results for the MoE architecture with a simple gate with an extra convolutional layer. Using raw input data as gate input has led to better results only on the highway data, while achieving the worst results on the urban data parts. Using the pooling layer of the last FRRU block as input, however, achieves the highest mIoU values on the mixed dataset. A possible reason might be that the extracted high-level features from the later layers serve as a better representation of the more complex urban images. The feature resolution shrinks via a series of pooling operations in the pooling stream, therefore the last FRRU blocks additionally lead to shorter training times. Since the best results on the mixed dataset were also achieved by a model with the second FRRU_{384} feature extraction layer, it was selected for all further experiments.

Gate architecture: To determine the best design choices for the gate architecture and whether to add additional convolutional layers after the gate, we train and evaluate

Table 1 `FRRN`-based architecture: mIoU/fwIoU for different feature extraction layers. The results are for the simple gate architecture with the extra convolutional layer

Layer	Feature map size	Highway	Ambiguous	Urban	All
Input	640×480×3	**0.449/0.949**	0.396/0.939	0.395/0.806	0.412/0.890
FRRU_{96}	240×320×16	0.342/0.938	0.419/0.936	0.479/0.861	0.478/0.911
FRRU_{192}	120×160×16	0.342/0.938	0.420/0.938	**0.480/0.861**	0.479/0.911
FRRU_{384}	60×80×16	0.346/0.940	0.420/0.938	**0.480/0.861**	0.480/0.912
2^{nd} FRRU_{384}	30×40×16	0.390/0.947	**0.445/0.940**	0.467/0.855	**0.483/0.913**

Table 2 FRRN-based architecture: mIoU/fwIoU for different gate architectures. The results are for the second $FRRU_{384}$ as a feature extraction layer

Gate	Highway	Ambiguous	Urban	All
Simple	0.349/0.938	0.418/0.937	0.473/0.859	0.475/0.910
Simple with conv.	**0.390/0.947**	**0.445/0.940**	0.467/0.855	**0.483/0.913**
Class-wise	0.373/0.942	0.443/0.939	0.472/0.861	0.475/**0.913**
Class-wise with conv.	0.366/0.939	0.408/0.938	**0.482/0.863**	**0.483**/0.912

Table 3 DeepLabv3+-based architecture: mIoU/fwIoU for different gate architectures

Gate	Highway	Ambiguous	Urban	All
Simple	0.405/**0.956**	0.349/0.942	0.334/**0.854**	0.345/**0.917**
Simple with conv.	**0.415**/0.955	0.339/0.942	**0.342/0.854**	**0.358/0.917**
Class-wise	0.399/0.953	0.360/**0.943**	0.314/0.849	0.333/0.914
Class-wise with conv.	0.404/0.954	**0.369**/0.942	0.333/0.853	0.351/0.916

MoE models with separate encoders (as shown in Fig. 2) for different combinations of these features.

The analysis of the gate's predictions has demonstrated that the simple gate tends to select the correct expert for each data subset. As an example, the average expert weights as predicted by a simple gate for the DeepLabv3+-based architecture are as follows: 0.8 for the highway expert and 0.2 for the urban expert on the highway data subset, 0.04 for the highway expert and 0.96 for the urban expert on the urban data subset, and 0.52 for the highway expert and 0.48 for the urban expert on the ambiguous data.

The class-wise gate, however, predicts almost uniform weights for the majority of classes. Only a small overweight is assigned to the highway expert on the highway data and correspondingly to the urban expert on urban data. Classes for which the urban expert is assigned a significantly higher weight on all inputs are Signal corpus, Sidewalk, Buildings, Curbstone, Nature object, and RD Normal Street. Moreover, we observed that the class-wise gate tends to predict the same weights for all inputs.

Overall, the simple gate demonstrates much higher flexibility to input data, which explains the better results of the corresponding architectures (see Tables 2 and 3). Also, regardless of the gate architecture, an additional convolutional layer also leads to higher mIoU values, because the overall model has a higher capacity. Both FRRN- and DeepLabv3+-based MoEs demonstrate the best performance for the simple gate with an additional convolutional layer after the gate.

5.2 MoE Performance

To compare the performance of the proposed MoE architecture with a baseline and with further aggregation methods, we focus in the following experiments on the MoE with the separate feature encoders (as shown in Fig. 2), which uses the simple gate architecture, an additional convolutional layer and, in case of the `FRRN`-based model, the second FRRU_{384} as its feature extraction layer. Our conclusions on the performance of this model are also valid for almost all further architectural choices, presented in the previous subsection (cf. Tables 1, 2, and 3). Results discussed in this section refer to Tables 4 and 5 and have been obtained on the test dataset.

MoE versus baseline versus experts: To ensure a proper basis for comparison, we train a baseline on a combined dataset of highway and urban data. Its architecture is identical to that of an individual expert. Since the baseline was exposed to twice as much data as each expert, it outperforms the expert models that are trained on their respective data subset only. As expected, each of the experts demonstrates the best results on its corresponding data subset, whereas the urban expert shows a slightly better generalization due to a higher diversity of traffic scenes in its training data. The performance of the `DeepLabv3+`-based MoE with the separate feature encoders (as shown in Fig. 2) surpasses that of the baseline on the mixed dataset, whereas the `FRRN`-based MoE approaches the baseline.

MoE versus ensemble: We also evaluated an ensemble of experts as a concurrent aggregation approach. For this, the class-wise probabilities, predicted by the pre-trained experts, were combined using either by taking a mean or a maximum over the experts. Both combination strategies consistently underperform compared to an

Table 4 `FRRN`-based architecture: mIoU/fwIoU for experts and the MoE. For the MoE, the results are shown for a simple gate with convolutional layer and second FRRU_{384} as its feature extraction layer

Model\Dataset	Highway	Ambiguous	Urban	All
Baseline	0.425/0.946	0.415/**0.940**	0.458/**0.859**	0.475/**0.914**
Highway expert	**0.498**/0.952	0.298/0.919	0.149/0.616	0.227/0.820
Urban expert	0.323/0.934	0.407/0.936	**0.476/0.859**	0.472/0.908
MoE	0.390/0.947	**0.445/0.940**	0.467/0.855	**0.483**/0.913
Ensemble (mean)	0.453/0.949	0.399/0.938	0.382/0.819	0.417/0.900
Ensemble (max)	0.436/0.951	0.410/0.935	0.390/0.823	0.439/0.900
Highway expert (shared)	0.494/**0.954**	0.359/0.939	0.272/0.786	0.227/0.820
Urban expert (shared)	0.380/0.944	0.431/**0.940**	0.466/**0.859**	0.472/0.908
MoE (shared)	0.386/0.952	0.410/0.893	0.438/0.810	0.462/0.873

Table 5 DeepLabv3+ -based architecture: mIoU/fwIoU for experts and the MoE. For the MoE, the results refer to the simple gate with convolutional layer

Model\Dataset	Highway	Ambiguous	Urban	All
Baseline	**0.421/0.956**	**0.364/0.944**	0.302/0.841	0.321/0.913
Highway expert	0.405/0.955	0.252/0.928	0.098/0.607	0.154/0.821
Urban expert	0.334/0.942	0.353/0.941	0.334/**0.854**	0.333/0.912
MoE	0.415/0.955	0.339/0.942	**0.342/0.854**	**0.358/0.917**
Ensemble (mean)	0.372/0.953	0.320/0.942	0.235/0.802	0.266/0.897
Ensemble (max)	0.401/0.954	0.362/0.943	0.291/0.830	0.317/0.908
Highway expert (shared)	0.400/0.955	0.284/0.941	0.173/0.766	0.214/0.884
Urban expert (shared)	0.396/0.952	0.357/**0.944**	0.294/0.835	0.311/0.910
MoE (shared)	0.401/0.952	0.293/0.941	0.252/0.813	0.267/0.911

MoE with separate feature encoders on all data subsets, except for the FRRN-based architecture on highway data. Moreover, of those two approaches, the combination via maximum has led to slightly better results.

MoE with a shared encoder: For the shared encoder architecture as shown in Fig. 3, the differences in performance of experts are less drastic, whereas the MoE demonstrates a slightly inferior mIoU when compared to an MoE with separate encoders. It seems as if the insufficient specialization of the experts negatively affects the performance of the MoE model. Although the shared encoder provides faster inference, a further drawback of this approach is that joint training of both shared experts is much more time-consuming. This might limit the usage of the architecture, especially when a fast replacement of certain experts during inference is considered.

5.3 Disagreement Analysis

We compare pixel-wise predictions of each expert and of the overall MoE architecture. We report percentage of pixels belonging to each disagreement class in Tables 6 and 7, using the test dataset. In both architectures, the experts and the overall MoE output the same predictions (perfect case) for the majority of pixels. Pixels belonging to the normal and critical cases take up to 5% of an image for highway and ambiguous data. For urban data, the MoE tends to rely heavily on the urban expert (up to 27% of pixels), the urban expert is also more accurate on this data according to the reported mIoU and fwIoU.

Table 6 FRRN-based architecture: percentage of pixels, belonging to each disagreement case. The results refer to the architecture with an additional convolutional layer after the gate and second $FRRU_{384}$ as a feature extraction layer

Model	Simple			Class-wise		
	Highway	Ambiguous	Urban	Highway	Ambiguous	Urban
Perfect case	97.00	95.84	70.76	96.93	95.84	70.74
Normal case 1	0.63	0.40	0.55	0.59	0.40	0.64
Normal case 2	2.14	3.51	27.87	2.18	3.51	27.70
Critical case	0.22	0.25	0.82	0.30	0.25	0.92

Table 7 DeepLabv3+ -based architecture: percentage of pixels, belonging to each disagreement case. The results refer to the architecture with an additional convolutional layer after the gate

Gate	Simple			Class-wise		
	Highway	Ambiguous	Urban	Highway	Ambiguous	Urban
Perfect case	97.53	96.67	72.94	97.52	96.54	72.92
Normal case 1	2.02	1.34	0.39	1.62	1.17	1.12
Normal case 2	0.28	1.78	25.98	0.61	1.91	24.94
Critical case	0.17	0.21	0.69	0.25	0.37	1.02

To facilitate visual analysis of the disagreement regions, we show a disagreement mask for each input (see Fig. 5). The perfect case pixels are left transparent, the normal case pixels are colored green, and blue for the highway and urban experts correspondingly. The critical case pixels are highlighted in red. The visual analysis is consistent with the semantics of the objects in the scene. Regions mapped to the normal case are those, which can confidently be segmented by the corresponding expert, because they mostly occur in its training set and not in the training set of a different expert.

The critical cases are small image areas (up to 1% of pixels). Because we consider ambiguous traffic scenes as out-of-distribution in our setting, they provide the most interesting material for the visual analysis of the critical cases (see Fig. 6). The critical case areas are usually the overexposed, blurred, ambiguous, or hardly visible regions of an image. Interestingly, sidewalk pixels are often classified as belonging to the critical case in the ambiguous dataset.

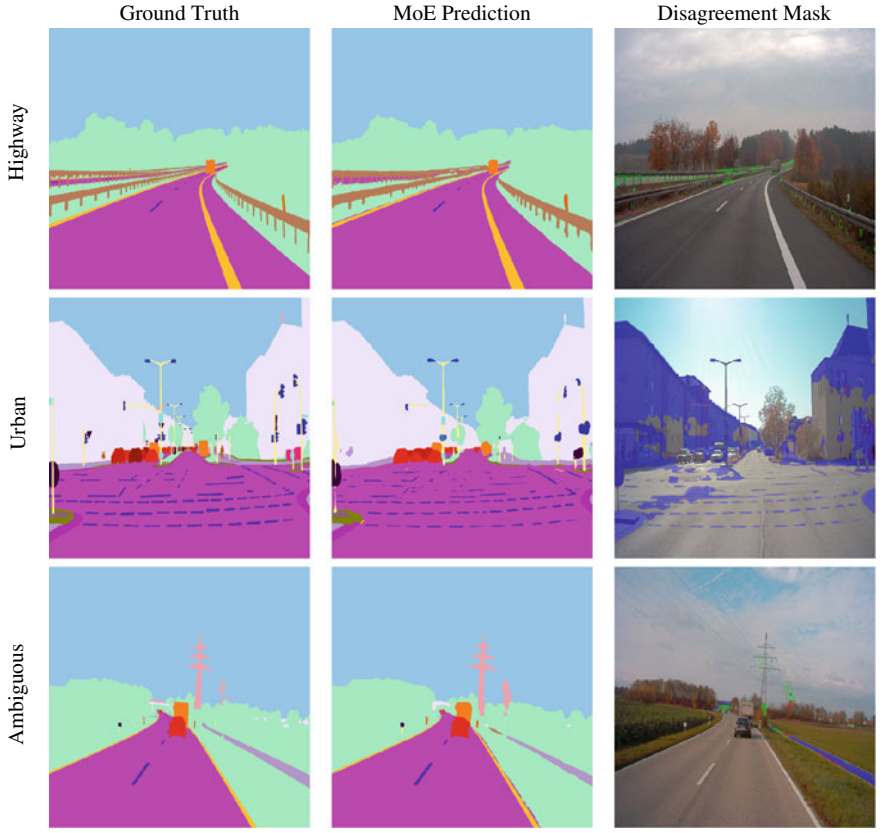


Fig. 5 Disagreement masks and predictions. Perfect case pixels are shown as image pixels, normal case pixels are highlighted green (agreement with the highway expert) or blue (agreement with the urban expert), and critical case pixels are highlighted red. The results are for the `DeepLabv3+` architecture with a simple gate and additional convolutional layer

Conclusions

Mixture of experts (MoE) is a network aggregation approach, which combines simplicity and intrinsic interpretability of ensembling methods with the possibility to construct more flexible models via the application of an additional gate module. In this chapter, we have studied how a mixture of experts can be used to aggregate deep neural networks for semantic segmentation to increase performance and gain additional interpretability.

Our experiments with two different expert architectures demonstrate that MoE is able to reach baseline performance and additionally reveal image regions, for which the model exhibits high uncertainty. In comparison to our previous experiments in [PHW+20], the models were trained on a full A2D2 label set, which led not only

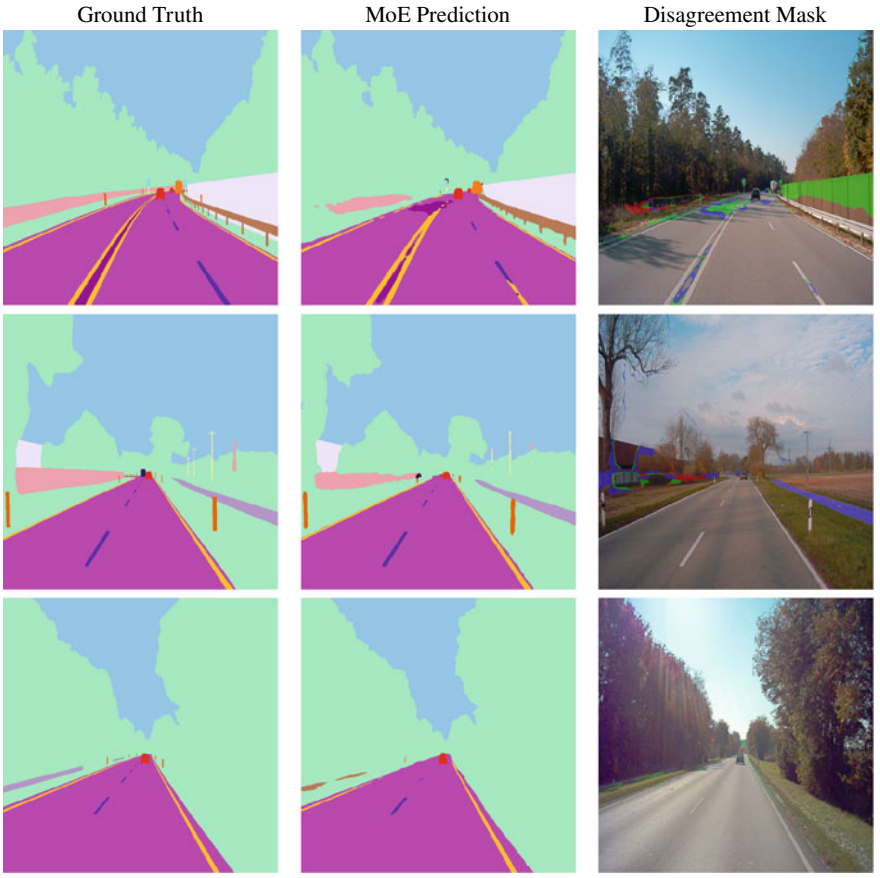


Fig. 6 Examples of disagreement masks with critical case from the ambiguous test set. Perfect case pixels are shown as image pixels, normal case pixels are highlighted green (agreement with the highway expert) or blue (agreement with the urban expert), and critical case pixels are highlighted red. The results are for the `DeepLabv3+` architecture with a simple gate and additional convolutional layer

to decreased performance on rare classes, as expected, but also to the increased occurrence of disagreements between the overall architecture and the experts. Furthermore, we have evaluated the possibility to share the parameters of the feature extraction layers of both experts. This leads to better cross-subset expert performance. However, apparently the experts with a shared encoder are no longer specialized enough which leads to a worse MoE performance when compared to an MoE with standalone experts. Our evaluation has also shown that a mixture of experts, enhanced with a gating network, beats a simple combination of experts via ensembling.

Further research directions might include conditional execution within an MoE model, combination of various modalities as inputs, as well as the perspective to further enhance the interpretability and robustness of the overall model.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)". The authors would like to thank the consortium for their successful cooperation.

References

[ABT16] K. Ahmed, M.H. Baig, L. Torresani, Network of experts for large-scale image categorization, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Amsterdam, The Netherlands, 2016), pp. 516–532

[CZP+18] L-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Munich, Germany, 2018), pp. 833–851

[ERS14] D. Eigen, M. Ranzato, I. Sutskever, Learning factored representations in a deep mixture of experts, in *Proceedings of the International Conference on Learning Representations (ICLR) Workshops* (Banff, AB, Canada, 2014), pp. 1–8

[FC19] S. Fang, A. Choromanska, Reconfigurable network for efficient inferencing in autonomous vehicles, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Montréal, QC, Canada, 2019), pp. 1183–1189 , pp. 1183–1189

[FC20] S. Fang, A. Choromanska, Multi-modal experts network for autonomous driving, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Virtual conference (2020), pp. 6439–6445

[GBM+16] Z. Ge, A. Bewley, C. McCool, P. Corke, B. Upcroft, C. Sanderson, Fine-grained classification via mixture of deep convolutional neural networks, in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)* (Lake Placid, NY, USA, 2016), pp. 1–6

[GKM+20] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A.S. Chung, L. Hauswald, V.H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, P. Schuberth, A2D2: Audi Autonomous Driving Dataset (2020), pp. 1–10. arXiv:2004.06320

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 770–778

[JJNH91] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts. Neural Comput. **3**(1), 79–87 (1991)

[KTS+14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L.F. Fei, Large-scale video classification with convolutional neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Columbus, OH, USA, 2014), pp. 1725–1732

[LD18] L. Liu, J. Deng, Dynamic deep neural networks: optimizing accuracy-efficiency trade-offs by selective execution, in *Proceedings of the AAAI Conference on Artificial Intelligence* (New Orleans, LA, USA, 2018), pp. 3675–3682

[LLX+21] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, Z. Chen, GShard: scaling giant models with conditional computation and automatic sharding, in *Proceedings of the International Conference on Learning Representations (ICLR)*, virtual conference (2021), pp 1–35

[LPB17] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Long Beach, CA, USA, 2017), pp. 6402–6413

[MEB16] O. Mees, A. Eitel, W. Burgard, Choosing smartly: adaptive multimodal fusion for object detection in changing environments, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, Daejeon, Korea, 2016), pp. 151–156

[PGM+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, et al. PyTorch: an imperative style, high-performance deep learning library, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Vancouver, BC, Canada, 2019), pp. 8024–8035

[PHML17] T. Pohlen, A. Hermans, M. Mathias, B. Leibe, Full-resolution residual networks for semantic segmentation in street scenes, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI, USA, 2017), pp. 4151–4160

[PHW+20] S. Pavlitskaya, C. Hubschneider, M. Weber, R. Moritz, F. Hüger, P. Schlicht, J. Marius Zollner, Using mixture of expert models to gain insights into semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2020), pp. 1399–1406

[RDS+15] Olga Russakovsky, Jia Deng, Su. Hao, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li. Fei-Fei, ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**(3), 211–252 (2015)

[SMM+17] N. Shazeer, A.Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean, Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer (2017), pp. 1–19. arXiv:1701.06538

[VDB16] A. Valada, A. Dhall, W. Burgard, Convoluted mixture of deep experts for robust semantic segmentation, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshops* (Daejeon, Korea, 2016), pp. 1–2

[VVDB17] A. Valada, J. Vertens, A. Dhall, W. Burgard, AdapNet: adaptive semantic segmentation in adverse environmental conditions, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Singapore, Singapore, 2017), pp. 4644–4651

[WYD+19] X. Wang, F. Yu, L. Dunlap, Y-A. Ma, R. Wang, A. Mirhoseini, T. Darrell, J.E. Gonzalez, Deep mixture of experts via shallow embedding, in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)* (Tel Aviv, Isreal, 2019), pp. 552–562

Safety Assurance of Machine Learning for Perception Functions

Simon Burton, Christian Hellert, Fabian Hüger, Michael Mock, and Andreas Rohatschek

Abstract The latest generation of safety standards applicable to automated driving systems require both qualitative and quantitative safety acceptance criteria to be defined and argued. At the same time, the use of machine learning (ML) functions is increasingly seen as a prerequisite to achieving the necessary levels of perception performance in the complex operating environments of these functions. This inevitably leads to the question of which supporting evidence must be presented to demonstrate the safety of ML-based automated driving systems. This chapter discusses the challenge of deriving suitable acceptance criteria for the ML function and describes how such evidence can be structured in order to support a convincing safety assurance case for the system. In particular, we show how a combination of methods can be used to estimate the overall machine learning performance, as well as to evaluate and reduce the impact of ML-specific insufficiencies, both during design and operation.

S. Burton
Fraunhofer Institute for Cognitive Systems IKS, Hansastraße 32, 80686 Munich, Germany
e-mail: simon.burton@iks.fraunhofer.de

C. Hellert
Continental AG, Bessie-Coleman-Str. 7, 60549 Frankfurt am Main, Germany
e-mail: christian.hellert@continental.com

F. Hüger
Volkswagen AG, Berliner Ring 2, 38440 Wolfsburg, Germany
e-mail: fabian.hueger@volkswagen.de

M. Mock
Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, Schloss Birlinghoven 1, 53757 Sankt Augustin, Germany
e-mail: michael.mock@iais.fraunhofer.de

A. Rohatschek (✉)
Robert Bosch GmbH, Robert-Bosch-Campus 1, 71272 Renningen, Germany
e-mail: andreas-juergen.rohatschek@de.bosch.com

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_12

1 Introduction

The objective of systems safety engineering is to avoid unreasonable risk of hazards that could lead to harm to the users of the system or its environment. The definition of an unreasonable level of risk must be determined for the specific system context in accordance with societal moral concepts and legal considerations regarding product liability. This requires a careful evaluation of the causes and impact of system failures, an evaluation of the probability of their occurrence, and the definition of strategies to eliminate or reduce the residual risk associated with such failures. Current automotive safety standards, such as ISO 26262 [ISO18] and ISO/DIS 21448 [ISO21], require a safety case to be developed for safety-related functions that forms a structured argument supported by systematically collected evidence that the appropriate level of residual risk has been achieved. The term "evidence" refers to work products created during the development, test, and operation of the system that support the claim that the system meets its safety goals.

In comparison to previous generations of vehicle systems, automated driving systems (ADS) that make use of machine learning (ML) introduce significant new challenges to the safety assurance process. Deep neural networks (DNNs), in particular, are seen as an enabling technology for ADS perception functions due to their ability to distinguish features within complex, unstructured data. This allows for the development of functions, such as pedestrian detection within crowded, urban environments that previously could not be specified and implemented based on algorithmic definitions. Paradoxically, this leads to one of the main challenges associated with the use of machine learning in safety-critical systems. The *semantic gap* [BHL+20] describes the challenge of deriving complete and consistent technical (safety) requirements on the system that fulfil potentially only implicitly understood, societal and legal expectations. The semantic gap is exacerbated in ML-based ADS due to the unpredictability and complexity of the operational domain and the reliance on properties of the training data rather than detailed specifications to derive an adequate approximation of the target function. The semantic gap can in turn lead to an unclear definition of the moral responsibility and legal liability for the system's actions as well as an incomplete assurance argument that the (potentially incompletely defined) safety goals for the system are met.

Furthermore, deep learning approaches have specific properties that limit the effectiveness of established safety measures for software. These include the inherent uncertainty in the outputs of the ML function, the opaque manner in which features are learnt by the function which is often not understandable by humans and the difficulty of extrapolating from test results due to non-linear behaviour of the function and sensitivity to small changes in the input domain. Previous work related to the safety assurance of machine learning has mainly focused on the structure of the assurance case and associated processes with respect to existing safety standards [BGH17, SQC17, GHP+20, ACP21, BKS+21]. Other work has focused on the effectiveness of specific metrics and measures on providing meaningful statements related to safety properties of the ML function [CNH+18, HSRW20, SKR+21, CKL21]. This chapter

complements this work by examining in more detail the role of combining various types of evidence to form a convincing argument that quantitative acceptance criteria as required by standards such as ISO 21448 are met.

This chapter discusses the challenge of deriving suitable acceptance criteria for ML functions and describes how supporting evidence can be combined to provide a convincing safety assurance case for the system. The aggregation of evidence can be structured as an overall evidence-based safety argumentation as depicted in [MSB+21]. In the following section, we describe the challenge of deriving a set of technical safety requirements and associated acceptance criteria on ML functions. In Sect. 3, we provide a categorisation of safety evidence for machine learning and argue that an assurance case requires an appropriate combination of complementary evidence. Section 4 describes a collaborative approach between domain and safety experts for collecting and evaluating the safety evidence. Sections 5.1–5.4 provide specific examples of each category of evidence and describe the conditions under which they can provide a meaningful contribution to the assurance case. Section 6 then demonstrates how the evidence can be combined into an overall assurance case structure. The chapter concludes with a summary of open research questions and an outlook for future work.

2 Deriving Acceptance Criteria for ML Functions

This section presents approaches to risk evaluation within the safety standards and discusses how safety acceptance criteria for ML functions can be derived in line with these approaches.

2.1 Definition of Risk According to Current Safety Standards

Current standards related to the safety of vehicle systems provide little specific guidance related to the use of ML. Therefore, the approach described in this chapter will be based upon an interpretation and transfer of the principles of ISO 26262, ISO/DIS 21448, and ISO/TR 4804 to the specific task of safety assurance for machine learning-based systems. ISO 26262 defines functional safety as the absence of unreasonable risk due to hazards caused by malfunctioning behaviour of electrical/electronic systems. Malfunctioning behaviour is typically interpreted as either random hardware faults or systematic errors in the system, hardware, or software design. ISO 26262 applies a predominantly *qualitative* approach to arguing the safety of software. Safety goals for a system are derived according to a hazard and risk analysis that evaluates the risk associated with each hazard according to qualitative criteria with various categories for the risk parameters: severity, exposure, and controllability. The standard provides a method for combining these parameters to derive an overall "Automotive Safety Integrity Level" (ASIL) within a range of A to D in order of increasing

risk. Functions with no safety impact are assigned the level "QM" (only standard quality management approaches are necessary). An ASIL is allocated to each safety goal derived from the hazards and to the functional and technical safety requirements derived from these until eventually software-level requirements, including associated ASILs, are determined. The standard then provides guidance on which measures to apply, depending on ASIL, to achieve a tolerable level of residual risk. By doing so, the standard avoids the need to define specific failure rates to be achieved by software but instead defines desirable properties (such as freedom from run-time errors caused by pointer mismatches, division-by-zero, etc.) and methods suited to ensuring or evaluating these properties (e.g. static analysis).

ISO/DIS 21448 addresses safety in terms of the absence of unreasonable risk due to functional insufficiencies of the system or by reasonably foreseeable misuse. In the context of ADS, this translates to a possible inability of a function to correctly comprehend the situation and operate safely, e.g. due to a lack of robustness regarding input variations or diverse environmental conditions. The risk model described by ISO/DIS 21448 can be summarised as identifying as many *known, unsafe* scenarios (where performance insufficiencies could lead to hazards) as possible so that mitigating measures can be applied, thus transforming them to *known, safe* scenarios. In addition, the number of residual *unknown, unsafe* scenarios should be minimised by increasing the level of understanding of properties of the environment that could trigger performance deficiencies. The definition of safety of the intended functionality (SOTIF), therefore, seems well suited for arguing the performance of the trained ML function over all possible scenarios within the operating domain. ISO/DIS 21448 follows a similar process to identify hazards, associated safety goals, and requirements as ISO 26262. However, instead of using risk categories according to ASILs, the standard requires the definition of *quantitative* acceptance criteria (also known as validation targets) for each hazard, which in turn can be allocated to subsystems such as perception functions. However, these acceptance criteria are not described in more detail and must, therefore, be defined according to the specific system context.

ISO/TR 4804 contains a set of guidelines for achieving the safety of automated driving systems with a focus on the Society of Automotive Engineers (SAE) levels 3–5 [SAE18]. ISO/TR 4804 defines safety acceptance criteria both in terms of a positive risk balance (the system shall be demonstrably safer than an average human driver) as well as the avoidance of unreasonable risk. In doing so, the standard recommends a combination of both qualitative and quantitative arguments. A similar philosophy is followed within this chapter when identifying and combining evidence for the safety of a machine learning function.

2.2 *Definition of Safety Acceptance Criteria for the ML Function*

In order to address the semantic gap described above, a systematic method of deriving a set of technical safety requirements on an ML function is needed. This should include an iterative and data-driven approach to analyse the societal and legal expectations, operating domain, and required performance on the ML function within the technical system context (see Fig. 1). In this chapter, we focus on steps 4 and 5, in particular, the systematic collection of evidence regarding the performance of an ML function to support a system-level assurance case.

In [BGH17], the authors recommend a contract-based design approach to derive specific safety requirements for the machine learning function. These requirements are expressed in the form of a set of safety guarantees that the function must fulfil and a set of assumptions which can be made in the system's environmental and technical context. This allows for a compositional approach to reasoning about the safety of the system where the guarantees of one function can be used to justify the assumptions on the inputs of another. This requires a suitable definition of safety at the level of abstraction of the ML function. For example, a superficially defined requirement, such as "detect a pedestrian on the road", would need to be refined to define which characteristics constitute a pedestrian and from which distance and level of occlusion pedestrians should be detected. This process of requirements elicitation should include the consideration of a number of stakeholder perspectives and sources of data. This could include current accident statistics and the consideration of ethical guidelines for AI as proposed by the European Commission [Ind19]. It is to be expected that for any particular automated driving function, a number of such contracts would be derived to define different properties of the ML function related to various safety goals, where each contract may also be associated with different quantitative acceptance criteria and sets of scenarios in the operating domain.

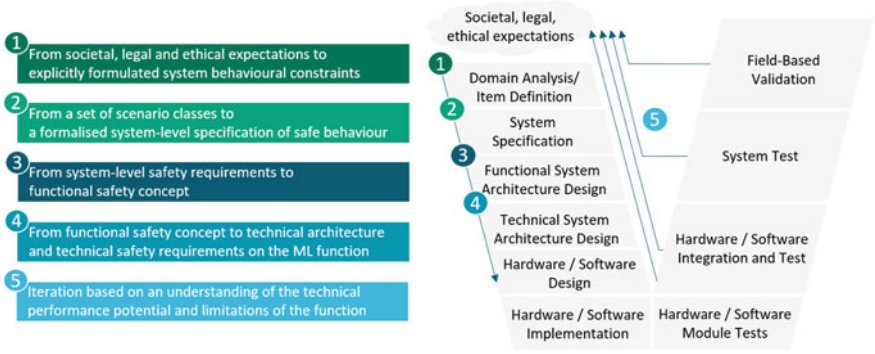


Fig. 1 Iterative steps during the systems engineering process to bridge the semantic gap associated with the definition of technical safety requirements on ML functions

Table 1 Example safety contract for a pedestrian detection

Input domain	The set of all possible situations within an urban environment in which pedestrians could be within the range of the vehicle's sensors
Assumptions	Safety-relevant pedestrians have a size 50 cm $\leq size \leq$ 250 cm and are within a range of 3 m $\leq range \leq$ 50 m
Guarantees	Bounding box accuracy per frame in the sequence shall be within 20 cm of the ground truth, and the pedestrian must be detected whenever the occlusion through other objects is <50%. The confidence score for successful classification per frame shall be ≥70%

Table 1 contains an example of a (not necessarily complete) safety contract for a bounding-box pedestrian detection model. This contract is defined within the scope of a system architecture where the quality of the camera signal acting as input to the function is either known or can be determined during development and where a sensor fusion and monitoring component receives the result of the ML-based detection algorithm and performs time-series plausibility checks on the results and compares with other sensor modalities such as RaDAR and LiDAR.

A *qualitative* approach to defining the assurance targets for the ML function could involve determining a suitable combination of development and test methods to be applied that are assumed to lead to a correct implementation of the function. However, due to typical failure modes and performance limitations of ML, an absolute level of correctness in the function is infeasible. Instead, *quantitative* assurance targets, in line with the requirements of ISO/DIS 21448, are required that would define an acceptable limit to the *probability* that guarantees cannot be met. For example, an acceptance criterion for pedestrian detection based on the remaining accuracy rate (RAR) metric [HSRW20] could be formulated as *"An RAR of 95% is achieved and residual errors are distributed equally across all scenarios"*, leading to the probability of a single pedestrian being undetected by both the ML function and the sensor fusion/monitor component being sufficiently low. The remaining accuracy rate is defined in [HSRW20] as the proportion of inputs where a confidence score above a certain threshold (e.g. 70%) leads to a true positive detection. Thus, the definition of the parameters for the quantitative acceptance targets must be defined based on a detailed understanding of the performance limits of both the ML function and the capabilities of the sensor fusion/monitoring component.

3 Understanding the Contribution of Safety Evidence

3.1 A Causal Model of Machine Learning Insufficiencies

Ideally, to demonstrate that the ML function meets its performance requirements, the probability of failure will be directly measured, for example, based on a sufficiently

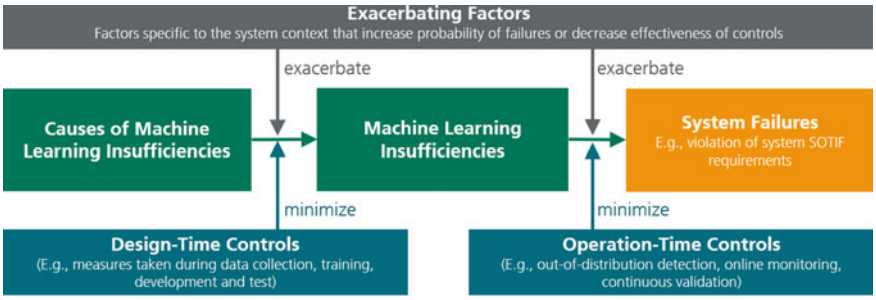


Fig. 2 Causal model of ML-related SOTIF risks demonstrating the relationships between system failures, machine learning insufficiencies, their causes, and exacerbating factors as well as the role of control measures to minimise the probability of hazardous system failures

large number of representative tests. However, this approach is feasible only for trivial examples. Therefore, multiple sources of evidence are required to reason about the presence of performance limitations and the ability of the system to detect and control such failures during operation.

The dependability model of Avizienis et al. [ALRL04] forms the foundation of many safety analysis techniques. In this model, the risk of a system violating its objectives is determined by analysing the propagation path from causes of individual *faults* in the system that could lead to an *erroneous* system state which in turn leads to a *failure* to maintain the system's objectives. Risk can thus be controlled by either eliminating the causes of faults or by preventing them from leading to potentially hazardous erroneous states of the system.

Figure 2 summarises how this approach to causal analysis can be applied to the problem of determining the safety-related performance of an ML function and is inspired by the safer complex systems framework presented in [BMGW21]. A system failure can be defined as a condition where the safety contract as defined in Table 1 is not met. An erroneous state leading to such a failure would relate to the presence of insufficiencies in the machine learning function which, in turn, could be caused by either faults in its execution or limitations in the training data. Examples for the latter are scalable oversight [AOS+16], uncertainty in the input domain, such as distributional shift [AOS+16], or the inherent inability of the ML approach to accurately represent the target function. Measures to improve the safety of the function can be categorised into those applied during design time to reduce the probability of insufficiencies in the trained model and those applied during operation in order to reduce the impact of residual insufficiencies. Both types of controls may be undermined by exacerbating factors specific to the system context, for example, the difficulty in collecting balanced and sufficiently complete training data due to the scarcity of critical scenarios or the difficulty of developing effective monitoring approaches due to the need for safe-operational fallback concepts. This model of causality for ML-related safety of the intended functionality (SOTIF) risks is

now used to determine the following contributions of safety evidence that will be illustrated in the next subsection.

3.2 Categories of Safety Evidence

Based on the causal model of SOTIF-related risk introduced above, the following four categories of evidence can now be defined. This category of evidence corresponds to development work products and results of verification and validation activities that support the argument of an acceptably low level of residual risk associated with ML insufficiencies:

(1) Confirmation of residual failure rates: This category of evidence provides a direct estimate of the performance of the machine-learning component, e.g. in terms of false negative rates, or the intersection-over-union (IoU) related to bounding box accuracy. However, due to non-linear behaviour in the function and the limited number of samples that can be realistically tested, such evidence is unlikely to provide a statistically relevant estimation. This category of evidence must, therefore, be used in combination with other measures that increase confidence in the ability to extrapolate the results to the entire input space that fulfils the contract's assumptions.

(2) Evaluation of insufficiencies: This category of evidence is used to directly indicate the presence or absence of specific ML insufficiencies that could not only lead to a violation of the safety contract, but could also undermine confidence in the category 1 evidence. The properties that would be evaluated by this category of evidence would include prediction uncertainty, generalisation, brittleness, fairness, and explainability.

(3) Evaluation of the effectiveness of design-time controls: This category of evidence is used to argue the effectiveness of design-time measures to increase the performance of the function or to reduce the presence of insufficiencies. In many cases, a direct correlation between the design-time measures and the performance of the function may not be measured, leading to qualitative arguments for the effectiveness of the measures. However, metrics can be used to measure the level of rigour by which the measures have been applied. These could include properties of training data and test data, or measures to increase the explainability of the trained model.

(4) Evaluation of the effectiveness of operation-time controls: This category of evidence is used to demonstrate the effectiveness of operation-time measures to either increase the performance of the function or to reduce the impact of insufficiencies. Examples of such could be a measurement of the effectiveness of out-of-distribution detection to identify inputs that are outside of the training distribution (and beyond the assumptions of the safety contract) such that they can be discarded.

In the following sections, we illustrate each category with specific examples and evaluate the conditions under which the evidence can provide a significant contribution to the assurance case.

4 Evidence Workstreams—Empowering Experts from Safety Engineering and ML to Produce Measures and Evidence

In order to identify, develop, and evaluate effective evidence related to the various categories defined above, we propose to follow the procedure of so-called evidence *workstreams*: experts from machine learning, safety, testing, and data working together according to the procedure depicted in Fig. 3. The main purpose of the process is to help demonstrate the effectiveness of the methods, metrics, and measures to be applied more generally. Related processes are also found in current literature. Picardi et al. [PPH+20] propose an engineering process to generate evidence at each stage of the ML life cycle. Apart from the process in general, different patterns are described that can be instantiated during the ML life cycle. Furthermore, a three-layer process model is described by McDermid and Jia [MJ21], featuring a collaborative model to bring together the expertise from the ML and safety domain to generate evidence for the assurance case. In this work, there are some case studies, but detailed steps for the process are missing.

The evidence workstreams proposed in this chapter enable a joint cooperation of different competencies and bring together new innovative methods and structured approaches rather than having separate work on each topic. An important key to the successful creation of an assurance case is the continuous interaction of the different contributors. Before explaining the procedure, we, therefore, define four roles for the contributors:

- The *method developer* is responsible to implement measures that mitigate specific insufficiencies of an ML component. In addition, this role specifies data

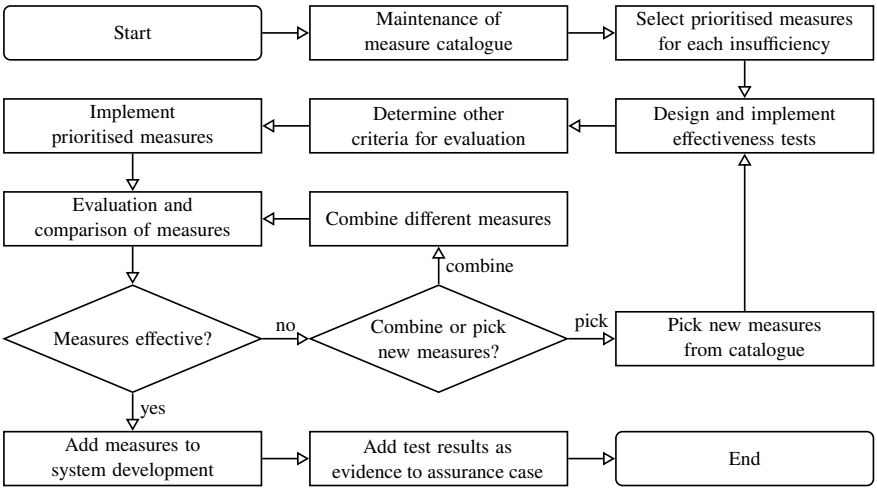


Fig. 3 Flowchart of the evidence workstream process

requirements for the implemented measures and defines metrics for evaluating the effectiveness of the measure.
- Apart from the method developer specifying data requirements, a *data engineer* is responsible for creating and managing the dataset throughout the development process as well as performing data analyses to estimate the data coverage.
- In order to test the effectiveness of the measures, the *test engineer* develops test specifications including required test data and metrics. These tests should then be performed by the method developer to demonstrate the effectiveness of the method.
- Finally, the *safety engineer* ensures that the developed measures and respective test results are valid and generate the evidence for the assurance case.

The process starts by creating and maintaining a catalogue of potential design-time and run-time measures to mitigate identified insufficiencies, which can be known a priori (see Sect. 5.2) or identified during the development phase. Here, *method developers* and *safety engineers* are involved in the maintenance and also perform a prioritisation of measures for each insufficiency. After the selection of measures, the *test engineer* designs and implements effectiveness tests. Furthermore, safety-aware metrics for the test are defined by the *method developer* and *safety engineer*, in addition to other criteria for the evaluation. Importantly, the *data engineer* needs to evaluate the database in order to allow for a statistical significance analysis for the designed tests. Afterwards, the prioritised measures are implemented following an evaluation and comparison according to the specified tests. Once the measures show sufficient effectiveness, they can be added to the system development and the *safety engineer* can use the test results as evidence for the assurance case. If a measure was not effective, then we propose the two following options:

- Firstly, measures can be combined and optimised to increase the effectiveness. After combination, a re-evaluation is performed.
- Secondly, new measures can be picked from the catalogue and the test design and implementation step is repeated.

Additionally, an iteration of the safety requirement derivation and system architecture design process could lead to recommendations for additional components or adjusted specifications in order to mitigate remaining insufficiencies.

5 Examples for Evidence

This section discusses examples for the different categories of safety evidence introduced in Sect. 3.2, which can be obtained by the process introduced in Sect. 4.

5.1 Evidence from Confirmation of Residual Failure Rates

The quantification of model performance or residual failure rates of a machine learning (ML) component is a crucial step in any ML-based system, especially for safety-critical applications. Typically, the performance of ML components is measured by metrics that calculate a mean performance over a specific test dataset. In [PNdS20] and [PPD+21], commonly used metrics for object detection are compared, including mean average precision (mAP) and mean intersection-over-union (mIoU). However, for safety-critical systems, only evaluating the mean performance is not sufficient. On the one hand, there are unknown risks introduced by residual failure rates, on the other hand, failures are not weighted by their relevance. To counteract this, for example, in [HSRW20], safety-aware metrics are proposed in the literature to incorporate uncertainty to evaluate the remaining accuracy rate and remaining error rate. In [CKL21], safety-aware metrics for semantic segmentation are proposed, putting emphasis on perturbation and relevance. Volk et al. [VGvBB20] propose a safety metric that incorporates object velocity, orientation, distance, size, and a damage potential. Furthermore, ISO/TR 4804 [ISO20] proposes a generic concept, whereby safety-aware metrics need to be evaluated for perception components realised with deep neural networks. In addition, it should be noted that the validity of metrics depends strongly on the utilised dataset. In general, the creation of safety-aware metrics is still an open field in the domain of automated driving and for perception components in particular. Based on the references and insights above, the following considerations should be taken into account to use performance metrics as evidence:

- Mean performance metrics should be evaluated within a specified input space.
- Statistical significance of the metrics value should be evaluated and shown by measuring dataset coverage.
- The failure rate related to specific classes of errors should be evaluated, including an analysis of their causes.

The evidence for performance or accuracy evaluation could be in the form of a test report, where the specification of the input space and the data coverage evaluation is defined. Apart from detection or classification accuracy, localisation precision is also an important factor and is often dependent on the accuracy. In addition to a summary of the safety-aware metrics, an analysis of various classes of errors, their potential causes and impact within the system should be included in the test report, allowing for a systematic evaluation of the residual risk associated with the function at a system level.

The following example provides an intuition of conditions that have to be met so that a performance metric can be used as evidence convincingly. The basis of nearly every performance metric is the calculation of the confusion matrix, containing the number of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) rates. Here, we consider the case of a 2D bounding box pedestrian detection on images for automated driving. In this case, the number of TNs is not

reasonable since there will be a nearly infinite amount of those bounding boxes. In order to compute the confusion matrix, the intersection-over-union (IoU) metric is typically used to determine if a detection can be considered as a TP, FP, or FN by using a threshold. This threshold, on the one hand, defines the localisation error per object which must not be exceeded for the detection to be counted as TP, and, on the other hand, also influences the mean accuracy on a test dataset. Therefore, in the test report, there should be an argument about why a certain threshold has been chosen. Furthermore, typically FP and FN are weighted equally, but for a safety analysis, FN might be more important and not all FN might be relevant. Hence, the test report should define relevant FN or FP based on the system architecture and assigned component requirements by introducing specific weights. Of course, performance metrics alone cannot give enough evidence to argue the safety of a perception system for automated driving. In addition, the performance needs to be evaluated in case of rarely occurring situations, and also the robustness against perturbations needs to be considered.

5.2 Evidence from Evaluation of Insufficiencies

An essential feature in the development of deep neural networks during training lies in the purely data-driven parameter fitting process without expert intervention: The deviation of the output (for a given parameterisation) of a neural network from a ground truth is measured. The loss function is chosen in such a way that the parameters depend on it in a differentiable way. As part of the gradient descent algorithm, the parameters of the network are adjusted in each training step depending on the derivative of the deviation (backpropagation). These training steps are repeated until some stopping criterion is satisfied. In this procedure, the model parameters are determined without an expert assessment or semantically motivated modelling. This has significant consequences for the properties of the neural network:

- Deep neural networks (DNNs) are largely opaque for humans and their calculations cannot be interpreted. This represents a massive limitation for systematic testing or formal verification.
- Deep neural networks are susceptible to harmful interference: Perturbations could be manually induced changes in the data (adversarial examples) or real-word corruptions (e.g. sensor noise, weather influences, certain colours, or contrasts by sensor degeneration).
- It is unclear to which input characteristics an algorithm sensitises. The execution of neural networks in another domain (training in summer, execution in winter, etc.) sometimes reduces the functional quality dramatically.

This leads to DNN-specific insufficiencies and safety concerns as described in detail by Willers et al. [WSRA20], Sämann et al. [SSH20], and Schwalbe et al. [SKS+20]. In our evidence strategy, we utilise metrics to evaluate the insufficiencies and use specific values and justified bounds as specific evidence. In the following, this proce-

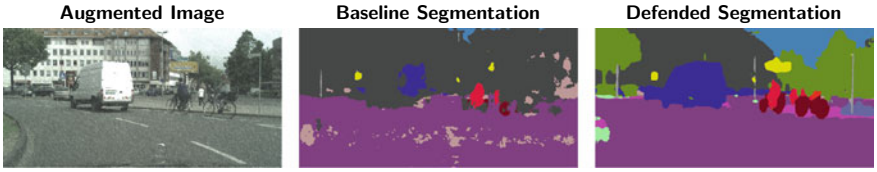


Fig. 4 Visualisation of the robustness of a semantic segmentation model: Rain augmentation via Hendryck's augmentations [HD19] on Cityscapes [COR+16] (left), baseline performance on the corrupted image (middle) and performance of a robustified model via AugMix with Jensen–Shannon divergence (JSD) loss [HMC+20] (right)

dure is shortly explained for the example of brittleness of DNNs. The strategies for the other insufficiencies follow basically the same pattern, but the elaboration would go beyond the scope of this chapter.

Measuring robustness of deep neural networks (DNNs): Regarding brittleness, the specific evidence that we want to define are test results achieving the required performance even under reasonable perturbations. If the required performance is achieved for all required conditions, we call the DNN "robust". To evaluate the robustness with respect to real-world corruptions, we suggest to use both, recorded and tagged real-word effects as well as augmented perturbations such as noise, blur, colour shift, etc., via Hendryck's augmentations [HD19] as depicted in Fig. 4. The required parameters for the test conditions are extracted from operational design domain (ODD), sensor, and data analysis.

5.3 Evidence from Design-Time Controls

Design-time controls systematically complement and integrate evidence gained via overall performance considerations or regarding specific insufficiencies. The major goal is to integrate the various aspects and metrics that have to be considered into a holistic view, allowing the AI developer, firstly to incorporate measures that are overall effective, such as architectural choices, and parameter optimisations, or data coverage measures, and secondly, to handle potential trade-offs between different optimisation targets and requirements. It is important to note that the design-time controls are to be applied in an iterative workflow during development, aiming to provide sufficient evidence for the overall safety argumentation. As a specific example of a design-time control, we sketch how developers and safety engineers can be supported by a visual analytics tool during the design and development phase.

Understanding DNN predictions via visual analytics: An important contribution to a complete safety argumentation can be made via methods that support humans in understanding and analysing why an AI system is coming to a specific decision. Insights into the inner operations of a DNN can increase trust into the entire application of that neural network. However, in a safety argumentation, it is not sufficient to

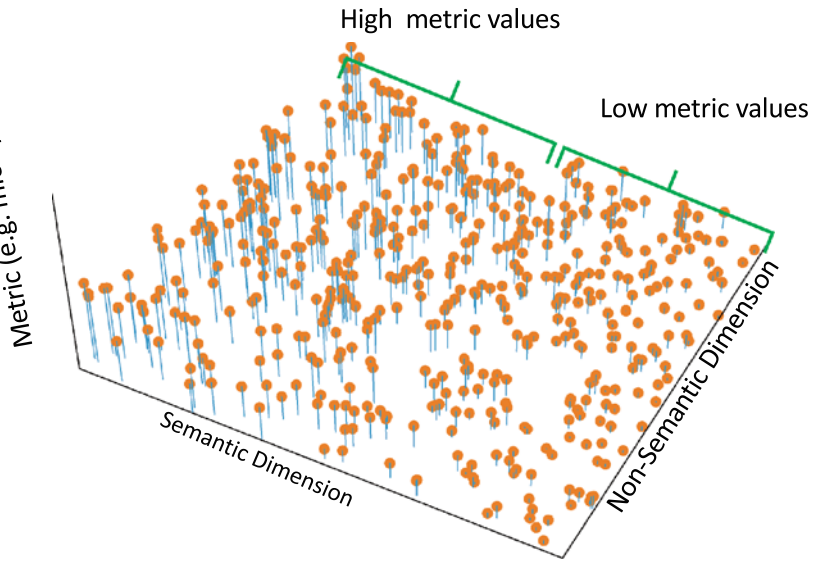


Fig. 5 Landscape of DNN performance over semantic and non-semantic dimensions

"explain" single decisions of a network, for example, in some kind of "post analysis" in case of an accident, which might have been caused by a failure of a DNN-based pedestrian detection, it should be possible to provide arguments about the understanding of the inner operations of a neural network considering a huge set of input test data at design time. The obvious solution for achieving this would be completely replacing the DNN with an interpretable model. However, this often is not feasible considering the trade-off with overall performance of the network. Ideally, such a "design-time understanding" of the overall behaviour of a DNN would result in an understanding of the network performance over a complete landscape spanned by semantic and non-semantic dimensions of the test data as illustrated in Fig. 5.

Figure 5 shows an idealised view of the performance of the DNN in a selected relevant metric, see explanations in Sect. 5.1, per input test data point over selected semantic and non-semantic dimensions. In the case of pedestrian detection in the automotive context, semantic dimensions may refer to the semantics of a scene description in the operational design domain, such as pedestrian attributes or environmental conditions, while non-semantic dimensions may refer to technical image effects, such as blurring or low contrast, which are supposed to have an influence on the performance of the DNN. Such a view can support the human to identify systematic weaknesses of a DNN in terms of human-understandable dimensions. To be able to create a view as depicted in Fig. 5, a human user, either in the role of a DNN developer, safety engineer, or auditor, is faced with two challenges:

- The sheer amount of test data needed to reach significant insights usually exceeds the cognitive capacity of the human brain.

- Finding and selecting the relevant dimensions that actually influence the network performance among all possible dimensions is far from being trivial, especially taking into account that the performance of a DNN mostly depends on a combination of different dimensions and usually cannot be explained by considering one dimension alone.

In order to overcome these two challenges, tool support is needed. In [SMHA21], an approach is described to support the human user in finding and evaluating machine learning models by means of visual analytics. In this approach, tool support is provided to enable the user to perform visual analyses over huge amounts of data in such a way that a quick and intuitive understanding of the nature of the underlying data and network performance can be gained. Such an initial understanding usually enables the user to create hypotheses about the relevance of semantic and non-semantic dimensions, which then can be checked or refined interactively. Human understanding about semantics helps in a semantic understanding of the DNN in an iterative process.

Figures 6 and 7 show a snapshot of the visual analytics tool being used to investigate the performance of a DNN for pedestrian detection based on a semantic segmentation of images. The so-called metadata that indicates the values regarding a specific dimension of an input image (such as number of pedestrians, brightness of the image, and overall detection performance in the image) and the values regarding specific pedestrians in the image (such as size, position in the image, detection performance for this pedestrian) is listed in a tabular interface as illustrated in Fig. 6. The user can issue arbitrary queries against this table to select interesting subsets of the data. The plots shown in Fig. 7 give an additional overview of various attributes. Selected images, predictions, or ground truth can be visualised as shown in Fig. 9.

Figure 8 shows an exemplary scatterplot of the pixel size of a pedestrian in an image versus the detection performance measured in IoU (intersection-over-union) for this pedestrian. The goal is to find out whether certain aspects, such as pixel size of the pedestrian in the image, have an effect on the detection performance of the underlying DNN. It can be seen that there is a general tendency for pedestrians with larger height (more to the left) to achieve higher (better) detection performance. However, there exist some large pedestrians with relatively low detection performance (as highlighted by the selection box in Fig. 8). The visual analytics tool allows for interactively selecting such sets of interesting points from a plot for further analysis just by drawing a selection box around the points of interest. Selected images will then be visualised in the drill down view of the tool shown in Fig. 9. In this particular example, some of the selected images were showing “pedestrians” riding bicycles as depicted in Fig. 9b). The human analyst could now use the visual analytics tool to further investigate the hypothesis that cyclists are not detected well enough by the DNN.

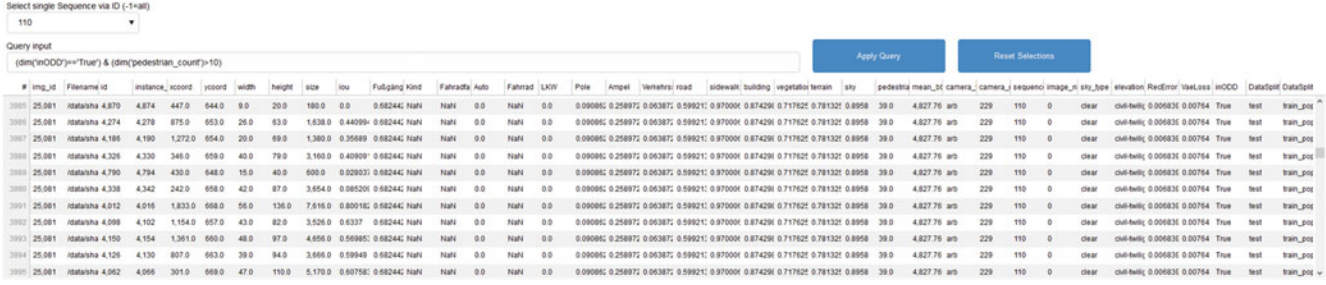


Fig. 6 Visual analytic tool for analysing DNN performance and weaknesses: tabular query interface on meta data

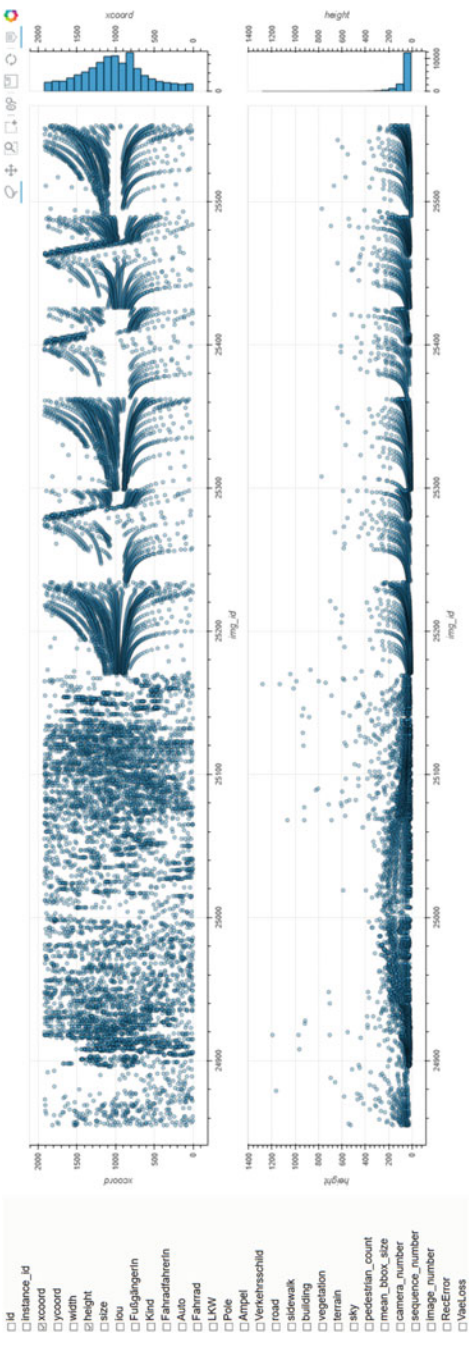


Fig. 7 Visual analytic tool for analysing DNN performance and weaknesses: plots of selected attributes

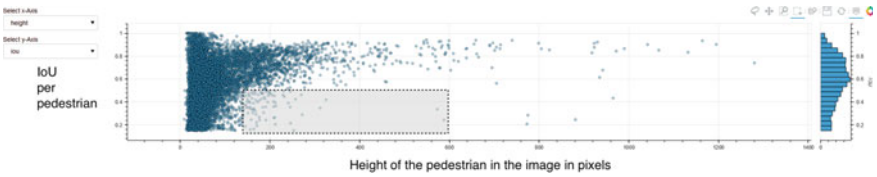


Fig. 8 Plot of the pedestrian detection performance (IoU) in dependence of the height of the pedestrian in the image in pixels

(a) Overlay of input image and ground truth

(b) Overlay of ground truth (left) and predictions (right): selected critical images often contain cyclists

Fig. 9 Visual analytic tool for analysing DNN performance and weaknesses: drill down view showing **a** a sample image, and **b** critical images selected from the correlation plot shown in Fig. 8 (Images: BIT Technology Solutions)

5.4 Evidence from Operation-Time Controls

Operation-time controls are typically used to mitigate unacceptable risk occurring during run-time of the system. For example, when the performance of a perception component for automated driving degenerates, controls could become active to mitigate the degeneration by handing vehicle controls back to a human driver or by using alternative perception components. The operation-time controls are particularly important for automated driving, since such systems operate in the real world. As during design time, not all possibly occurring situations can be foreseen, architectural solutions are required.

In this section, we give two examples of how evidence from operation-time controls can be derived. The first example deals with out-of-distribution detection and the second one goes into detail for uncertainty estimation.

Out-of-distribution detection: One of the major causes of ML insufficiencies is the fact that the data distribution of an operational domain is unknown. Therefore, it is only possible to sample from this unknown distribution and approximate this distribution by statistical approaches or machine learning (ML) techniques. During the construction of the safety contract, a certain input domain is defined at a semantic level, which directly leads to a gap to the real data distribution of the operational domain. One of the resulting ML insufficiencies is that there is an unknown behaviour when the ML function is presented with samples that are not within the training data distribution. Therefore, a requirement could be that leaving the operational design

domain (ODD) has to be detected at run-time. Such a requirement could introduce a monitoring component using complementary methods. Three example methods could be:

- When the ODD defines certain areas of operation, e.g. urban intersections, localisation information in combination with map data can be utilised to detect leaving the ODD.
- Some sensors can directly measure an environmental state, such as a rain sensor, which can be used to set bounds according to the ODD to detect whether the ODD is left.
- Out-of-distribution methods [HG17, XYA20] can be used to detect a distributional shift of the input samples during run-time with respect to the database used for training a ML model for perception or planning.

For the last method, there could furthermore be two types of evidence. Firstly, the effectiveness of the methods should be shown by a test report including an evaluation of a metric that indicates the separation precision between in- and out-of-distribution samples. Secondly, it has to be shown that the training and test data used to train the out-of-distribution detector is sufficient.

Uncertainty estimation: Uncertainty estimation methods can also be applied as operation-time controls. Uncertainty is an inherent property of any machine learning model, which may result from either aleatoric sources, such as a non-deterministic behaviour of the real world or issues in the data or labelling, or epistemic sources, referring to the inherent limitations of the machine learning model itself. Uncertainty estimation methods aim to enable the DNN to indicate the uncertainty related to a specific DNN prediction given a specific input at run-time. This may also contribute to out-of-distribution detection under the assumption that the uncertainty increases when the DNN is applied outside of its training data distribution. This assumption of course must be rigorously tested in order to provide the corresponding evidence. Among popular uncertainty estimation methods, methods based on Monte-Carlo (MC) dropout [GG16] receive particular attention in embedded applications as they approximate the performance of Bayesian Networks and hence usually outperform techniques purely based on post-processing calibration, and come with an acceptable run-time overhead (compared to, e.g. full Bayesian networks or deep ensembles). Although originally intended for capturing epistemic uncertainty only, it can be extended to capture aleatoric uncertainty as well [KG17, SAP+21].

6 Combining Safety Evidence in the Assurance Case

While the insights sketched above surely help to understand weaknesses and improve the development of the DNN under investigation, the step towards arguing evidence in an assurance case still has to be performed. According to the principles of ISO 26262, ISO/DIS 21448, and ISO/TR 4804, the assurance case shall state in a convincing

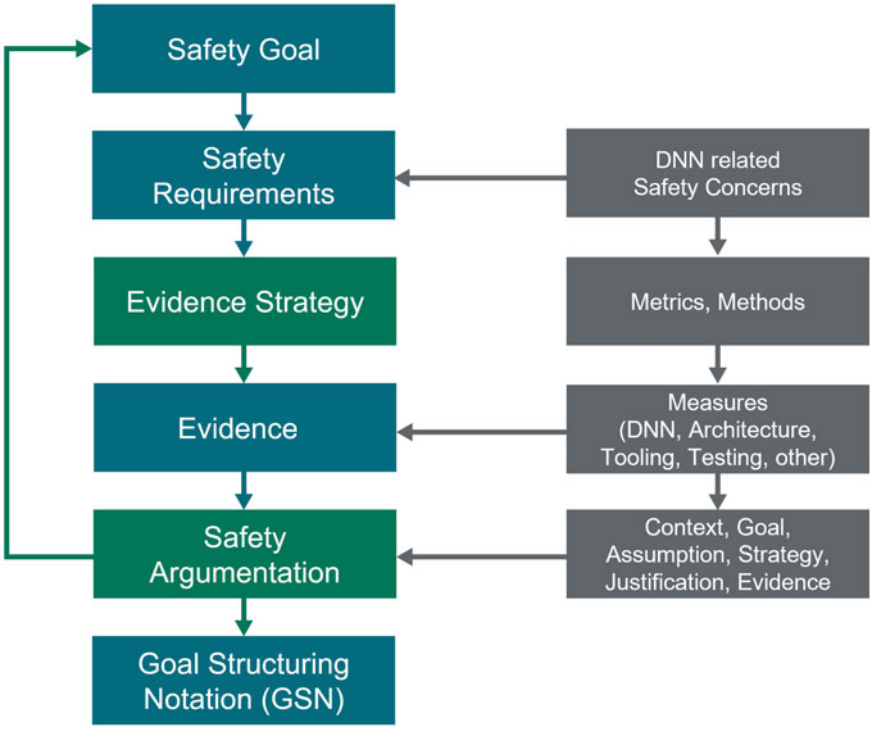


Fig. 10 Assurance case development: Integration of evidence into an assurance case

way: *"The system is safe because..."*. However, the assurance of DNNs leads to several problems, since this technology requires new paradigms in development. The software is no longer explicitly developed. Instead, the neural network is trained and the network's behaviour is implicitly influenced by the training models and data. The combination of safety evidence in the assurance case provides central elements for a holistic assurance strategy.

The core aspect of safety argumentation is to show that the mitigation of insufficiencies was successful. If the insufficiency is reduced to an acceptable level, this provides evidence to be used in the safety argumentation. As shown in Fig. 10, one possibility to combine safety evidence in the assurance case is to start on the top level with the definition of safety goals. These are goals that define mandatory steps to avoid hazards. Then the safety requirements are refined step-by-step based on the described causal model of SOTIF-related risk using the categories of evidence. This is supported by considering DNN-related safety concerns. Moreover, several metrics are defined to show the effectiveness of measures that mitigate the effects of insufficiencies. The goal structuring notation (GSN) can be used to assemble evidence collected from various methods as they were presented in Sects. 5.1, 5.2, 5.3, and 5.4 to provide a structured overall safety argumentation. The GSN visualises

the elements of the safety argumentation. An assurance case can be presented in a clear and structured way in the GSN. A GSN tree consists of three central elements: the argumentation goal, the description of the argumentation strategy, and the evidence. These three elements are supported by assumptions, justifications, and context information. A central aspect is the iterative nature of this technique to refine understanding of insufficiencies in the function. Further iterations are started on the top level (definition of safety goals).

7 Conclusions

Machine learning is an enabler technology for automated driving, especially for, but not limited to, the required perception components. However, assuring safety for such components is a challenge as standard safety-argumentation concepts are not sufficient to capture the inherent complexity and data-dependency of functions based on machine learning. For example, a component realised with an ML function is formally hard to describe, which in turn makes it especially difficult to define appropriate evidence for the safety argumentation. Therefore, we introduce different types of evidence as a structuring guidance: evidence from confirmation of residual failure rates, evidence from evaluation of insufficiencies, evidence from design-time controls, and evidence from operation-time controls.

In order to create appropriate evidence, a process is required to ensure that knowledge from different domains (machine learning, safety, and testing) are brought together. Therefore, we propose the process of evidence workstreams to define evidence in a structured way. Furthermore, we show how to integrate evidence into an assurance case.

One of the main questions still remains open: Can a convincing assurance case be constructed? We argue "yes" but only by explicitly acknowledging the insufficiencies in the ML function within the system design, and by being able to determine the residual failures with sufficient confidence. This implies directly defining an acceptable residual risk that is acknowledged by social acceptance and legal conditions, which is an open challenge.

Future research topics might be how to combine multiple quantitative and qualitative pieces of evidence into the safety argumentation w.r.t. a given system architecture and how to balance them. Moreover, there is demand for derivation of evidence from the appropriate coverage of all possible situations within the operational design domain (ODD) based on a structured ground context including tests. Further, the iterative and dynamic process of constructing the assurance case (or "continuous assurance") requires work on formal models of the assurance case and on the continuous evaluation of the assurance case.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung

von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)". The authors would like to thank the consortium for the successful cooperation.

References

[ACP21] R. Ashmore, R. Calinescu, C. Paterson, Assuring the machine learning lifecycle: desiderata, methods, and challenges. ACM Comput. Surv. (CSUR) **54**(5), 1–39 (2021)

[ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secur. Comput. (TDSC) **1**(1), 11–33 (2004)

[AOS+16] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, D. Mané, Concrete problems in AI safety (2016), pp. 1–29. arXiv:1606.06565

[BGH17] S. Burton, L. Gauerhof, C. Heinzemann, Making the case for safety of machine learning in highly automated driving, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP)* (Trento, Italy, 2017), pp. 5–16

[BHL+20] S. Burton, I. Habli, T. Lawton, J. McDermid, P. Morgan, Z. Porter, Mind the gaps: assuring the safety of autonomous systems from an engineering, ethical, and legal perspective. Artif. Intell. **279**, 103201 (2020)

[BKS+21] S. Burton, I. Kurzidem, A. Schwaiger, P. Schleiss, M. Unterreiner, T. Graeber, P. Becker, Safety assurance of machine learning for chassis control functions, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP)* (York, UK, 2021), pp. 149–162

[BMGW21] S. Burton, J.A. McDermid, P. Garnett, R. Weaver, safety, complexity, and automated driving: holistic perspectives on safety assurance. Computer **54**(8), 22–32 (2021)

[CKL21] C.-H. Cheng, A. Knoll, H.-C. Liao, Safety metrics for semantic segmentation in autonomous driving (2021), pp. 1–8, arXiv:2105.10142

[CNH+18] C.-H. Cheng, G. Nührenberg, C.-H. Huang, H. Ruess, H. Yasuoka, Towards dependability metrics for neural networks, in *Proceedings of the ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)* (Beijing, China, 2018), pp. 43–46

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The Cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 3213–3223

[GG16] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, in *Proceedings of the International Conference on Machine Learning (ICML)* (New York, NY, USA, 2016), pp. 1050–1059

[GHP+20] L. Gauerhof, R.D. Hawkins, C. Picardi, C. Paterson, Y. Hagiwara, I. Habli, Assuring the safety of machine learning for pedestrian detection at crossings, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP)* (Virtual conference, 2020), pp. 197–212

[HD19] D. Hendrycks, T. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–15

[HG17] D. Hendrycks, K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Toulon, France, 2017), pp. 1–12

[HMC+20] D. Hendrycks, N. Mu, E.D. Cubuk, B. Zoph, J. Gilmer, B. Lakshminarayanan, Augmix: a simple data processing method to improve robustness and uncertainty,

in *Proceedings of the International Conference on Learning Representations (ICLR)* (Virtual Conference, 2020), pp. 1–15

[HSRW20] M. Henne, A. Schwaiger, K. Roscher, G. Weiss, Benchmarking uncertainty estimation methods for deep learning with safety-related metrics, in *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI)* (New York, NY, USA, 2020), pp. 1–8

[Ind19] Independent High-Level Expert Group on Artificial Intelligence. *Ethics Guidelines for Trustworthy AI* (European Commission, 2019)

[ISO18] ISO. *ISO 26262: Road Vehicles—Functional Safety* (International Organization for Standardization (ISO), 2018)

[ISO20] ISO. *ISO/TR 4804: Road Vehicles—Safety and Cybersecurity for Automated Driving Systems—Design, Verification and Validation* (International Organization for Standardization (ISO), 2020)

[ISO21] ISO. *ISO/DIS 21448: Road Vehicles—Safety of the Intended Functionality* (International Organization for Standardization (ISO), 2021)

[KG17] A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision? in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Long Beach, CA, USA, 2017), pp. 5574–5584

[MJ21] J. McDermid, Y. Jia, Safety of artificial intelligence: a collaborative model, in *Proceedings of the Workshop on Artificial Intelligence Safety* (Virtual Conference, 2021), pp. 1–8

[MSB+21] M. Mock, S. Scholz, F. Blank, F. Hüger, A. Rohatschek, L. Schwarz, T. Stauner, An integrated approach to a safety argumentation for AI-based perception functions in automated driving, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP) Workshops* (York, UK, 2021), pp. 265–271

[PNdS20] R. Padilla, S.L. Netto, E.A.B. da Silva, A survey on performance metrics for object-detection algorithms, in *Proceedings of the IEEE International Conference on Systems, Signals and Image Processing (IWSSIP)* (Niteiro, Brazil, 2020), pp. 237–242

[PPD+21] R. Padilla, W.L. Passos, T.L.B. Dias, S.L. Netto, E.A.B. da Silva, A comparative analysis of object detection metrics with a companion open-source toolkit. Electronics **10**(3), 1–28 (2021)

[PPH+20] C. Picardi, C. Paterson, R.D. Hawkins, R. Calinescu, I. Habli, Assurance Argument Patterns and Processes for Machine Learning in Safety-Related Systems. In: *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI)* (New York, NY, USA, 2020), pp. 1–8

[SAE18] SAE International. *SAE J3016: Surface Vehicle Recommended Practice—Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles* (SAE International, 2018)

[SAP+21] J. Sicking, M. Akila, M. Pintz, T. Wirtz, A. Fischer, S. Wrobel, A novel regression loss for non-parametric uncertainty optimization, in *Proceedings of the Symposium on Advances in Approximate Bayesian Inference* (Virtual conference, 2021), pp. 1–27

[SKR+21] F. Schwaiger, M.H. Fabian Küppers, F.S. Roza, K. Roscher, A. Haselhoff, From black-box to white-box: examining confidence calibration under different conditions, in *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI)* (Virtual Conference, 2021), pp. 1–8

[SKS+20] G. Schwalbe, B. Knie, T. Sämann, T. Dobberphul, L. Gauerhof, S. Raafatnia, V. Rocco, Structuring the safety argumentation for deep neural network based perception in automotive applications, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP) Workshops* (Virtual Conference, 2020), pp. 383–394

[SMHA21] E. Schulz, M. Mock, S. Houben, M. Akila, *ScrutinAI: an iterative workflow for the semantic analysis of DNN predictions* (Technical report, Fraunhofer IAIS, St. Augustin, 2021)

[SQC17] R. Salay, R. Queiroz, K. Czarnecki, An analysis of ISO 26262: using machine learning safely in automotive software (2017), pp. 1–6. arXiv:1709.02435

[SSH20] T. Sämann, P. Schlicht, F. Hüger, Strategy to increase the safety of a DNN-based perception for HAD systems (2020), pp. 1–14. arXiv:2002.08935

[VGvBB20] G. Volk, J. Gamerdinger, A. von Betnuth, O. Bringmann, A comprehensive safety metric to evaluate perception in autonomous systems, in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)* (Virtual Conference, 2020), pp. 1–8

[WSRA20] O. Willers, S. Sudholt, S. Raafatnia, S. Abrecht, Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP) Workshops* (2020), pp. 336–350

[XYA20] Z. Xiao, Q. Yan, Y. Amit, Likelihood regret: an out-of-distribution detection score for variational auto-encoder (2020), pp. 1–19. arXiv:2003.02977

A Variational Deep Synthesis Approach for Perception Validation

Oliver Grau, Korbinian Hagn, and Qutub Syed Sha

Abstract This chapter introduces a novel data synthesis framework for validation of perception functions based on machine learning to ensure the safety and functionality of these systems, specifically in the context of automated driving. The main contributions are the introduction of a generative, parametric description of three-dimensional scenarios in a validation parameter space, and layered scene generation process to reduce the computational effort. Specifically, we combine a module for probabilistic scene generation, a variation engine for scene parameters, and a more realistic sensor artifacts simulation. The work demonstrates the effectiveness of the framework for the perception of pedestrians in urban environments based on various deep neural networks (DNNs) for semantic segmentation and object detection. Our approach allows a systematic evaluation of a high number of different objects and combined with our variational approach we can effectively simulate and test a wide range of additional conditions as, e.g., various illuminations. We can demonstrate that our generative approach produces a better approximation of the spatial object distribution to real datasets, compared to hand-crafted 3D scenes.

1 Introduction

This chapter introduces an automated data synthesis approach for the validation of perception functions based on a generative and parameterized synthetic data generation. We introduce a multi-stage strategy to sample the input domain of the possible generative scenario and sensor space and discuss techniques to reduce the required vast amount of computational effort. This concept is an extension and generaliza-

O. Grau (✉) · K. Hagn · Q. Syed Sha
Intel Deutschland GmbH, Lilienthalstraße 15, 85579 Neubiberg, Germany
e-mail: oliver.grau@intel.com

K. Hagn
e-mail: korbinian.hagn@intel.com

Q. Syed Sha
e-mail: syed.qutub@intel.com

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_13

tion of our previous work on parameterization of the scene parameters of concrete scenarios, called validation parameter space (VPS) [SGH20]. We extend this parameterization by a probabilistic scene generator to widen the coverage of the generated scenarios and a more realistic sensor simulation, which also allows to variate and simulate different sensor characteristics. This 'deep' synthesis concept overcomes currently available systems (as discussed in the next section) or manually, i.e., by human-operator-generated synthetic data. We describe, how our synthetic data validation engine makes use of the parameterized, generative content to implement a tool supporting complex and effective validation strategies.

Perception is one of the hardest problems to solve in any automated system. Recently, great progress has been made in applying machine learning techniques to deep neural networks to solve perceptional problems. Automated vehicles (AVs) are a recent focus as an important application of perception from cameras and other sensors, such as LiDAR and RaDAR [YLCT20]. Although the current main effort is on developing the hardware and software to implement the functionality of AVs, it will be equally important to demonstrate that this technology is safe. Universally accepted methodologies for validating safety of machine learning-based systems are still an open research topic.

Techniques to capture and render models of the real world have matured significantly over the last decades and are now able to synthesize virtual scenes in a visual quality that is hard to distinguish from real photographs for human observers. Computer-generated imagery (CGI) is increasingly popular for training and validation of deep neural networks (DNNs) (see, e.g., [RHK17, Nik19]). Synthetic data can avoid privacy issues found with recordings of members of the public and can automatically produce ground truth data at higher quality and reliability than costly manually labeled data. Moreover, simulations allow synthesis of rare scene constellations helping validation of products targeting safety-critical applications, specifically automated driving.

Due to the progress in visual and multi-sensor synthesis, building systems for validation of these complex systems in the data center becomes feasible now and offers more possibilities for the integration of intelligent techniques in the engineering process of complex applications. We compare our approach with methods and strategies targeting testing of automated driving [JWKW18].

The remainder of this chapter is structured as follows: The next section will give an outline of related work in the field. In Sect. 3 we give an overview of our approach. Section 4 describes an outline of our synthetic data validation engine, our parameterization, including a realistic sensor simulation, and the effective computation of the required variations. In Sect. 5 we present evaluation results, followed by Sect. 6 with some concluding remarks.

2 Related Work

The use of synthesized data for development and validation is an accepted technique and has been also suggested for computer vision applications (e.g., [BB95]). Several methodologies for verification and validation of AVs have been developed [KP16, JWKW18, DG18] and commercial options exist.[1] These tools were originally designed for virtual testing of automotive functions, such as braking systems, and then extended to provide simulation and management tools for virtual test drives in virtual environments. They provide real-time-capable models for vehicles, roads, drivers, and traffic which are then being used to generate test (sensor) data as well as APIs for users to integrate the virtual simulation into their own validation system.

What is getting presented in this chapter is focusing on the validation of perception functions, which is an essential module of automated systems. However, by separating the perception as a component, the validation problem can also be decoupled from the validation of the full driving stack. Moreover, this separation allows, on the one hand, the implementation of various more specialized validation strategies and, on the other hand, there is no need to simulate dynamic actors and the connected problem of interrelations between them and the ego-vehicle. The full interaction of objects is targeted by upcoming standards like OpenScenario.[2]

Recently, specifically in the domain of driving scenarios, game engines have been adopted for synthetic data generation by extraction of in-game images and labels from the rendering pipeline [WEG+00, RVRK16]. Another virtual simulator system, which gained popularity in the research community, is CARLA [DRC+17], also based on a commercial game engine (`Unreal4` [Epi04]). Although game engines provide a good starting point to simulate environments, they usually only offer a closed rendering setup with many trade-offs balancing between real-time constraints and a subjectively good visual appearance to human observers. Specifically, the lighting computation in this rendering pipelines is limited and does not produce physically correct imagery. Instead, game engines only deliver fixed rendering quality typically with 8 bit per RGB color channel and only basic shadow computation.

In contrast, physical-based rendering techniques have been applied to the generation of data for training and validation, as in the Synscapes dataset [WU18]. For our experimental deep synthesis work, we use the physical-based open-source `Blender Cycles` renderer[3] in high dynamic range (HDR) resolution, which allows realistic simulation of illumination and sensor characteristics increasing the coverage of our synthetic data in terms of scene situations and optical phenomena occurring in real-world scenarios.

The effect of sensor and lens effects on perception performance has not been studied a lot. In [CSVJR18, LLFW20], the authors are modeling camera effects to improve synthetic data for the task of bounding box detection. Metrics and parameter estimation of the effects from real camera images are suggested by [LLFW20] and

[1] For example, `Carmaker` from IPG or `PreScan` from TASS International.

[2] https://www.asam.net/standards/detail/openscenario/.

[3] https://www.blender.org/.

[CSVJR19]. A sensor model including sensor noise, lens blur, and chromatic aberration was developed based on real datasets [HG21] and integrated into our validation framework.

Looking at virtual scene content, the most recent simulation systems for validation of a complete AD system include simulation and testing of the ego-motion of a virtual vehicle and its behavior. The used test content or scenarios are therefore aimed to simulate environments spanning a huge virtual space and are then virtually driving a high number of test miles (or km) in the virtual world provided [MBM18, WPC20, DG18]. Although this might be a good strategy to validate full AD stacks, one remaining problem for validation of perception systems is the limited coverage of data testing critical scene constellations (sometimes called 'corner cases') and parameters that lead to drop in performance of the DNN perception.

A more suitable approach is to use probabilistic grammar systems [DKF20, WU18] to generate 3D scenarios which include a catalog of different object classes, and places them relative to each other to cover the complexity of the input domain. In this chapter we demonstrate the effectiveness of a simple probabilistic grammar system together with our previous scene parameter variation [SGH20] with a novel multi-stage strategy. This approach allows to systematically test conditions and relevant parameters for validation of perceptional function in a structured way.

3 Concept and Overview

The novelty of the framework introduced in this chapter is the combination of modules for parameterized generation and testing of a wide range of scenarios and scene parameters as well as sensor parameters. It is tailored towards exploration of factors that (hypothetically) define and limit the performance of perception modules.

A core design feature of the framework is the consequent parameterization of the scene composition, scene, and sensor parameters into a *validation parameter space* (VPS) as outlined in Sect. 4.2. This parameterization only considers the near proximity of the ego-car or sensor; in other words, only the objects visible to the sensor are generated. This allows a much more well-defined test of constellations involving a specific number of object types, environment topology (e.g., types and dimensions of streets), and relation of objects, usually as an implicit function of where objects are positioned relative in the scene.

This leads to a different data production and simulator pipeline than for conventional AV validation which typically provides a virtual world with a large extent to simulate and test the driving functions down to a physical level, inspired by real-world validation and test procedures [KP16, MBM18, DG18, JWKW18, WPC20].

Figure 1 shows the building blocks of our `VALERIE` system. The system runs an expansion of the VPS specified in the 'validation task' description. Our current implementation is based on a probabilistic description of how to generate the scene

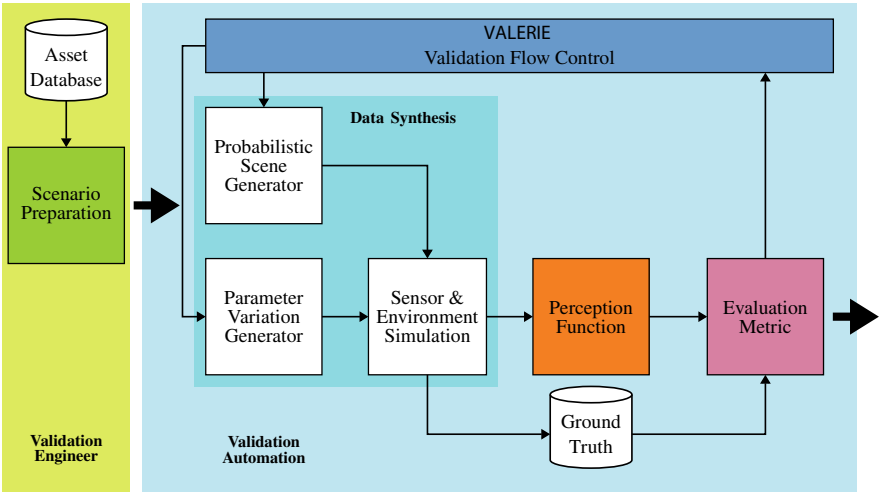


Fig. 1 Block diagram of the proposed validation approach

and defines the parameter variations in the parameter space. In the future, the validation task should also include a more abstract target description of the evaluation metrics.

The *data synthesis* block consists of three sub-components: The *probabilistic scene generator* generates a scene constellation, including a street layout, and places three-dimensional objects from the *asset database* according to probabilistic placement rules laid out in the scenario preparation. The *parameter variation generator* produces variations of that scene, including sun and light settings and variations of the placement of objects (see Fig. 2 for some examples). The *sensor & environment simulation* is using a rendering engine to compute a realistic simulation of the sensor impressions.

Further, ground truth data is provided through the rendering process, which can be used for a pixel-accurate depth map (distance from camera to scene object) or meta data, like pixel-wise label identifiers of classes or object instances. Depending on the perception task, this information is specifically used for training and evaluation of semantic segmentation (see Sect. 4.5).

The output of the sensor simulation is passed to the *perception function* under test and the response to that data is computed. An evaluation metric specific to the validation task is based on the perception response. The *ground truth* data, as generated by the rendering process is usually required here, e.g., to compute the similarity to the known appearance of objects. In the experiments presented in this chapter we used known performance metrics for DNNs, such as the mean intersection-over-union (mIoU) metric, as introduced by [EVGW+15].

The parameterization along with the computation flow are described in detail in the next section.

4 VALERIE: Computational Deep Validation

The goal of validation is usually to demonstrate that the perception function (DNN) is performing to a level defined by the validation goal for all cases included and specified in the ODD (operational design domain) [SAE18].

The framework presented in this contribution supports the validation of perception with the data synthesis modules outlined above. Further, we suggest to consider three levels or layers, which are supported in our framework:

1. The *scene variation* generates 3D scenes using a probabilistic grammar.
2. The *scene parameter variation* generates variations of scene parameters, such as moving objects or changing the illumination of the scene by changing the time of the day.
3. The *sensor variation* generates sensor faults and artifacts.

For an actual DNN validation, an engineer or team would build various scenarios and specify variations within these scenarios. The variations can contain lists of alternative objects (assets), different topology and poses (including position and orientation of objects) and expansions of streets and object and global parameters such as direction of the sun. Our modular multi-level approach enables strategies to sample the input space, typically by applying scene variation and this can be then combined with more in-depth parameter variation runs and variation of sensor parameters, as required.

Specifically, the ability to either combine two or all three levels of our framework allows to cover a wider range of object and scene constellations. In particular, with our integrated asset management approach, a validation engineer can ensure that certain, e.g., known critical object classes are included in the validation runs, i.e., he can explicitly control the coverage of these object classes. By combination with our parameter variation sub-system, local changes are varied, including relative positioning of critical objects, the positioning of the ego-vehicle or camera, global scene parameters such as the sun angle, etc. can be achieved.

4.1 Scene Generator

In computer graphics, a scene is considered as a collection $\mathcal{O} = \{\mathbf{o}_1, \mathbf{o}_2, \ldots\}$ of objects $\mathbf{o}_i$ and these are usually organized in scene graphs (see, e.g., [Wer94]) and this model is also basis for file format specifications to exchange 3D models and scenes, e.g., VRML[4] or glTF.[5]

Each object in this graph can have a position and orientation and scale in a (world) coordinate system. These are usually combined into a transformation matrix $\mathbf{T}$. Several parameterizations for position and orientations are possible, for the position

[4] ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004 https://www.web3d.org.

[5] www.khronos.org/gltf.

Table 1 Example placement description (`json` format)

```
{ "class_id" : "tree",
  "copies" : 5,
  "target": { "v1" : [59.0, 20.0, 0.15],   "v2": [59.8, 72.50, 0.15] },
  "rot_range" : [0.0, 360.0],
  "asset_list" : [
   "e649326c-061e-4881-b0a8-f2ad6297124c",
   "3a742848-4166-4256-a045-c4a8f8ef94bc",
   "4e153905-0ed6-4077-be33-74ef4c7509f5" ] }
```

usually a Cartesian 3D vector for orientation notations such as Euler angles or quaternions are common.

Objects $\mathbf{o}_i$ are described as geometry, e.g., as a triangular mesh and appearance (material).

Sensors, such as a camera, can also be represented in a scene graph and so can be light sources. Both also have a position and orientation, and accordingly, the same transformation matrices as for objects can be applied (except scaling).

The probabilistic scene generator (depicted in Fig. 1) places objects $\mathbf{o}_i$ according to a grammar file that specifies rules for placements and orientations in specific areas of the 3D scene. The example `json` file in Table 1 specifies the placement of tree objects in a rectangular area of the scene: The `tree` objects are randomly drawn from a database, the field `assets_list` specifies a list of possible assets in universally unique identifier (UUID) notation. The 3D models in the database are tagged with a `class_id`, which specifies the type of objects, e.g., humans or buildings. The class information will be used in the generation of meta-data, semantic class labels, and an instance object identifier which allows to determine on a pixel level the originating 3D object.

The placement and orientation are determined by a pseudo random number generator, with a controlled seed for the ability to exactly re-run experiments if required. The scene generator handles other constraints, such as specific target densities in specific areas, distant ranges between objects, and it finally checks that the placed object neither collides nor intersects with other objects in the scene.

The street base is also part of the input description for the scene generator. A street can be varied in width, type of crossings, and textures for street and sidewalk. In the current simplistic implementation, the street base is limited to a flat 'lego world', i.e., only rectangular structures are implemented. Each call of the scene generator generated a different randomized scene according to the rules in the generation description. Figure 2 shows scenes generated by a number of runs of the scene generator.

Fig. 2 Examples scenes with randomly selected and placed objects

4.2 Validation Parameter Space (VPS)

Another core aspect of our validation approach is to parameterize all possible variations of scene, sensor parameters, and states in a unified validation parameter space (VPS): Objects in a scene graph can be manipulated by considering their properties or attributes as a list of variable parameters. A qualitative overview of those parameters is given in Table 2. Most attributes are of geometrical nature, but also materials or properties of light sources can be varied, as depicted in Fig. 3.

In addition to static properties, a scene graph can include object properties that vary over time. Some of them are already included in Table 2, such as the trajectories of objects and sensors, indicated as $\mathcal{T} = (\mathbf{T}(t))$, with discrete time instants t. Computer

Table 2 Overview of parameters to vary in a scene

Object class	Variable parameters
Static object, e.g., buildings	Limited to position, orientation, and size
Streets, roads	Geometry (e.g., position, size of lanes, etc.), friction (as function of weather conditions)
Vehicles	$\mathbf{T}_\mathrm{v}$ = (position, orientation), trajectory $\mathcal{T}_\mathrm{v} = (\mathbf{T}_\mathrm{v}(t))$
Humans (pedestrian)	$\mathbf{T}_\mathrm{p}$ = (position, orientation), trajectory $\mathcal{T}_\mathrm{p} = (\mathbf{T}_\mathrm{p}(t))$
Environment	Light, weather conditions
Sensors	$\mathbf{T}_\mathrm{s}$ = (position, orientation), trajectory $\mathcal{T}_\mathrm{s} = (\mathbf{T}_\mathrm{s}(t))$, sensor attributes

Fig. 3 Example of scene parameter variation; in this case the time of the day is varied, causing dramatic changes in the scene illumination according to contrast variations

graphic systems handle these temporal variations, also known as animations, and in principle, any attribute can be varied over time by these systems.

We introduce an important restriction in the current implementation of our validation and simulation engine: Our animations are fixed, i.e., they do not change during the runtime of the simulation in order to allow deterministic and repeatable object appearance, like poses of characters. This could be different for example when a complete autonomous system is simulated, as the actions of the system might change the way other scene agents react. We will include these aspects in the discussion and outlook and will discuss how these aspects could be mitigated.

For the use in our validation engine, as described in the next section, we augment a description of the scene in a scene graph (the asset) as outlined above, with an explicit description of those parameters which are variable in a validation run. Currently, our engine considers a list of numerical parameters with the following attributes:

```
parameter_name, scene_graph_ref, type, minimum, maximum
```

A specific example to describe variations of the position of a person in a 2-D plane in pseudo markup notation is

```
{p1, scene.person-1.pos.x, FLOAT, 0.0, 20.0}
```

Parameters, such as `p1`, are unique parameter identifiers used in the validation engine to produce and test variations of the scene.

4.3 Computation of Synthetic Data

Synthetic data is generated using computer graphics methods. Specifically for color (RGB) images, there are many software systems available, both commercially and as open source. For our experiments in this chapter, we are using `Blender`,[6] as this tool allows importing, editing, and rendering of 3D content, including scripting.

The generation of synthetic data involves the following steps: First, a 3D scene model with a city model and pedestrians is generated using the probabilistic scene generator and is stored in one or more files.

The scene files are loaded into one scene graph and objects have a unique identifier and can be addressed by the following naming convention:

```
root_object.{subcomponent}.attribute
```

For the example used in Sect. 4.2, `scene.person-1.pos.x` refers to a path from the root object `scene` to the object `person-1` and addresses the attribute `pos.x` in `person-1`. The object names are composed of `ObjectClass-ObjectInstanceID`. These conventions are used to assign a class or instance labels during ground truth generation.

The labels for object classes will be mapped to a convention used in annotation formats (i.e., as used with in the Cityscapes dataset [COR+16]) for training and evaluation of the perception function. The 2D image of a scene is computed along with the ground truth extracted from the modeling software rendering engine.

Using a second parameter `pos.y`, as included in the example in Sect. 4.2, would allow the positioning of the person in a plane, spanned by x- and y-axis of the coordinate system defined by the scene graph.

4.4 Sensor Simulation

We implemented a sensor model with the function blocks described in the chapter 'Optimized Data Synthesis for DNN Training and Validation by Sensor Artifact Simulation' [HG22], Sect. 3.1 'Sensor Simulation', and depicted in Fig. 2. The module expects images in linear RGB space and floating point resolution as provided by the state-of-the-art rendering software.

We simulate a camera error model by applying *sensor noise*, as additive Gaussian noise (with zero mean and freely selectable variance) and an automatic, histogram-

[6] www.blender.org.

Fig. 4 Realistic sensor effect simulation: Standard `Blender` tone-mapped output (left), and the sensor simulation output (right)

based exposure control (linear tone-mapping), followed by non-linear *Gamma correction*. Further, we simulate the following lens artifacts *chromatic aberration* and *blur*. Figure 4 shows a comparison of the standard tone-mapped 8-bit RGB output of `Blender` (left) with our sensor simulation. The parameters were adapted to match the camera characteristic of Cityscape images. The images do not only look more realistic to the human eye, they also are closing the domain gap between the synthetic and real data (for details see the chapter 'Optimized Data Synthesis for DNN Training and Validation by Sensor Artifact Simulation' [HG22]).

4.5 Computation and Evaluation of Perceptional Functions

Perception functions consist of a multitude of different approaches considering the wide range of different tasks. For experiments presented in this chapter, we are considering the tasks of semantic segmentation and 2D bounding box detection. In the first task, the perception function segments an input image into different objects by assigning a semantic label to each of the input image pixels. One of the main advantages of semantic segmentation is the visual representation of the task which can be easily understood and analyzed for flaws by a human.

For semantic segmentation we consider two different topologies: `DeeplabV3+` as proposed in [CPK+18] and `Detectron2` [WKM+19], both are utilizing `ResNet101` [HZRS16] backbones.

These algorithms are trained on three different datasets to create three different models for evaluation. The first dataset is the Cityscapes dataset [COR+16], a collection of European urban street scenes during daytime with good to medium weather conditions. The second dataset is A2D2 [GKM+20]. Similar to the Cityscapes dataset it is a collection of European urban street scenes and additionally it has sequences from driving on a motorway. The last dataset, KI-A tranche 3, is a synthetic dataset

provided by BIT-TS, a project partner of the KI-Absicherung project,[7] consisting of urban street scenes inspired by the preceding two real-world datasets. All of these datasets are labeled on a subset of 11 classes which are alike in these datasets to provide comparability between the results of the different trained and evaluated models.

For the second task, the 2D-bounding box detection, we utilize the single-shot multibox detector (SSD) by [LAE+16], a 2D-bounding box detector trained on the synthetic data for pedestrian detection. This bounding box detector is applied on our variational data in Sect. 5.

To measure the performance of the task of semantic segmentation, the mean intersection-over-union (mIoU) from the COCO semantic segmentation benchmark task is used [LSD15]. The mIoU is denoted as the intersections between predicted semantic label classes and their corresponding ground truth divided by the union of the same, averaged over all classes. Another performance measure utilized is the pixel accuracy ($pAcc$) which is defined as follows:

$$pAcc = \frac{TP + TN}{TP + FP + FN + TN}. \tag{1}$$

The number of true positives (TP), true negatives (TN), false positives (FP), and true negatives (TN) are used to calculate $pAcc$, which can also be seen as a measure for correctly predicted pixels over all pixels considered for evaluation.

For the 2D-bounding box detection we are interested in cases where, according to our definition, the performance-limiting factors are within bounds where the network should still be able to correctly predict a reasonable bounding box for each object to detect. For each synthesized and inferred image, the true positive rate (TPR) is calculated. The TPR is defined as the number of correctly detected objects (TP) over the sum of correctly detected and undetected objects (TP+FN). As we are interested in prediction failure cases we can then filter out all images with a true positive rate (TPR) of 1 and are left with images where the detection has omitted objects to detect.

4.6 Controller

The `VALERIE` controller (as depicted in Fig. 1, validation flow control) executes the validation run. This run can be configured in multiple ways depending on how much synthetic data is generated and evaluated. Two aspects have a major influence on this: First, the specification of parameters to be varied, and second, the used sampling strategy, which also depends on the validation goal. Both aspects are briefly described in the following.

Specification of variable validation parameters: As outlined in Sect. 4.2, the approach depends on the provision of a generative scene model. This consists of a parameterized 3D scene model and includes 3D assets in the form of static and

[7] https://www.ki-absicherung-projekt.de/.

dynamic objects. On top of this, we define variable parameters in this scene as an explicit list, as explained in Sect. 4.2.

For the specification of a validation run, all or a subset of these parameters are selected and a range and sampling distribution for that specific parameter is added. For example, to vary the x-position of a person in the scene along a line with the uniform or homogeneous distribution and a step size of 1 m, we define

```
{p1, UNIFORM, 1.5, 5.5, 1.0}
```

The parameters refer to the following: Parameter `p1` refers to parameter declarations of x position of `person-1` in the example of Sect. 4.2. The field `UNIFORM` refers to a uniform sampling distribution. Other modes include `GAUSSIAN` (Gaussian distribution). The parameters 1.5, 5.5, 1.0 refer to the parameter range [1.5...5.5] and the initial step size of 1m.

Sampling of variable validation parameters: The actual expansion or sampling of the validation parameter space can be further configured and influenced in the `VALERIE` controller by selecting a sampler and validation strategy or goal.

The *sampler* object provides an interface to the *controller* to the validation parameter space, considering the parameter ranges and optionally the expected parameter distribution. We support uniform and Gaussian distributions.

In our current implementation, the *controller* can be configured to either sample the validation parameter space by a full grid search, or by a Monte-Carlo random sampling.

However, the step size can be iteratively adapted depending on the validation goal. One option here is to automatically refine the search for edge cases (or corner cases) in the parameter space: As an edge case, we consider here a parameter instance, where the evaluation function is changing between an 'acceptable' state to a 'failed' state (using a continuous performance metric). For our use case of person detection, that means a drop in the performance metric below a threshold.

Other validation goals we are planning to implement could be the automated determination of sensitive parameters or (ultimately) more intelligent search through high-dimensional validation parameter spaces.

4.7 Computational Aspects and System Scalability

Our approach is designed for execution in data centers. The implementation of the components described above is modular and makes use of containerized modules using `docker`.[8] For the actual execution of the modules we use the `Slurm`[9] scheduling tool, which allows running our validation engine with a high number of variants in parallel, allowing the exploration of many states in the validation parameter space.

[8] www.docker.com.

[9] https://slurm.schedmd.com.

The results presented here are produced on an experimental setup using six dual `Xeon` server nodes, each equipped with 380 GB RAM. The runtime of the rendering process as outlined above is mainly determined by the rendering and in the order of 10...15 min per frame, using the high-quality physically based rendering (PBR) `Cycles` render engine.

5 Evaluation Results and Discussion

To evaluate the effectiveness of our data synthesis approach, we conducted experiments in generating scenes, variation of a few important parameters, and then we evaluated the perception performance including an analysis of performance-limiting factors, such as occlusions and distance to objects.

We used our scene generator to generate variations of street crossings, as depicted in Fig. 2. For these examples a base ground is generated first, with flexible topology (crossings, t-junction) and dimensions of streets, sidewalks, etc. In the next step, buildings, persons, and objects, including cars, traffic signs, etc. , are selected from a database and randomly placed by the scene generator, taking into account the probabilistic description and rules. The approach can handle any number of object assets. The current experimental setup includes a total of about 500 assets, with about 60 different buildings, 180 different person models, and other objects, including vegetation, vehicles, and so on.

Scene parameter variation: Within the generated scenes, we vary the position and orientations of persons and some occluding objects.

Further, we change the illumination by changing the time of the day. This has two main effects: First, it is changing the illumination intensity and color (dominant at sunset and sunrise), and second, it is generating a variation of shadows casted into the scene. In particular, from our experience, the latter creates challenging situations for the perception.

Comparison of object distribution: Fig. 5 shows the spatial distribution of persons in a) the Cityscapes dataset, b) KI-A tranche 3 dataset, and c) a dataset using our generative scene generator, as depicted in Figs. 2 and 3. The diagrams present a top-view of the respective sensor (viewing cone) and the color encodes the frequency of persons within the sensor viewing cone, i.e., they give a representation of distance and direction of persons in all considered frames of the dataset. The real-world Cityscapes dataset has a distribution that corresponds with most persons located left and right of the center, i.e., the street. There are slightly more persons on the right side, which can be explained by the fact that often sidewalks on the left hand are occluded by vehicles from the other road side. The distribution of our dataset resembles as expected this distribution, with slightly less occupation in the distance. In contrast, the distribution of the KI-A tranche 3 dataset shows a very sharp cumulation of the distribution on what corresponds to a narrow band on the sidewalks of their 3D simulation.

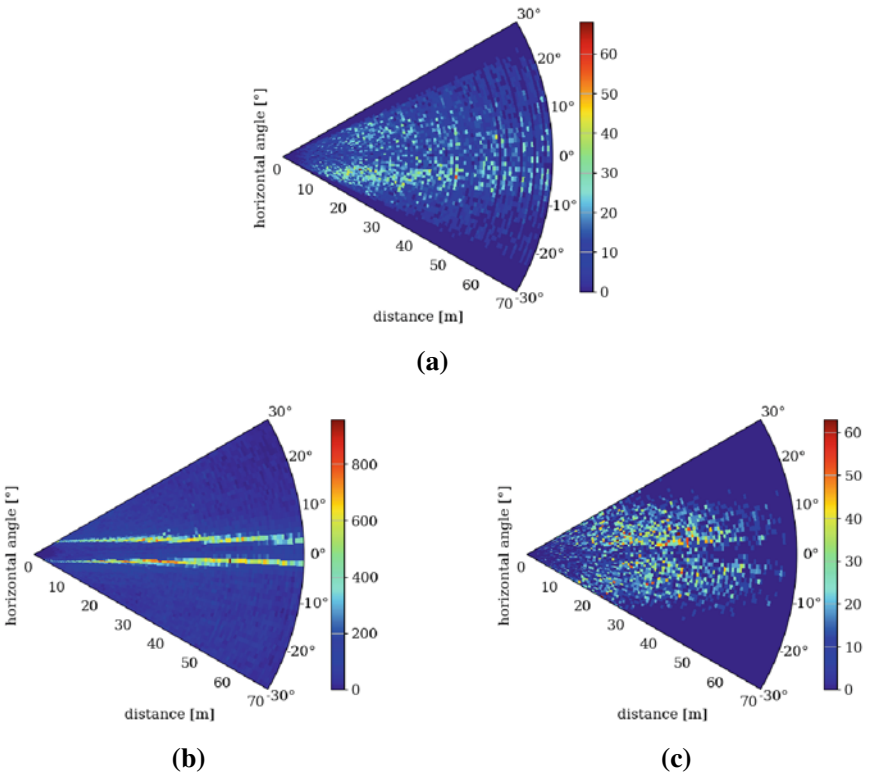


Fig. 5 Pedestrian distribution over horizontal angle and distance. **a**: Cityscapes. **b**: KI-A tranche 3. **c**: Our synthetic data

Influence of different occluding objects on detection performance: A number of object and attribute variations are depicted in Fig. 6. On the left side, the SSD bounding box detector [LAE+16] is applied to the three images with different occluding objects in front of a pedestrian. In all three images, two bounding boxes are predicted for the same pedestrian. While one bounding box includes the whole body, the second bounding box only covers the non-occluded upper part of the pedestrian. On the right side, the `DeeplabV3+` model trained on the KI-A tranche 3 is used to create a semantic map of the same three images. Besides the arms, the pedestrian is detected, even partially through the occluding fence. However, another interesting observation can be made: The ground the pedestrian stands on is always labeled as sidewalk. We interpret this as an indication to a bias in the training data, as the training data does not include enough images of pedestrians on the road, just on the sidewalk. This hypothesis can be further strengthened when we inspect the pedestrian distributions in Fig. 5b, where the pedestrians are distributed narrowly left and right off the street in the middle. Additionally, both bounding box prediction and the

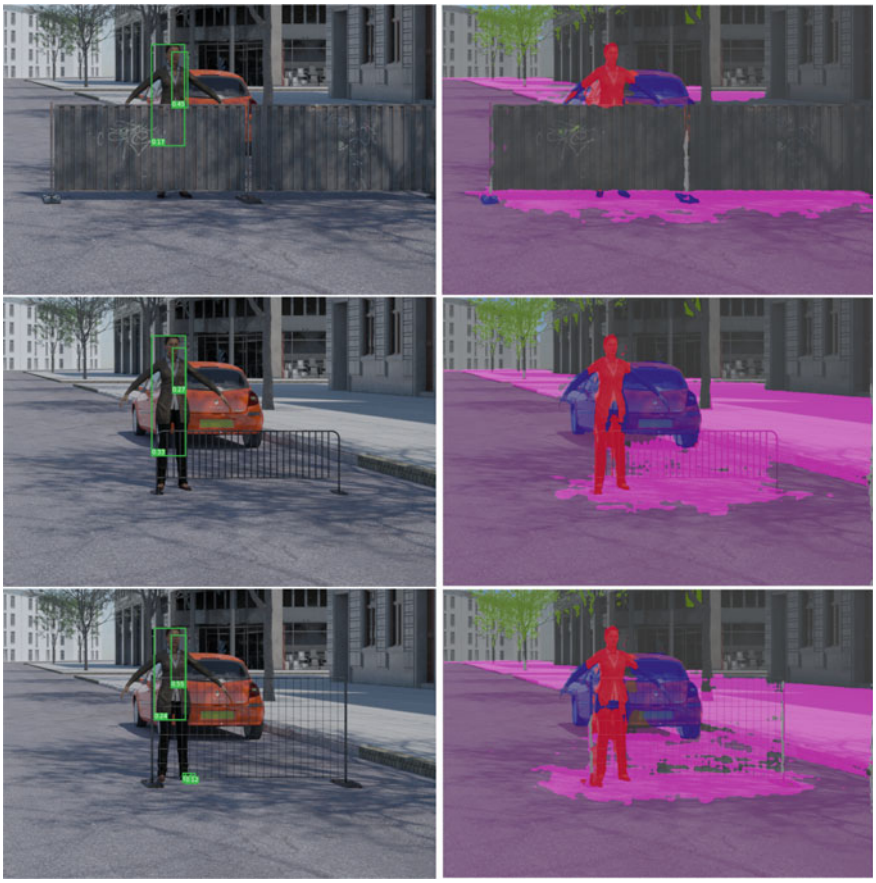


Fig. 6 Scene with variation of occluding objects. Left: 2D bounding box detection. Right: semantic segmentation

semantic segmentation do not include the pedestrian's arms in their predictions. This can also be attributed to a bias in the training data.

Influence of noise on detection performance: An experiment demonstrating our sensor simulation determines the influence of sensor noise on the predictive performance. In Fig. 7, Gaussian noise with increasing variance is applied to an image, and three `DeeplabV3+` models trained on A2D2, Cityscapes, and a synthetic dataset, respectively, are used to predict on the data. While image color pixels are represented in the range $x_i \in [0, 255]$, the noise variance is in the range of $\sigma^2 \in [0, 20]$ with a step size of 1. For each noise variance step, the mIoU performance metric on the image prediction per model is calculated. While initially the models trained on Cityscapes and the synthetic dataset increase in performance, all models' predictive

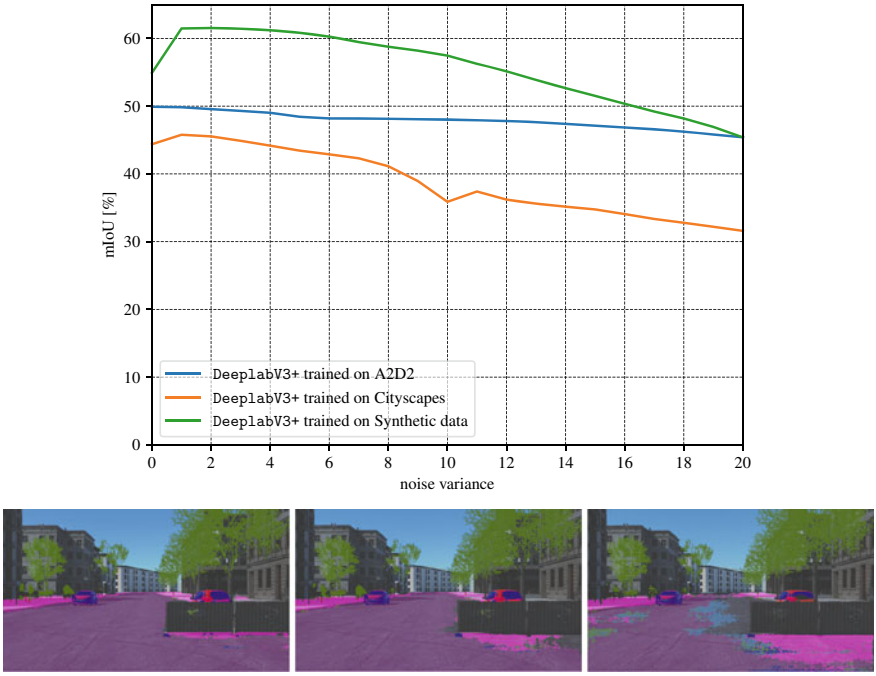


Fig. 7 Top: mIoU performance decreases with increasing noise variance. Bottom (left to right): segmentation maps with increasing noise variance $\sigma^2 \in \{0, 10, 20\}$, image pixels $x_i \in [0, 255]$

performance ultimately decreases with an increasing level of sensor noise. The initial increase can be explained to stem from the domain shift of training to validation data, where in the training data a small noise variance can be observed.

Analysis of performance-limiting factors: Some scene parameters have a major influence on the perception performance. This includes the occlusion rate of objects, with totally occluded objects that are obviously not detectable or the object size (in pixels) in the images, also with a natural boundary where detection breaks up if the object size is too small. Other performance-limiting factors include contrast and other physically observable parameters.

To measure the influence or sensitivity of perception functions against performance-limiting factors we designed an experiment using about 30,000 frames containing one person each. The person is moved and rotated on the sidewalk and on the street. The occlusions are determined by rendering a mask of the person and comparison with the actual instance mask considering occluding objects. A degree of 100% represents a fully occluded object. Figure 8 shows results of this experiment, each gray dot representing one frame and the colored curves showing regression plots with differently clothed persons.

The figure shows a $pAcc$ downwards trend with increasing occlusion rates. The `Detectron2` model (trained on Cityscapes) is comparatively more robust than

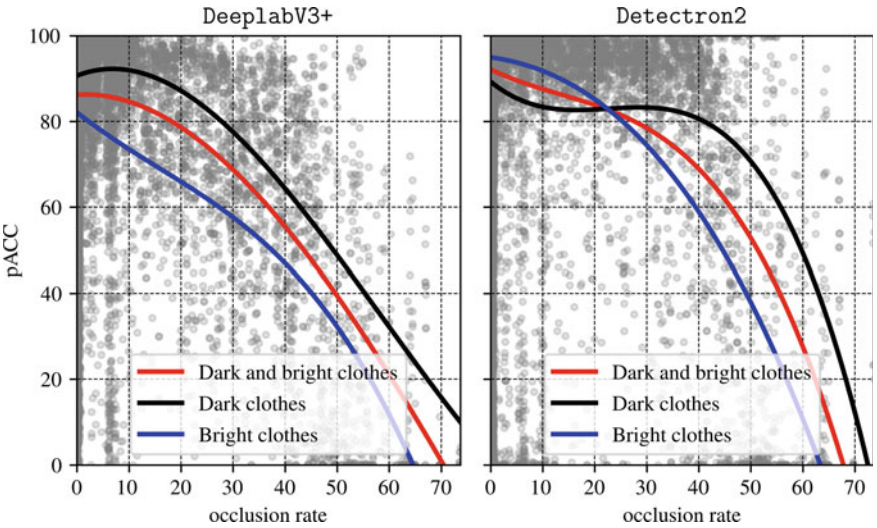


Fig. 8 Polynomial regression curves (of order 3) on pedestrian detection rate $pAcc$ of `DeeplabV3+` (left) and `Detectron2` (right) for various occlusion rates of a pedestrian wearing dark or bright clothes

`DeeplabV3`. The plot shows that `Detectron2` offers stable detection with occlusion rates $<35\%$ and then the performance drops. `DeeplabV3's` (also trained on Cityscapes) performance drops after 15% occlusion rate. The curves are not linearly following a trend due to the fact that there are other scene parameters (sunlight, shadow, direction of pedestrian) which are not constant across the rendered images.

What can also be seen in the figures is that, despite the trend of the regression curves, there is a great variation in the data—visible by the widely scattered grey points. That means that the performance depends also on other factors besides the occlusion rate. Figure 8 is showing one example of analysis possible with the metadata provided by our framework. More parameters are considered in our previous work [SGH20].

Data bias analysis: Another experiment we conducted considers failure cases, i.e., false negatives (FN) of the SSD 2D-bounding box detector regarding pedestrian detection. To accomplish this, we rendered 2640 images with our variational data synthesis engine. These images are then inferred by the SSD model and evaluated. Only pedestrians with a bounding box width greater than $0.1 \times$ image width and a height of $0.1 \times$ image height are considered valid for evaluation. Additionally, only objects with an occlusion rate below 25% are considered valid. These restrictions guarantee that pedestrians in the validation are of sufficient size, i.e., close to the camera, and clearly visible due to little occlusion and would therefore be easy to detect.

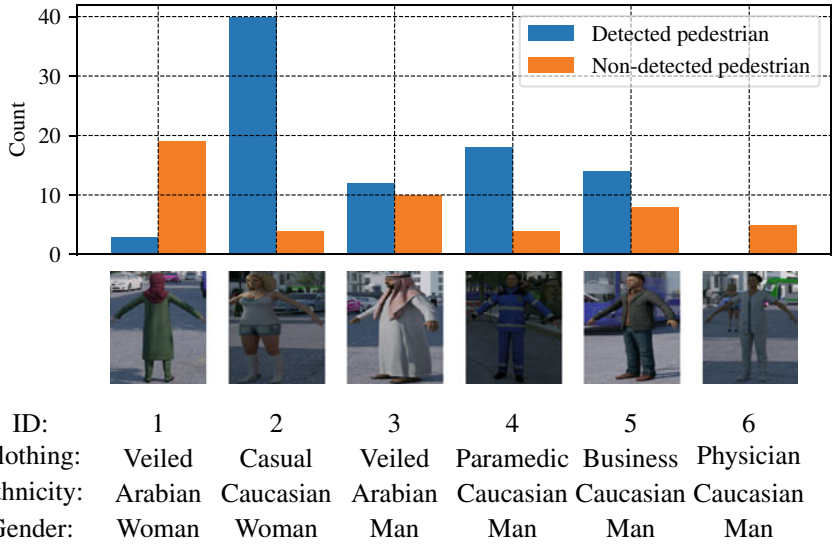


Fig. 9 Count of detected and non-detected pedestrians for different pedestrian assets, i.e., different clothing, ethnicity, and gender

With these restrictions in place we found that from all the pedestrian assets the synthesis engine placed in the scene, there were six assets that were omitted by the SSD model as can be seen in Fig. 9.

The asset ID 1 is an Arabian woman wearing traditional clothes effectively veiling the person. Asset ID 2 is a Caucasian woman clothed in summer casual, i.e., short pants and short sleeves, revealing parts of her skin. The second Arabian ethnicity asset with the ID 3 is similar to asset 1 clothed in traditional veiling clothes but of male gender. The remaining assets 4, 5, and 6 are of male gender and Caucasian ethnicity wearing different work clothes, i.e., a blue paramedical outfit for ID 4, business casual jeans and jacket for ID 5, and white physician clothes for ID 6.

The asset ID 2 with the summer casual clothed woman is only miss-detected a few times, in most cases the detection worked well, indicating no data bias for this asset. In contrast, the pedestrian asset ID 6 of a physician dressed in white hospital clothing has not been detected at all. Additionally, two of the assets that were relatively most often overlooked by the network are the Arabian clothed woman with asset ID 1, as well as an Arabian clothed man with the ID 3. This result would suggest that these kind of pedestrian assets, i.e., IDs 1, 3, and 6, were not present in the data for training the model and adding them to it will lead to a mitigation of this exact failure case.

6 Outlook and Conclusions

This chapter has introduced a new generative data synthesis framework for the validation of machine learning-based perception functions. The approach allows a very flexible description of scenes and parameters to be varied and systematical tests of parameter variations in our unified validation parameter space.

The conducted experiments demonstrate the benefits of splitting the validation process into scene variation that looks into randomized placement of objects and a variation of scene parameters and sensor simulation. Our simple probabilistic scene generator is scalable and able to produce scenes with a high number of different objects—as provided by an asset database. The spatial distribution of the positioned objects, as demonstrated for persons in Fig. 5, is more realistic compared to manually crafted 3D scenes. Along with our sensor simulation (results discussed in the chapter 'Optimized Data Synthesis for DNN Training and Validation by Sensor Artifact Simulation' [HG22]), we present a step to close the domain-gap between synthetic and real data. Future work will continue to analyze the influence of other factors, such as rendering fidelity, scene complexity, and composition, to further improve the capabilities of the framework and make it even more applicable for the validation of real-world AI functions.

Our experiments with performance-limiting factors, as shown for occlusion rates and object size (as a function of distance to the camera) in the previous section gives clear evidence that the performance of perception functions cannot be characterized by only a few factors. It is, however, a complex function of many parameters and aspects, including scene complexity, scene lighting and weather conditions, and the sensor characteristics. The deep validation approach described in this chapter is addressing this multi-dimensional complexity problem and we designed a system and methodology for flexible validation strategies to span all these parameters at once.

Our validation parameterization, as demonstrated in the results section, is an effective way to detect performance problems in perception functions. Moreover, it allows in its flexible design the sampling and a practical computation at scale allowing for deep exploration of the multi-variate validation parameter space. Therefore, we see our system as a valuable tool for the validation of perception functions.

Moving forward we are looking into using the deep synthesis approach to implement sophisticated algorithms to support more complex validation strategies. As another key direction we target improvements in the computational efficiency of our validation approach, allowing coverage of more complexity and parameter dimensions.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project 'Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)'. The authors would like to thank the consortium for the successful cooperation.

References

[BB95] W. Burger, M.J. Barth, Virtual reality for enhanced computer vision, in *Virtual Prototyping: Virtual Environments and the Product Design Process*, ed. by J. Rix, S. Haas, J. Teixeira (Springer, 1995), pp. 247–257

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The Cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 3213–3223

[CPK+18] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: semantic image segmentation with deep convolutional nets, Atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **40**(4), 834–848 (2018)

[CSVJR18] A. Carlson, K.A. Skinner, R. Vasudevan, M. Johnson-Roberson, Modeling camera effects to improve visual learning from synthetic data, in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops* (Munich, Germany, 2018), pp. 505–520

[CSVJR19] A. Carlson, K.A. Skinner, R. Vasudevan, M. Johnson-Roberson, Sensor transfer: learning optimal sensor effect image augmentation for sim-to-real domain adaptation, pp. 1–8 (2019). arXiv:1809.06256

[DG18] W. Damm, R. Galbas, Exploiting learning and scenario-based specification languages for the verification and validation of highly automated driving, in *Proceedings of the IEEE/ACM International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)* (Gothenburg, Sweden, 2018), pp. 39–46

[DKF20] J. Devaranjan, A. Kar, S. Fidler, Meta-Sim2: learning to generate synthetic datasets, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Virtual conference, 2020), pp. 715–733

[DRC+17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: an open urban driving simulator, in *Proceedings of the Conference on Robot Learning CORL* (Mountain View, CA, USA, 2017), pp. 1–16

[Epi04] Epic Games, Inc. Unreal Engine Homepage (2004). [Online; accessed 2021-11-18]

[EVGW+15] M. Everingham, L.V. Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge: a retrospective. Int. J. Comput. Vis. (IJCV) **111**(1), 98–136 (2015)

[GKM+20] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A.S. Chung, L. Hauswald, V.H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, P. Schuberth, A2D2: Audi autonomous driving dataset (2020), (pp. 1–10). arXiv:2004.06320

[HG21] K. Hagn, O. Grau, Improved sensor model for realistic synthetic data generation, in *Proceedings of the ACM Computer Science in Cars Symposium (CSCS)* (Virtual Conference, 2021), pp. 1–9

[HG22] K. Hagn, O. Grau, Optimized data synthesis for DNN training and validation by sensor artifact simulation, in *Deep Neural Networks and Data for Automated Driving—Robustness, Uncertainty Quantification, and Insights Towards Safety* ed. by T. Fingscheidt, H. Gottschalk, S. Houben (Springer, 2022), pp. 149–170

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 770–778

[JWKW18] P. Junietz, W. Wachenfeld, K. Klonecki, H. Winner, Evaluation of different approaches to address safety validation of automated driving, in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)* (Maui, HI, USA, 2018), pp. 491–496

[KP16] Nidhi Kalra, Susan M. Paddock, Driving to safety: how many miles of driving would it take to demonstrate autonomous vehicle reliability? Transp. Res. Part A: Policy Pract. **94**, 182–193 (2016)

[LAE+16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Amsterdam, The Netherlands, 2016), pp. 21–37

[LLFW20] Zhenyi Liu, Trisha Lian, Joyce Farrell, Brian Wandell, Neural network generalization: the impact of camera parameters. IEEE Access **8**, 10443–10454 (2020)

[LSD15] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, MA, USA, 2015), pp. 3431–3440

[MBM18] T. Menzel, G. Bagschik, M. Maurer, Scenarios for development, test and validation of automated vehicles, in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)* (Changshu, China, 2018), pp. 1821–1827

[Nik19] S.I. Nikolenko, *Synthetic Data for Deep Learning* (2019), pp. 1–156. arXiv:1909.11512

[RHK17] S.R. Richter, Z. Hayder, V. Koltun, Playing for benchmarks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Venice, Italy, 2017), pp. 2232–2241

[RVRK16] S.R. Richter, V. Vineet, S. Roth, V. Koltun, Playing for data: ground truth from computer games, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Amsterdam, The Netherlands, 2016), pp. 102–118

[SAE18] SAE International. SAE J3016: surface vehicle recommended practice—taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. SAE Int. (2018)

[SGH20] Q.S. Sha, O. Grau, K. Hagn, DNN analysis through synthetic data variation, in *Proceedings of the ACM Computer Science in Cars Symposium (CSCS)* (Virtual Conference, 2020), pp. 1–10

[WEG+00] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, A. Sumner, et al., TORCS—The Open Racing Car Simulator (2000). [Online; accessed 2021-11-18]

[Wer94] J. Wernecke, *The Inventor Mentor: Programming Object-Oriented 3D Graphics With Open Inventor* (Addison-Wesley, 1994)

[WKM+19] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, R. Girshick, Detectron2 (2019). [Online; assessed 2021-11-18]

[WPC20] Mingyun Wen, Jisun Park, Kyungeun Cho, A scenario generation pipeline for autonomous vehicle simulators. HCIS **10**, 1–15 (2020)

[WU18] M. Wrenninge, J. Unger, Synscapes: a photorealistic synthetic dataset for street scene parsing (2018), pp. 1–13. arXiv:1810.08705

[YLCT20] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, Kazuya Takeda, A survey of autonomous driving: common practices and emerging technologies. IEEE Access **8**, 58443–58469 (2020)

The Good and the Bad: Using Neuron Coverage as a DNN Validation Technique

Sujan Sai Gannamaneni, Maram Akila, Christian Heinzemann, and Matthias Woehrle

Abstract Verification and validation (V&V) is a crucial step for the certification and deployment of deep neural networks (DNNs). Neuron coverage, inspired by code coverage in software testing, has been proposed as one such V&V method. We provide a summary of different neuron coverage variants and their inspiration from traditional software engineering V&V methods. Our first experiment shows that novelty and granularity are important considerations when assessing a coverage metric. Building on these observations, we provide an illustrative example for studying the advantages of pairwise coverage over simple neuron coverage. Finally, we show that there is an upper bound of realizable neuron coverage when test data are sampled from inside the operational design domain (in-ODD) instead of the entire input space.

1 Introduction

In the past few years, there has been rapid growth in the usage of deep neural networks (DNNs) for safety-critical applications. Computer vision models, in particular, are being used in, e.g., autonomous driving or medical diagnostics. This shift in use cases is met by increasing demand for verification and validation (V&V) methods for DNNs [BEW+18, ZHML20, RJS+20]. However, unlike in traditional software

S. S. Gannamaneni (✉) · M. Akila
Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS,
Schloss Birlinghoven 1, 53757 Sankt Augustin, Germany
e-mail: sujan.sai.gannamaneni@iais.fraunhofer.de

M. Akila
e-mail: maram.akila@iais.fraunhofer.de

C. Heinzemann · M. Woehrle
Robert Bosch GmbH, Corporate Research, Robert-Bosch-Campus 1,
71272 Renningen, Germany
e-mail: christian.heinzemann@de.bosch.com

M. Woehrle
e-mail: matthias.woehrle@de.bosch.com

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_14

engineering, developing rigorous techniques for V&V of DNNs is challenging as their intrinsic working is not directly of human design. Instead, the relevant parameters are learned, and their meaning often eludes the human engineer.

With the strengthened focus on V&V for DNNs, there was a gradual shift away from mere performance metrics to developing techniques that measure and investigate, e.g., (local) interpretability or robustness. While insightful, such approaches focus on specific instances of a given (test) dataset. However, they typically do not detail the required number of elements in those datasets in the sense of sufficient testing.

Inspired by testing in software engineering, Pei et al. proposed a variant of code coverage called neuron coverage (NC) [PCYJ19]. In analogy to checking if a line of code is executed for any given test sample, NC provides us with the percentage of activated neurons for a given test dataset.

For a test (or in this case: coverage) metric, the granularity or "level of difficulty" is crucial; if a test is too easy to fulfill, the chances are low that it will uncover errors. If, on the other hand, it is barely possible (or even impossible) to fully perform the test, it becomes ill-defined as no stopping criterion exists or can be found (cf. [SHK+19]). This makes the level of granularity of a test a decisive criterion. We can illustrate this on the example of classical code coverage for the pseudo-code shown in Algorithm 1.

Algorithm 1 Code coverage example

```
1: Set a
2: c ← 2
3: if a > 2 then
4:     c ← −4
5: end if
6: if a < 4 then
7:     a ← c · a
8: end if
9: return a + c
```

It takes a (real) input variable a and maps it onto a (real) output. Following code coverage requirements, we would need to find inputs a_i such that in total, each line of code is executed at least once. The two values $\{a_1 = 1, a_2 = 5\}$ would lead the algorithm to either pass through the first or the second if-clause, respectively, and would thereby together fulfill this requirement. The advantage of this test is that, besides being simple, it has a well-defined end when the full code is covered. Conversely, should not all of the code be coverable, it directly revealed unreachable statements that can be removed from the code without changing its functionality. The latter would be seen as a step towards improving the code's "readability" and as avoiding potentially unintended behavior, should those parts of code be reachable by some unforeseen form of a statement. However, this form of coverage can provide only a rudimentary test of the algorithm. For instance, the output in the example differs significantly if the algorithm runs through both clauses simultaneously, which would be the case, e.g., for $a_3 = 3$. We cannot (or at least not with a reasonable amount

of effort) test a given code for all potential eventualities. However, we can evaluate with more complex test criteria, pair-wise tests in the place of the single-line tests. A reasonable approach could be to test if both if-clauses "A" and "B" were used in the same run or whether only one of them was passed. While this approach would surely have uncovered the potentially erroneous behavior from the third input a_3, it also scales quadratically with the number of clauses, while the previous criterion scaled linearly. Additionally, it is a priori unclear to which extent an algorithm can be covered in principle. For instance, in the example, no value for a can be found such that both if-clauses would be skipped simultaneously.

The above example on algorithmic code coverage can be transferred to the coverage of neurons within a DNN. Assuming, as we will also do in the following, an architecture based on ReLU non-linearities, where $\text{ReLU}(y) = \max(y, 0)$, we may count a unit (neuron) as covered, if, for a given input, its output after application of the ReLU is non-zero.[1] Full coverage, for this measure, would thus be achieved if a given test set $\mathcal{X}$ can be found such that for any unit of the DNN, at least one data sample $\mathbf{x}_i \in \mathcal{X}$ exists where that unit is covered. As with the code coverage before, the inability to construct or find such a sample $\mathbf{x}_i$ could indicate that the unit in question is superfluous and might be removed (pruned) without changing the functionality of the DNN. The generalization to pair-wise metrics is straightforward but owing to the often sequential structure of feed-forward DNNs, a further restriction to adjacent layers is considered, i.e., pairs of only adjacent layers are considered when evaluating with this metric. There are, however, two important distinctions to code coverage. At first, interaction among layers, although reminiscent of the single control flow in the above code example, is significantly more complicated. Examining Algorithm 1 more closely, the interaction was mediated by a latent variable c. To some extent, each unit within a layer can be seen as such a variable allowing for various behaviors in the subsequent layer. Moreover, not activating a unit is not an omission but a significant statement in its own right, compare for instance [EP20]. As a second point, input parameters and variations in classical software are often reasonably understood. For DNNs, on the contrary, often their use is motivated by the inability to specify such variations. Therefore, a secondary use of coverage metrics to determine *the novelty of a data point* $\mathbf{x}_j$ w.r.t. a test set $\mathcal{X}'$ ($\mathbf{x}_j \notin \mathcal{X}'$) becomes apparent by measuring how strongly the coverage values between $\mathcal{X}'$ and $\mathcal{X}' \cup \{\mathbf{x}_j\}$ differ. Such a concept is based on the assumption that coverage, to some degree, represents the internal state (or "control flow" if seen from a software point of view) of the DNN and could thus be used to determine the completeness of the test set in the sense that no further novel states of the DNN could be reached.

In this chapter, we address the following aspects: On the level of single unit coverage, we investigate the effect of granularity, i.e., to which degree layer-wise resolved metrics allow for a more detailed test coverage. In this context, we also investigate whether the coverage metric is informative regarding the novelty of samples by comparing trained and untrained DNNs on CIFAR-10. For both single and pairwise

[1] While generalizations to non-zero thresholds exist, see for instance [MJXZ+18], we restrict ourselves to this case as it is a natural choice for ReLU non-linearities.

coverage, we address under which circumstances (full) coverage can be reached. In a (synthetic) toy example, we also differentiate how strongly these statements depend on the restrictions posed on the test set data when staying within a DNN's intended input domain (i.e., the operational design domain, ODD).

2 Related Works

Pei et al. [PCYJ19] introduced neuron coverage, i.e., looking at the percentage of neurons activated in a network, as a testing method. Using a dual optimization goal, they generate realistic-looking test data that increases the percentage of activated neurons. The method proposed in [MJXZ+18], a variant of [PCYJ19], uses the same definition of NC and additionally introduces k-multisection coverage and neuron boundary coverage. Another variant, [TPJR18], uses image transformations on synthetic images to maximize neuron coverage.

Sun et al. [SHK+19], inspired from modified condition/decision coverage (MCDC) in the software field, proposed four condition decision-based coverage methods. Furthermore, they showed that the proposed methods subsumed earlier works, i.e., satisfying their proposed methods also satisfies the weaker earlier coverage methods.

However, there are also several works expressing criticism of NC as a validation technique [DZW+19, HCWG+20, AAG+20]. While in [DZW+19], it is shown that there is limited to no correlation between robustness and coverage, Harel-Canada et al. [HCWG+20] show that test sets generated by increasing NC, as defined in [PCYJ19, TPJR18], fail in specific criteria such as defect detection, naturalness, and output impartiality. Abrecht et al. [AAG+20] showed that there was a discrepancy in the way how neurons in different layers were evaluated in earlier works due to differing definitions of what constitutes a neuron in a DNN. Furthermore, they show that evaluating NC at a more granular level provides more insight than just looking at coverage over the entire network. However, they also show that even this granular coverage is still easy to achieve with simple augmentation techniques performing better than test data generation methods like DeepXplore [PCYJ19].

Abrecht et al. [AGG+21] presented an overview of testing methods of computer vision DNNs in the context of automated driving. This includes a discussion of different types of adequacy criteria for testing and concretely of structural coverage for neural networks as one form of an adequacy criterion that supports testing.

In this chapter, we follow the definitions of layer-wise coverage (LWC) [AAG+20] to determine NC and adjust it for pairwise coverage (PWC). While the fundamental definition, i.e., a neuron or a unit is considered as covered as long as there is at least one sample $\mathbf{x}_i$ in the test set $\mathcal{X}$ such that the unit produces a non-zero output, remains unchanged across definitions, the precise meaning of what constitutes a unit differs. Compared to other metrics, e.g., DeepXplore [PCYJ19], LWC provides a more fine granular resolution of coverage, especially concerning convolutional layers. While one could treat each channel of the convolutional layer as a single unit with respect

to coverage, LWC demands that each instance of the use of the filter is counted as an individual unit to be covered. In terms of an example, if a two-channel convolutional layer results in an output of $3 \times 3 \times 2$ real numbers, where the first two numbers are due to the size of the input image, LWC would require to cover each of the 18 resulting "neurons" while DeepXplore would consider only two "neurons", one for each channel.

3 Granularity and Novelty

Several works [SHK+19, AAG+20] have shown that achieving high NC is trivial with only a few randomly selected in-domain non-adversarial images. Abrecht et al. [AAG+20] have further shown that trained and untrained networks show similar NC behavior for a given test dataset. Intuitively, one would expect that, first, a metric used to investigate the safety of a DNN in terms of finding new input samples should not be easily satisfiable and, second, that the metric should depend on the "quality" of the input. Typically, a DNN is trained to solve a specific problem that only covers a very small subset of all possible inputs, e.g., classifying an image as either showing a cat or dog could be contrasted by completely unfitting inputs such as the figures in this chapter. Following the analogy of code coverage, this would be the difference between expected and fully random (and mostly unfitting) inputs, where for the latter, one would expect nonsensical outputs or error messages. Especially for neural networks, this second point is of relevance as they are typically used in cases where a distinction between reasonable and unreasonable input can be made only by human experts or DNNs but not via "conventional" algorithms. This makes an extension of the input coverage to novel examples challenging, as "errors" discovered on erroneous inputs might lead to questionable insights into the DNNs trustworthiness or reliability.

Within this section, we elaborate on the two raised concerns by investigating an example DNN on the level of (single-point) NC. For this, we will look not only at the reached coverage but take a closer look at its saturation behavior. Further, we compare both trained and untrained (e.g., randomly initialized) versions of the DNN using either actual test data (from the dataset) or randomly generated Gaussian input. To be more precise, we use `VGG16` [SZ15] networks with ReLU activations and train them on CIFAR-10 [Kri09], a dataset containing RGB images of (pixel-, channel-) size $32 \times 32 \times 3$ with 10 classes (containing objects, such as automobiles, but also creatures, such as frogs or cats) from scratch. After training,[2] our network reaches an average accuracy of 82% on the test set. As stated above, we employ the layerwise coverage (LWC) notion defined by Abrecht et al. [AAG+20] to measure NC as it subsumes less granular metrics, such as the often-used definitions of DeepXplore [PCYJ19]. Subsumption implies that if DeepXplore does not reach full coverage, neither will LWC but not necessarily the other way around.

[2] We employ an SGD optimizer with a learning rate of 10^{-3} for 20 epochs and a learning rate decay of 0.1 at epochs 10 and 15.

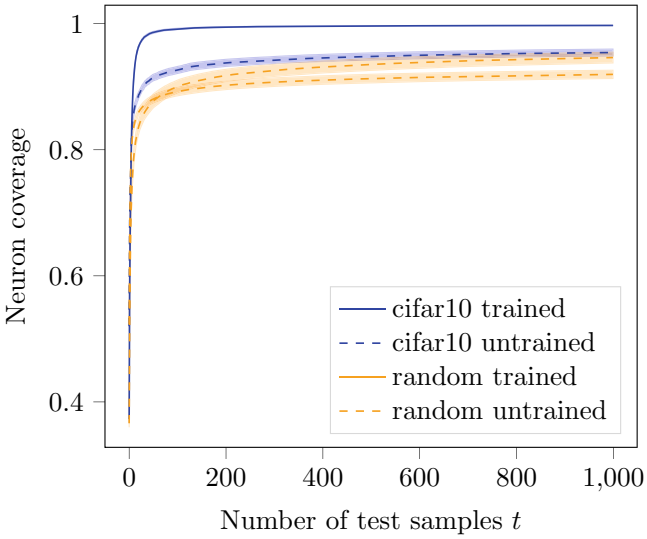


Fig. 1 Here, we show the neuron coverage in a `VGG16` network. The blue solid and dashed lines show mean coverage for trained and untrained models on CIFAR-10 test images, respectively. The orange solid and dashed lines show mean coverage for trained and untrained models on randomly generated data samples, respectively. The standard deviation is shown for each experiment as the shaded area

In Fig. 1, we plot the NC of the used `VGG16` DNNs with respect to 1000 test samples for all four test cases. For each case, we employed five different DNNs, independently initialized and/or trained, and averaged the results for further stability. For evaluation, we use both the real CIFAR-10 test set data and randomly generated data samples from a Gaussian distribution loosely fitting the overall data range. While the coverage behaves qualitatively similar in all cases, a clear separation between (assumed) limiting values and saturation speeds can be seen. This difference is most pronounced between the trained DNN that is tested with CIFAR-10 test data, representing the most structured case as both the data and the weights carry (semantic) meaning, and all other cases, which contain at least one random component.

A rough understanding of the behavior of NC can be acquired when considering the following highly simplified thought experiment. Assuming that the activation of a neuron was an entirely probabilistic event in which a given input sample activates the neuron with probability $0 < p < 1$, then the chance of said neuron to remain inactive for t samples would be q^t with $q := 1 - p$. While this suggests an exponential saturation rate for the coverage of a single neuron, the idea only carries over to the full set of all considered neurons if one assumes further that their activation probabilities p' were identical and their activations independent. Even omitting only the first condition (identicality) can lead to qualitatively different behavior, in which specific properties would depend on the distribution of p's among the neurons. A typical

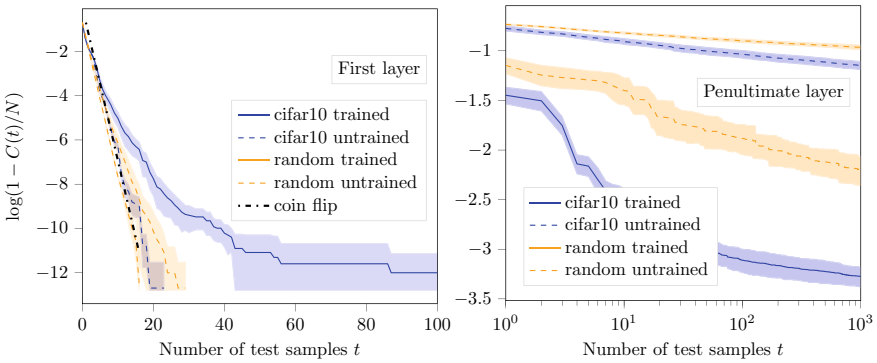


Fig. 2 Log plot of mean coverage in first (left) and penultimate (right) layer for trained and untrained networks. In the first layer, neuron coverage shows similar behavior for trained and untrained networks. However, in the penultimate layer, there is a significant difference in behavior between trained and untrained networks. The standard deviation is shown as shaded region. As we consider $\log(1 - C(t)/N)$, with the number of covered neurons $C(t)$, total neurons N and test samples t, the value approaches "negative infinity" when full coverage is achieved. This behavior can be seen in the figure on the left. The x-axis on the right is on log scale

expectation for a broad distribution would be an approximately algebraic saturation.[3] Unsurprisingly, we can recover both types of behavior within the experiments.

A connection to the above thought experiment is obtained when investigating NC layer-wise. While from the perspective of testing, one might argue that the coverage of any given layer is subsumed in the coverage of the full model, the differing behavior of the layers warrants investigating NC on this level of granularity. For the first and the penultimate layer of `VGG16`, the results are depicted in Fig. 2. In the first layer, one would expect that, at least for random input, each neuron is equally well suited to be activated, and thus saturation would be reached exponentially. As can be seen, this is almost ideally given for the untrained DNN if tested on random input, and in good approximation still when using the structured CIFAR-10 input or when testing the trained DNN with the random samples. In the latter case, the decision boundaries of each trained neuron are "random" with respect to the incoming data, which does not represent any meaningful semantic concepts or correlations. Note that we do not show the coverage in the figure but rather the number of not-yet-activated neurons in a logarithmic fashion; straight lines represent exponential saturation. Regarding the speed of saturation, i.e., the exponent or probability p, the naive assumption is that a neuron separates the input via a half-plane and each case, activation or inactivation, is equally likely. This corresponds to a coin-flip with probability $p = q = 1/2$ and is depicted in the figure for comparison. Only the trained DNN, when tested on actual test data, deviates from this behavior and saturates more slowly. This might be

[3] Consider, for comparison, how a superposition or mixture of Gaussians turns into a heavy-tailed distribution, e.g., Student's t, if the standard deviations of each Gaussian differ strongly enough, cf. [Bar12].

explained by the fact that the network can use feature-based correlation in the data, and a new test sample is therefore of a more limited novelty than a fully random, and thus unexpected one.

Turning to the penultimate layer, shown on the right-hand side in Fig. 2, we choose a double-logarithmic visualization such that algebraic saturation is now given by straight lines. To some extent, all four test cases seem to satisfy this type of saturation. However, the exponent, i.e., the speed of saturation, differs significantly and is slowest for both cases of the untrained DNN. A possible explanation might simply be the propagation of information that is not optimized in a random DNN. By design, a (classification) DNN maps a large input (here 3072-dimensional) onto a low-dimensional output (10 classes) and, thus, has to discard large amounts of information along the layers. This heuristic assumption is supported by the behavior of the trained DNN, which achieves high coverage significantly faster, even more so if the input contains features it was optimized to recognize (i.e., actual test data).

As shown, NC does depend on the training and the use of an appropriate test set. Even then, the final coverage might be reached only slowly. However, returning to the original Fig. 1, relatively high coverage is reached early on with only a few samples. It is, therefore, more a question of resolving a "hand-full" of not-covered neurons that might not be receptive to the data used. In the next section, we will look more closely into whether or not all neurons can even be covered if the used data are restricted to the operational domain of the network. This property, catching large amounts of the coverage with only limited effort, might not be desirable from the perspective of testing, especially if the novelty of samples is supposed to be judged. For such use cases, more granular tests might be more appropriate. A straightforward extension might be to include not only "positive" neuron coverage (NC+) as done before but also "negative" coverage (NC-) where a neuron was not activated at least once. For a neuron, it might be the case that it is almost always active, and therefore the rare cases where it does not fire might be of interest. For this reason, both types of coverage are to some degree independent, i.e., while it can be possible for a DNN to reach full NC+ coverage, this does not necessarily ensure full (NC-) coverage, and vice versa. Especially for networks with ReLU activations, (NC-) coverage is of interest as the non-linearity only takes effect when a neuron is not active. For a discussion, see, for instance, [EP20].

As discussed in the introduction, a further logical extension could be a switch from measuring single neuron activations or inactivations to a notion of pairwise coverage (PWC). Here, we follow a simplified version of the sign-sign coverage introduced in [SHK+19]. Due to the typically feed-forward type of network structure, we limit ourselves to adjacent layers with a pair of neurons as the unit of interest. A pair of neurons, $n_{\ell,i}, n_{\ell+1,j}$ from layers ℓ and $\ell+1$, counts as covered with respect to PWC++, if for a given input both neurons are active, i.e., assuming values greater than 0. Similarly, we define PWC−− for cases where the pre-activations are both non-positive, and the neuron outputs thus zero. Likewise, the two alternating activation patterns, PWC+– and PWC–+, can be defined. It is easy to see that if a DNN is fully PWC++ (or PWC−−) covered, it is also fully NC+ (NC–, respectively) covered. Therefore, the PWC metrics subsume the single neuron NC metrics. However, this

Table 1 NC and PWC: This table shows the big difference between the number of countable units of neurons in NC and pairwise evaluation in PWC. Neurons are defined as per LWC [AAG+20]. For simplicity, we do not consider the effect of skip connections

Network	NC [10^5]	PWC [10^5]
`LeNet-5` [LBBH98]	0.081	81
`VGG-16` [SZ15]	3.159	82,030
`VGG-19` [SZ15]	3.425	85,428
`ResNet-18` [HZRS16]	0.548	1,673
`ResNet-34` [HZRS16]	0.804	2,265
`ResNet-50` [HZRS16]	2.309	12,117
`ResNet-101` [HZRS16]	3.354	13,721

increased granularity comes at a price. While the number of tests or conditions required to be fulfilled to reach full NC coverage scales linearly with the number of neurons or units, pairwise metrics follow a quadratic growth. Due to the restriction to neighboring layers, the growth is effectively reduced by the number of layers, and the number of conditions can be approximated by $(\#\text{neurons})^2/\#\text{layers}$. Exact numbers for a selected few architectures are reported in Table 1 including the `VGG16` example used so far. Note that for simplicity, skip connection combinations, although forming direct links between remote layers, are not considered. Including them into the pairwise metric would lead to larger numbers. Even without, the sheer magnitude shows that it might be unlikely to test for full coverage. Additionally, it is unclear whether full coverage can be reached in principle if the neuron output is correlated across layers. As this might pose a problem for tests that require an end criterion signifying when the test is finished, we include those more elaborate metrics in the next section where we investigate whether full coverage can be reached.

4 Formal Analysis of Achievable Coverage

In this section, we analyze coverage metrics based on a very simple task and a simple domain as described in Sect. 4.1. The key design objectives for the creation of the task are that (i) arbitrary many data samples can be generated, (ii) a ground truth oracle exists in closed form, and (iii) the trained network and the input domain are sufficiently small such that specific activation patterns can be computed analytically. We describe the experimental setup in detail in Sect. 4.2, before providing the experimental results in Sect. 4.3, followed by a discussion in Sect. 4.4.

4.1 Description of the Task

Our simple task is counting the number of input "pixels" with positive values. We restrict ourselves to a system with four input pixels, whose values are constrained to the interval $\mathcal{I} = [-1, 1]$. We call this the operational design domain (ODD) of our network, following [KF19]. This is also our training distribution. We can obviously also perform inference on samples outside this ODD, but these will be outside the training distribution since we only train on ODD samples. We frame this task as a classification problem with five output classes: the integers from 0 to 4. Thus, our input domain is $\mathcal{I}^4$, our output domain is $\mathcal{O} = \{0, 1, 2, 3, 4\}$, and we can define a ground truth oracle as cardinality $|\{i | i \in \mathcal{I}, i > 0\}|$, i.e., we count the pixels with a value above 0. In addition, we explicitly define all inputs in $\mathbb{R}^4 \setminus \mathcal{I}^4$ as out-of-distribution samples, with $\mathcal{I}^4 \subset \mathbb{R}^4$.

The rationale of our approach is as follows: What can be achieved with perfect knowledge of the system in the sense of (*i*) symbolically computing inputs to achieve activation patterns, (*ii*) having a clear definition of in-ODD and out-of-ODD, and (*iii*) having a perfect labeling function for novel examples?

For this task, we use a multilayer perceptron (MLP) with three hidden fully connected layers FC(n) with n output nodes. The architecture of the MLP is shown in Fig. 3. We train our network using `PyTorch`. We generate one million inputs uniformly sampled from $\mathcal{I}^4$. We split these samples into training and test data sets

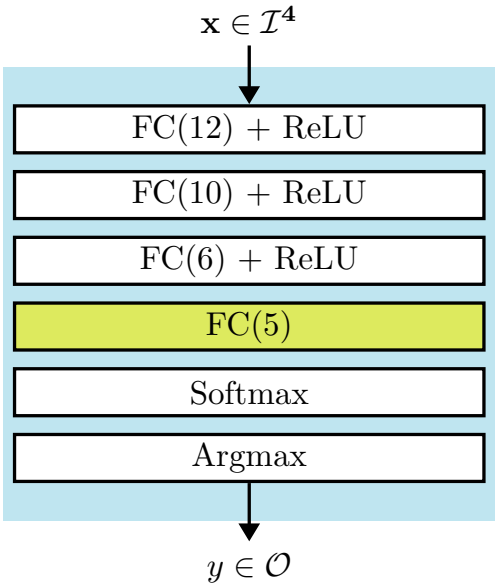


Fig. 3 Architecture of the MLP

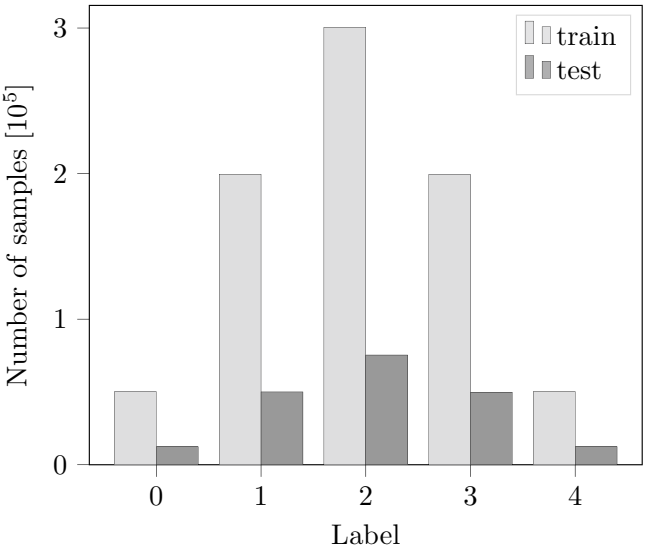


Fig. 4 Distribution of labels in training and test set, relative distribution is identical for both

Table 2 Accuracy [%] overall and per class for MLP_{adv} and $\text{MLP}_{\text{plain}}$. Best numbers are in boldface. Performance of $\text{MLP}_{\text{plain}}$ is slightly better, but differences are insignificant

Network	Overall	0	1	2	3	4
MLP_{adv}	99.54	**99.94**	99.64	99.16	99.83	99.89
$\text{MLP}_{\text{plain}}$	**99.89**	99.87	**99.83**	**99.88**	**99.96**	**99.97**

using an 80/20 split, i.e., we end up with 800,000 training samples and 200,000 test samples. The distribution of the respective ground truth labels is shown in Fig. 4. The distribution of the samples is as expected as there are 16 unique combinations of positive and negative pixels in $\mathcal{I}$ of which 6 yield two positive pixels, 4 yield one or three positive pixel(s), respectively, and 1 combination yields 0 or 4 positive pixels, respectively.

In the following, we study the impact of coverage models and the ODD on specific networks to show specific effects rather than a statistical evaluation. Nevertheless, we show that this is not a singular effect by training the MLP, whose architecture is depicted in Fig. 3, with two training setups, leading to two different MLPs, MLP_{adv} and $\text{MLP}_{\text{plain}}$. We train both MLPs for 100 epochs using ADAM as an optimizer with default `PyTorch` settings, a batch size of 100, standard cross-entropy loss, and a single initialization. For increasing robustness, MLP_{adv} leverages adversarial training, here with the fast gradient sign method (FGSM) [GSS15], using $\epsilon = 10^{-4}$, which corresponds to the attack strength. For $\text{MLP}_{\text{plain}}$, we use the same training setup but do not apply adversarial training. Table 2 summarizes the final performances on the test set for the overall accuracy and the per-class accuracy for both networks.

We can see that $\text{MLP}_{\text{plain}}$ has a slightly higher accuracy overall, as well as for four out of the five output classes, than MLP_{adv}. Note that the lower accuracy of MLP_{adv} is in line with standard robust training and shows that when trying to train a robust network, more capacity may be necessary [GSS15]. We focus in the following on the adversarially trained model for the sake of brevity since both setups resulted in similar results.

4.2 Experimental Setup

In our experiments, we want to evaluate whether full coverage according to neuron coverage (NC) and pairwise coverage (PWC) can be achieved. The networks for the experiments are the trained MLPs described in the previous section. The goal of the experiment is to calculate a minimal number of inputs that maximizes coverage on the MLP and, if required, a number of non-coverable items.

Since the MLP is simple enough, we can solve the coverage generation symbolically. For this, we use a satisfiability modulo theory (SMT) solver, namely, `Z3` [dMB08], and encode the problem of finding inputs maximizing coverage as SMT queries based on the theory of unquantified linear real arithmetic, i.e., Boolean combinations of inequations between linear polynomials over real variables. Note that there are also further approaches based on SMT targeting larger neural networks like `Reluplex` [KBD+17], which are, however, not needed for our example. We iteratively compute solutions for individual neurons (or combinations of neurons) as shown in Algorithm 2: We determine all coverage items for a neural network as a set $\mathcal{B}$. While there are still coverage items, we select one of these and try to cover it as described in the following. We formulate the coverage of item $i \in \mathcal{B}$ as an SMT query and call the SMT solver on this query using `runSMT()`. If it succeeds as indicated by the return value *success* being true, we run the network `NN` on the resulting input $\mathbf{x}$ by executing `callMLP()`. This returns all resulting coverage items $\mathcal{J}$, which are added to the set of already covered coverage items $\mathcal{C}$. Minimally, we find the coverage item i from the SMT query, but there may be more. If the SMT solver cannot cover item i, we add it to the set of uncoverable items $\mathcal{U}$. Note that this implicitly also depends on a specific coverage metric, which we omit in the algorithm for simplicity of the presentation.

The function `runSMT()` generates individual samples based on a program for `Z3` [dMB08], or `Z3Py`[4], respectively, as a list of constraints. These constraints encode the network, the input domain, and the coverage item. When the SMT determines the constraint system as satisfiable, it returns a model. From this model, we can extract the corresponding inputs to the network that lead to a satisfiable result.

For each coverage item, we run two experiments with and without a constrained input domain. First, we use constraints as shown above that allow the `Z3` solver to only consider in-ODD samples from $\mathcal{I}$[4]. Second, we also allow `Z3` to generate

[4] https://pypi.org/project/z3-solver.

Algorithm 2 High-Level Algorithm

```
1: procedure ComputeSMTCoverage(NN)                ▷ NN is the network to be covered
2:     B ← getCoverageItems(NN)
3:     C ← ∅                                       ▷ items already covered
4:     U ← ∅                                       ▷ uncoverable items
5:     while B \ {C ∪ U} ≠ ∅ do
6:         Select i ∈ B \ {C ∪ U}
7:         success, x ← runSMT(NN, i)              ▷ x is an input for NN
8:         if success then
9:             J ← callMLP(x)                      ▷ compute all possible coverage items for x
10:            C = C ∪ J
11:        else
12:            U = U ∪ {i}
13:        end if
14:    end while
15: end procedure
```

arbitrary samples from $\mathbb{R}^4$, i.e., we allow `Z3` to also generate out-of-ODD samples. Naturally, the achievable coverage using only in-ODD samples can be at most as high as the coverage for the entire $\mathbb{R}^4$.

As a baseline, we compare the coverage achieved by the samples from `Z3` with two sampling-based approaches: (*i*) We use random sampling from the input domain (further described below) since it is a commonly applied strategy for data acquisition. (*ii*) Due to the compactness of our input space, we can use a grid-based approach of the input domain. We include this comparison to check how easy or difficult it is to achieve maximum coverage using sampling-based approaches. In particular, we use a grid-based sampling with a fixed number of samples per dimension, and we sample at random from a distribution. As for the structured generation, we perform two experiments for each approach: One permitting only in-ODD samples from $\mathcal{I}^4$, and one also permitting out-of-ODD samples. Of course, since we know the exact maximum coverage for each of the considered coverage metrics from `Z3`, we are only interested in how far below the optimum the sampling-based approaches remain.

For the grid-based approach, we sample from $\mathcal{I}^4$ with a resolution of 100 equidistant samples per dimension for the in-ODD samples. In addition, we sample from $\mathcal{R}^4 \subset \mathbb{R}^4, \mathcal{R} = [-2, 2]$ with a resolution of 100 equidistant samples per dimension for a mixture of in-ODD and out-of-ODD samples. For the random sampling approach, we sample 10^8 data points uniformly at random from $\mathcal{I}^4$ for in-ODD samples. In addition, we sample 10^8 data points from a Gaussian distribution with zero mean and unit variance, which results in 78% out-of-ODD samples. For all data sets, we measure coverage based on `PyTorch` [PGM+19] as described above in Algorithm 2 for the SMT-based approach.

4.3 Experimental Results

Table 3 shows the number of generated samples, whereas Table 4 summarizes the coverage results from our experiments, both for MLP_{adv}. Please note that these results are for a single network. However, we are not interested in the specific numbers here but rather compare different coverage models and data generation approaches qualitatively on a specific network. The structure of Table 4 is as follows: Each row contains the results for one data generation approach. Thus, the first row contains the result when optimizing for positive neuron coverage (NC (+)), i.e., getting a positive activation for each individual neuron. The second row contains the results for neuron coverage in both directions (NC (both)), i.e., each individual neuron has at least once a positive and a negative activation. The third row contains the results when optimizing PWC. The final two rows contain the results for the two baseline sampling approaches.

The columns of the table show the resulting coverage for the different coverage metrics. For each column, we provide separate columns *All* and *ODD*: The *ODD* columns contain the results when only in-ODD samples are used, whereas the *All* columns contain the result when also allowing for out-of-ODD samples. For NC (+) and NC (both), the cells contain the coverage results for each of the three layers, denoted by L1 to L3. Please note that L1 is the layer directly after the input and L3 constructs the representation, which is used in the final classification layer L4. For PWC, we further split the *All* and *ODD* columns based on the layer combinations that are considered. The L1–L2 columns contain the results for the coverage of the interaction between the neurons on layers L1 and L2 while the L2–L3 columns contain the results for the interaction between L2 and L3, respectively. In these columns, the entries in the single cells refer to the coverage of the four possible combinations of activations for the considered pairs of neurons. As an example, in column L1-L2, "$+-$" refers to all cases where a neuron in L1 has a positive activation, and a neuron in L2 has a negative activation. In each row, we marked the cells with gray background where the reported coverage corresponds to the optimization target in `Z3`. These entries signify the maximally possible coverage values.

In the following, we highlight some interesting aspects of the results. Note that, in general, the coverage in the *All* column must be larger or equal to the coverage in the *ODD* column for each of the coverage metrics as $\mathbb{R} \supset \mathcal{I}$.

Intuitively, we would expect that the more elaborate a coverage metric is (i.e., the more elements it requires to be covered), the more test samples are required to achieve coverage. In our results, this expectation is satisfied, i.e., we need more tests

Table 3 Number of tests per approach

	NC (+)	NC (both)	PWC	Grid	Sample
All	6	10	39	10^8	10^8
ODD	5	10	47	10^8	10^8

Table 4 Results from the coverage analysis, Lℓ means layer ℓ. The rows show different data generation approaches (solver-based and sampling-based). The columns show coverage based on different coverage metrics. Grey shaded cells show the results of metric that was optimized in SMT

Data Generation Approach	Results for different coverage metrics							
	NC (+)		NC (–)		PWC			
	All	ODD	All	ODD	All		ODD	
					L1-L2	L2-L3	L1-L2	L2-L3
NC (+)	L1: 1.00 L2: 1.00 L3: 1.00	L1: 1.00 L2: 0.80 L3: 0.83	L1: 1.00 L2: 0.90 L3: 1.00	L1: 0.50 L2: 0.60 L3: 0.50	++: 0.85 +−: 0.81 −+: 0.62 −−: 0.69	++: 0.82 +−: 0.72 −+: 0.83 −−: 0.73	++: 0.67 +−: 0.56 −+: 0.34 −−: 0.26	++: 0.57 +−: 0.37 −+: 0.47 −−: 0.27
NC (both)	L1: 1.00 L2: 1.00 L3: 1.00	L1: 1.00 L2: 0.80 L3: 0.83	L1: 1.00 L2: 1.00 L3: 1.00	L1: 1.00 L2: 1.00 L3: 1.00	++: 0.89 +−: 0.84 −+: 0.79 −−: 0.77	++: 0.92 +−: 0.80 −+: 0.78 −−: 0.77	++: 0.78 +−: 0.81 −+: 0.65 −−: 0.68	++: 0.67 +−: 0.72 −+: 0.82 −−: 0.53
PWC	L1: 1.00 L2: 1.00 L3: 1.00	L1: 1.00 L2: 0.80 L3: 0.83	L1: 1.00 L2: 1.00 L3: 1.00	L1: 1.00 L2: 1.00 L3: 1.00	++: 0.98 +−: 1.00 −+: 0.97 −−: 1.00	++: 0.95 +−: 0.95−+: 0.98 −−: 0.97	++: 0.80 +−: 0.94 −+: 0.78 −−: 0.88	++: 0.67 +−: 0.77 −+: 0.82 −−: 0.77
Grid	L1: 1.00 L2: 1.00 L3: 0.83	L1: 1.00 L2: 0.80 L3: 0.83	L1: 1.00 L2: 1.00 L3: 1.00	L1: 1.00 L2: 1.00 L3: 1.00	++: 0.97 +−: 0.99 −+: 0.96 −−: 0.95	++: 0.82 +−: 0.92 −+: 0.82 −−: 0.88	++: 0.79 +−: 0.93 −+: 0.76 −−: 0.88	++: 0.67 +−: 0.73 −+: 0.82 −−: 0.75
Sample	L1: 1.00 L2: 1.00 L3: 0.83	L1: 1.00 L2: 0.80 L3: 0.83	L1: 1.00 L2: 1.00 L3: 1.00	L1: 1.00 L2: 1.00 L3: 1.00	++: 0.98 +−: 1.00 −+: 0.96 −−: 0.99	++: 0.82 +−: 0.92 −+: 0.83 −−: 0.93	++: 0.80 +−: 0.94 −+: 0.78 −−: 0.88	++: 0.67 +−: 0.75 −+: 0.82 −−: 0.77

for PWC than for NC, and we need more tests for NC (both) than for NC (+) as shown in Table 3. In addition, we would expect that coverage is proportional to the number of tests for the first three rows, i.e., more tests imply higher coverage, given that we only add new tests if they actually increase coverage. Interestingly, this is not satisfied for PWC, where we generate more tests for in-ODD samples than for *All*, even though the resulting coverage for *ODD* is lower than for *All*. It seems that in our particular MLP, it is easier to stimulate multiple neurons at once when using out-of-ODD samples.

Let us now consider NC. When using arbitrary test samples, it is possible to achieve 100% coverage for both NC (+) and NC (-). However, when restricting the generation to only in-ODD samples, i.e., a subset of the complete input space of the MLP, full coverage can no longer be reached for NC (+). In particular, we observe that some neurons on layers L2 and L3 can no longer be stimulated to have a positive

activation. Considering the use of SMT leading to this observation, we thus not only have empirical evidence but also a tool-based mathematical proof that 100% coverage is not possible in this case.[5]

If you compare the coverage in the NC (+) column with the corresponding coverage for the two sampling-based approaches, you will notice that they achieve the same non-100% coverage for in-ODD samples and a smaller non-100% coverage for *All* samples. Thus, the sampling-based approaches failed to achieve 100% coverage for NC (+) in both cases. While we know in this example that no better result was theoretically possible for in-ODD samples, more coverage could have been possible for *All*. However, for a more complex task, these two cases cannot be distinguished, i.e., if we have coverage less than 100%, we cannot decide whether it is impossible to achieve more or whether our test set is yet incomplete. In addition, the result shows that even in this simple domain, a relatively high number of samples do not achieve maximal coverage. Thus, it is highly unlikely to achieve the theoretically possible coverage with respect to more involved metrics such as PWC on complex domains in computer vision using random or grid-based sampling.

As stated above, we can achieve higher coverage when also allowing for out-of-ODD samples. In our example, the network was not trained on such out-of-ODD samples, so the question remains whether it is actually meaningful to include such samples in a test set just for increasing coverage. In our example, we can give some intuition on this. For this, we look in detail at samples generated by our SMT-based approach with and without input constraints. The results are shown in Fig. 5, where we plot samples SMT generated for the NC (both) criterion. Each run creates 10 samples, as is shown in Table 4. Each of these samples is labeled according to our labeling function, and the corresponding cross-entropy loss J^{CE} is computed. Since the loss varies drastically, we show a logarithmic plot and since the loss can be zero for perfect predictions, we specifically plot

$$\log(J^{\text{CE}} + 1). \tag{1}$$

Remember from Table 3 that there are 10 samples, each for the full input space and the ODD space. As we can see, for both input domains, for the first three samples, the network generates perfect predictions and therefore $\log(J^{\text{CE}} + 1) := 0$. For the ODD samples, we can see that the loss is quite low and at most in the order of the maximum training loss (as indicated by the dotted line). We also annotate for each sample of the full input space (All) whether the resulting sample is in the ODD, labeled as All (in) and indicated by a dot, or whether the sample is out-of-ODD, labeled as All (out) and indicated by a diamond shape. As we can see, when optimizing coverage without constraints (All), most of the samples (7/10) produced by `Z3` are out-of-ODD, even though also in-ODD samples would exist for the most part. In the figure, we can see that these out-of-ODD samples are not suitable for a realistic evaluation: Each out-of-ODD sample has a very high loss, at least an order of magnitude higher than

[5] Please note that the large-scale experiments with random and grid-based sampling provide additional empirical evidence.

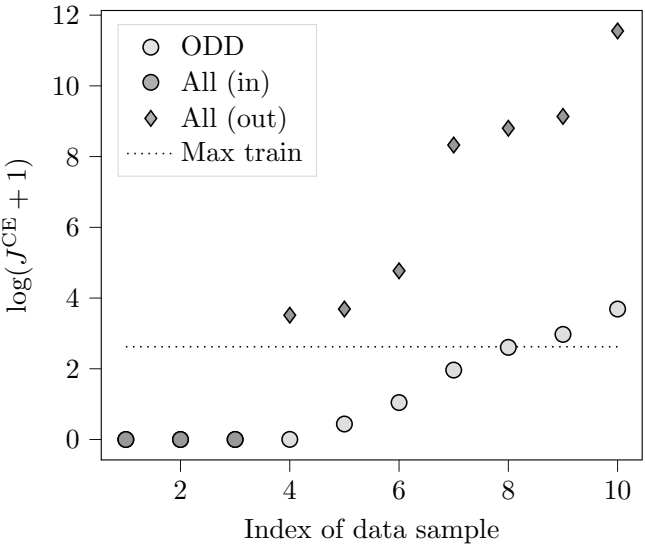


Fig. 5 Log-modified cross-entropy losses (see (1)) on samples generated by SMT for NC (both). ODD refers to in-ODD samples, All (in/out) refer to in-ODD/out-of-ODD samples sampled from a mixed in-ODD/out-of-ODD input space, and Max train is the maximum training loss

the maximum training loss, marked by a dotted line. It can also be observed that in-ODD samples have a significantly lower loss. Nevertheless, even here, we see that the SMT-based approach generates two very difficult examples.

Intuitively, we would expect that achieving a high coverage is more difficult on deeper layers because achieving a specific activation also requires establishing necessary activations on the previous layers. This means these previous layers work like constraints for the computation of an input that creates an activation on a deeper layer. Thus, the deeper the layer, the more constraints need to be satisfied. While our results confirm this intuition in most cases, these are not as strong as you can see, e.g., when comparing the coverage for PWC. Here, the coverage for L2–L3 is only slightly lower in three out of four cases than for L1–L2, and in the fourth case, it is even higher for the deeper layers. This is most probably the case because the network is rather shallow.

Note that we additionally performed these experiments on $\mathrm{MLP}_{\mathrm{plain}}$. While the results are largely the same in direction (reductions), the magnitude of differences is slightly larger. We hypothesize that this is due to the lower robustness of representations and, thus, preferred to show the results for the adversarially trained MLP.

4.4 Discussion

Our results for the formal analysis of achievable coverage clearly show that even for simple networks, 100% coverage may not be achievable. This has two reasons: (*i*) Networks are trained in a way that they contain some activations that may only ever be used in one direction. We can see this in the *All* columns of Table 4 that have coverage levels $< 100\%$. In traditional software, we would refer to this as "dead code" that results in non-perfect coverage. One could argue that as with "dead code" in traditional software, there may be a way to remove this by some form of static analysis. (*ii*) Even if we remove this "static dead code", we can still see a difference between *All* and *ODD*. This difference is harder to identify in practice as—in contrast to our simple example—this distinction between *All* and *ODD* is not always obvious for computer vision tasks. Hence, this is a harder form of "dead code" depending on the particular input/operational distribution. Both of these sources result in incomplete coverage.

Let us summarize major differences between state-of-the-art DNNs used, e.g., in computer vision for automated driving functions, and our presented MLP example:

- In a complex input domain, we do not have a clear definition of ODD and All, and we may not be able to decide whether a specific input is reasonable or not. For example, it is not clear whether a new image lies in the challenging part of the ODD or is considered to lie outside the ODD.
- We usually want to focus on in-ODD examples.
- Formal approaches such as the one used in these examples do not scale to typical state-of-the-art DNNs. Note that we cannot directly use more efficient, approximate works, e.g., on adversarial examples, since they perform an over-approximation while we are interested in under-approximations for coverage.

As a result, the identification of both sources of "dead code" is typically infeasible. This means that we do not know the real upper threshold for achievable coverage and must over-approximate it with the complete number of coverage items, including both sources of "dead code".

5 Conclusion

In summary, while neuron coverage remains promising in certain aspects for V&V, it comes with many caveats. The granularity of the metric in use determines how effective the metric is at uncovering faults. The granularity of application of the metric, layerwise instead of full model, also reveals further interesting behavior of the model under test. Similarly, when studying the novelty of input samples, we saw that random inputs show more novelty and lead to more coverage due to their unexpected nature. Structured input with a trained network has lower novelty in comparison. While test generation using NC focuses on generating realistic novel

samples, it is unclear if the increase in coverage from these generated samples is due to some random noise injected into the generated image or meaningful semantic change.

Coverage metrics are also hard to interpret if the metric converges to a value below 100%. This means if we determine that a certain neuron is not coverable, we cannot determine (*i*) whether it is coverable with *some* input, and moreover (*ii*) with some input inside our ODD. Note that, depending on the ODD, using an adversarial example in some ϵ-environment may not be reasonable for the ODD. Consequently, the coverage metrics are not actionable in the sense that we can derive information on which particular inputs we lack in our test set.

As we have shown, filling the test set with out-of-ODD samples for increasing coverage has a questionable effect. Since they are out-of-ODD, the network under test will usually not be trained on such samples. As a result, the network needs to extrapolate for classifying them and yields a high loss in our samples in Fig. 5. However, in real tasks, we do not necessarily know whether the newly generated data is out of the ODD, and the high loss may indicate that such samples should further be considered, e.g., added to the training set.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)". The authors would like to thank the consortium for the successful cooperation.

References

[AAG+20] S. Abrecht, M. Akila, S.S. Gannamaneni, K. Groh, C. Heinzemann, S. Houben, M. Woehrle, Revisiting neuron coverage and its application to test generation, in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP) Workshops* (Virtual Conference, 2020), pp. 289–301

[AGG+21] Stephanie Abrecht, Lydia Gauerhof, Christoph Gladisch, Konrad Groh, Christian Heinzemann, Matthias Woehrle, Testing deep learning-based visual perception for automated driving. ACM Trans. Cyber-Phys. Syst. **5**(4), 1–28 (2021)

[Bar12] D. Barber, *Bayesian Reasoning and Machine Learning* (Cambridge University Press, 2012)

[BEW+18] M. Borg, C. Englund, K. Wnuk, B. Duran, C. Levandowski, S. Gao, Y. Tan, H. Kaijser, H. Lönn, J. Törnqvist, Safely entering the deep: a review of verification and validation for machine learning and a challenge elicitation in the automotive industry, pp. 1–29 (2018). arXiv:1812.05389

[dMB08] L. Mendonça de Moura, N. Bjørner. Z3: an efficient SMT solver, in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)* (Budapest, Hungary, 2008), pp. 337–340

[DZW+19] Y. Dong, P. Zhang, J. Wang, S. Liu, J. Sun, J. Hao, X. Wang, L. Wang, J.S. Dong, D. Ting, There is limited correlation between coverage and robustness for deep neural networks, pp. 1–12 (2019). arXiv: 1911.05904

[EP20] T. Ergen, M. Pilanci, Convex geometry of two-layer ReLU networks: implicit autoencoding and interpretable models, in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)* (Virtual Conference, 2020), pp. 4024–4033

[GSS15] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in *Proceedings of the International Conference on Learning Representations (ICLR)* (San Diego, CA, USA, 2015), pp. 1–11

[HCWG+20] F. Harel-Canada, L. Wang, M.A. Gulzar, Q. Gu, M. Kim, Is neuron coverage a meaningful measure for testing deep neural networks? in *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE))* (Virtual Conference, 2020), pp. 851–862

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 770–778

[KBD+17] G. Katz, C. Barrett, D.L. Dill, K. Julian, M.J. Kochenderfer, Reluplex: an efficient SMT solver for verifying deep neural networks, in *Proceedings of the International Conference on Computer Aided Verification (CAV)* (Heidelberg, Germany, 2017), pp. 97–117

[KF19] P. Koopman, F. Fratrik, How many operational design domains, objects, and events? in *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI)* (Honolulu, HI, USA, 2019), pp. 1–8

[Kri09] A. Krizhevsky, *Object Classification Experiments* (Technical report, Canadian Institute for Advanced Research, 2009)

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. IEEE **86**(11), 2278–2324 (1998)

[MJXZ+18] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, J. Zhao, Y. Wang, DeepGauge: multi-granularity testing criteria for deep learning systems, in *Proceedings of the ACM/IEEE International Conference on Automated Software Engineering (ASE)* (Montpellier, France, 2018), pp. 120–131

[PCYJ19] Kexin Pei, Yinzhi Cao, Junfeng Yang, Suman Jana, DeepXplore: automated whitebox testing of deep learning systems. Commun. ACM **62**(11), 137–145 (2019)

[PGM+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, et al., PyTorch: an imperative style, high-performance deep learning library, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Vancouver, BC, Canada, 2019), pp. 8024–8035

[RJS+20] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, Paolo Tonella, Testing machine learning based systems: a systematic mapping. Empir. Softw. Eng. **25**(6), 5193–5254 (2020)

[SHK+19] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, Rob Ashmore, Structural test coverage criteria for deep neural networks. ACM Trans. Embed. Comput. Syst. (TECS) **18**(5s), 1–23 (2019)

[SZ15] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in *Proceedings of the International Conference on Learning Representations (ICLR)* (San Diego, CA, USA, 2015), pp. 1–14

[TPJR18] Y. Tian, K. Pei, S. Jana, B. Ray, DeepTest: automated testing of deep-neural-network-driven autonomous cars, in *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)* (2018), pp. 303–314

[ZHML20] J.M. Zhang, M. Harman, L. Ma, Y. Liu, Machine learning testing: survey, landscapes and horizons. IEEE Trans. Softw. Eng. **48**(01), 1–36 (2022). https://doi.org/10.1109/TSE.2019.2962027

Joint Optimization for DNN Model Compression and Corruption Robustness

Serin Varghese, Christoph Hümmer, Andreas Bär, Fabian Hüger, and Tim Fingscheidt

Abstract Modern deep neural networks (DNNs) are achieving state-of-the-art results due to their capability to learn a faithful representation of the data they are trained on. In this chapter, we address two insufficiencies of DNNs, namely, the lack of robustness to corruptions in the data, and the lack of real-time deployment capabilities, that need to be addressed to enable their safe and efficient deployment in real-time environments. We introduce hybrid corruption-robustness focused compression (HCRC), an approach that jointly optimizes a neural network for achieving network compression along with improvement in corruption robustness, such as noise and blurring artifacts that are commonly observed. For this study, we primarily consider the task of semantic segmentation for automated driving and focus on the interactions between robustness and compression of the network. HCRC improves the robustness of the `DeepLabv3+` network by 8.39% absolute mean performance under corruption (mPC) on the Cityscapes dataset, and by 2.93% absolute mPC on the Sim KI-A dataset, while generalizing even to augmentations not seen by the network in the training process. This is achieved with only minor degradations on undisturbed data. Our approach is evaluated over two strong compression ratios (30% and 50%) and consistently outperforms all considered baseline approaches. Additionally, we perform extensive ablation studies to further leverage and extend existing state-of-the-art methods.

S. Varghese (✉) · C. Hümmer · F. Hüger
Volkswagen AG, Berliner Ring 2, 38440 Wolfsburg, Germany
e-mail: john.serin.varghese@volkswagen.de

C. Hümmer
e-mail: christoph.heummer@volkswagen.de

F. Hüger
e-mail: fabian.hueger@volkswagen.de

A. Bär · T. Fingscheidt
Institute for Communications Technology (IfN), Technische Universität Braunschweig, Schleinitzstr. 22, 38106 Braunschweig, Germany
e-mail: andreas.baer@tu-bs.de

T. Fingscheidt
e-mail: t.fingscheidt@tu-bs.de

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_15

1 Introduction

Motivation: Image classification [KBK20], object detection [YXC20], machine translation [SVS19], and reading comprehension [ZHZ11] are just some of the tasks where deep neural networks (DNNs) excel at. They have proven to be an effective way to extract information from enormous amounts of data, and they are only expected to become more advanced over time. Despite their rapid progress, two insufficiencies of DNNs need to be addressed before deployment in real-time systems. First, in real-world applications, the edge devices on which these networks are deployed have limited capabilities in terms of the availability of memory and computational complexity (operations per second) that are required for neural network deployment. Second, the DNNs suffer from being not robust to even slight changes in the input (such as noise and weather conditions), which makes deployment in safety-critical applications challenging (Fig. 1).

Lack of DNN efficiency: To overcome the lack of efficiency, techniques such as pruning [HZS17, MTK+17, TKTH18], quantization [CLW+16, JKC+18, JGWD19] including quantization-aware trainings [GAGN15], knowledge distillation [HVD14], and encoding techniques [HMD16] are commonly used. All of these strategies seek to take advantage of the available redundancy in large DNNs to achieve run-time speedup.

Lack of DNN robustness: In addition to this insufficiency, recent studies [AMT18, HD19, BHSFs19] show that DNNs are not robust to even slight changes to the input image. These changes vary from carefully crafted perturbations called adversarial attacks [XLZ+18, DFY+20, BKV+20], to real-world augmentations such as snow, fog, additive noise, etc. [HD19]. The changes in the input image could vary from changes in just a few pixels [SVS19] to more global changes such as contrast and brightness [ZS18]. In the real world, such local or global changes are to be expected. For example, varying lighting conditions or foggy weather conditions can cause changes in the brightness and contrast of the input image.

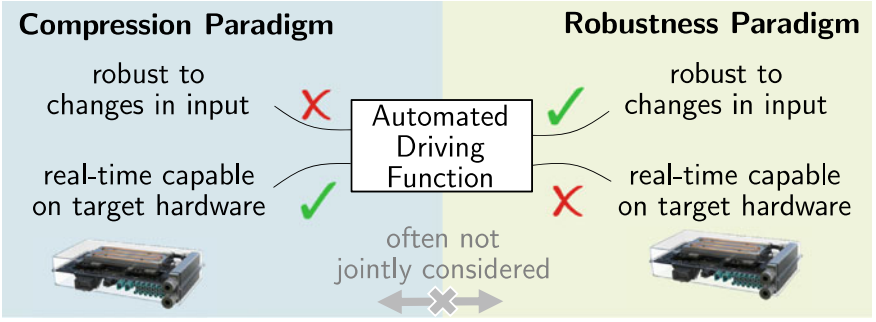


Fig. 1 Automated driving functions underly the two possibly opposing goals of compression and robustness. Approaches in the compression paradigm (*left*) are focused on enabling real-time efficient networks and rarely consider the effect of such a compression on the robustness properties of the network. Similarly, the robustness paradigm (*right*) typically does not consider real-time properties of the network

In this chapter, we tackle the insufficiencies that were mentioned above and introduce hybrid corruption-robustness focused compression (HCRC), an approach to jointly optimize a neural network for achieving network compression along with improvement in corruption robustness. By corruption, we refer to real-world augmentations, such as noise, blur, weather conditions, and digital effects, which are commonly occurring in the real world, and are therefore of significance. Our major contributions in this chapter are described below.

First, HCRC focuses on real-world corruption robustness and proposes a hybrid compression strategy, combining pruning and quantization approaches. We obtain a more robust and compressed network and also perform comparisons with sequential application of robustification and compression methods. Second, we approach the problem of robustness by training with augmentations in a controlled severity fashion. With our method, we show a further improvement under corruption (rPC) not only to the corruptions used during training but also to unseen corruptions including noise and blurring artifacts. Third, since all the methods discussed so far are only evaluated on small datasets for image classification, such as MNIST [LBBH98], CIFAR-10 [Kri09], and SVHN [NWC+11], there remains the question of their transferability to complex tasks, such as semantic segmentation [XWZ+17]. We, for the first time, perform such a study on two road-scenes datasets (Cityscapes [COR+16] and Sim KI-A) and a state-of-the-art semantic segmentation `DeepLabv3+` [CPK+18] network. sss

This chapter is structured as follows: In Sect. 3, we describe the individual components of such a system and our HCRC methodology in detail. In Sect. 4, we describe the corruptions that are used during training and evaluation, the datasets that are used, and the metrics used in the experiments. In Sect. 5, we present our experimental results and observations. Finally, in Sect. 6, we conclude our chapter.

2 Related Works

It is only recently that there have been studies to investigate the interaction between the two techniques, model compression and network robustness that tried to individually address the above-mentioned insufficiencies of DNNs.

Zhao et al. [ZSMA19] report one of the first investigations to empirically study the interactions between adversarial attacks and model compression. The authors observe that a smaller word length (in bits) for weights, and especially activations, makes it harder to attack the network. Building upon the alternating direction method of multipliers (ADMM) framework introduced by Ye et al. [YXL+19], Gui et al. [GWY+19] evaluated the variation of adversarial robustness (FGSM [GSS15] and also PGD [MMS+18]) with a combination of various compression techniques such as pruning, factorization, and quantization. In summary, so far we observe that network compression affects adversarial robustness, and a certain trade-off exists between them. The extent of the trade-off and the working mechanism behind it remains unsolved [WLX+20].

Some works have used compression techniques such as pruning and quantization that were traditionally used to obtain network compression, to improve the

robustness of networks. For example, Lin et al. [LGH19] use quantization not for acceleration of DNNs, but to control the error propagation phenomenon of adversarial attacks by quantizing the filter activations in each layer. On a similar line, Sehwag et al. [SWMJ20] propose to select the filters to be pruned by formulating an empirical minimization problem by incorporating adversarial training (using the PGD attack [MMS+18]) in each pruning step. Very recently, in addition to proposing the new evaluation criterion AER (that stands for accuracy, efficiency, and robustness) for evaluating the robustness and compressibility of networks, Xie et al. [XQXL20] describe a blind adversarial pruning strategy that combines adversarial training along with weight pruning.

In this chapter, we focus on real-world corruption robustness as opposed to the robustness to adversarial attacks. Additionally, we focus on the study of the interactions between robustness, quantization, and pruning methods within our proposed approach, supported by ablation studies.

3 HCRC: A Systematic Approach

Our goal is to improve the robustness of common image corruptions and at the same time reduce the memory footprint of semantic segmentation networks in a systematic way. In this section, we describe our systematic hybrid corruption-robustness focused compression (HCRC) approach to achieve compressed models that are also robust to commonly occurring image corruptions. Our proposed system can be broadly divided into two objectives: the *robustness objective* and the *compression objective*. Both will be described in the following subsections.

3.1 Preliminaries on Semantic Segmentation

We define $\mathbf{x} \in \mathbb{I}^{H \times W \times C}$ to be a clean image of the dataset $\mathcal{X}$, with the image height H, image width W, $C = 3$ color channels, and $\mathbb{I} = [0, 1]$. The image $\mathbf{x}$ is an input to a semantic segmentation network $\mathbf{F}(\mathbf{x}, \boldsymbol{\theta})$ with network parameters $\boldsymbol{\theta}$. Further, we refer to a network layer using an index $\ell \in \mathcal{L} = \{1, \ldots, L\}$, with $\mathcal{L}$ being the set of layer indices. Within layer ℓ we can define $\boldsymbol{\theta}_{\ell,k} \in \mathbb{R}^{H_\ell \times W_\ell}$ to be the kth kernel, where $k \in \mathcal{K}_\ell = \{1, \ldots, K_\ell\}$, with the set of kernel indices $\mathcal{K}_\ell$ of layer ℓ. The image input $\mathbf{x}$ is transformed to class scores by

$$\mathbf{y} = \mathbf{F}(\mathbf{x}, \boldsymbol{\theta}) \in \mathbb{I}^{H \times W \times S}. \tag{1}$$

Each element in $\mathbf{y} = (y_{i,s})$ is a posterior probability $y_{i,s}(\mathbf{x})$ for the class $s \in \mathcal{S} = \{1, 2, \ldots, S\}$ at the pixel position $i \in \mathcal{I} = \{1, \ldots, H \cdot W\}$ of the input image $\mathbf{x}$, and S denoting the number of semantic classes. A segmentation mask $\mathbf{m} = (m_i) \in \mathcal{S}^{H \times W}$ can be obtained from these posterior probabilities with elements

$$m_i = \underset{s \in \mathcal{S}}{\text{argmax}}\ y_{i,s}, \tag{2}$$

by assigning a class to each pixel i. The accuracy of the prediction is evaluated by comparing this obtained segmentation mask $\mathbf{m}$ against the labeled (ground truth) segmentation mask $\overline{\mathbf{m}} \in \overline{\mathcal{M}}$, that has the same dimensions as the segmentation mask $\mathbf{m}$. Likewise, $\overline{\mathbf{y}} \in \{0, 1\}^{H \times W \times S}$ is the one-hot encoded vector ground truth in three-dimensional tensor format that can be retrieved from $\overline{\mathbf{m}}$.

3.2 Robustness Objective

Data augmentation: In Fig. 2, the green data augmentation block on the left depicts the image pre-processing method following Hendryks et al. [HMC+20]. Here, the input image is augmented by mixing randomly sampled corruptions. The key idea is to introduce some amount of randomness in both, the type and the superposition of image corruptions. To achieve this, the input image is first split into three parts and then passed as an input to the data augmenter sub-blocks. Within a data augmenter sub-block, initially, a uniformly sampled corruption $\mathbf{A}_n \in \mathcal{A}^{\text{train}}$ is applied to the input. Here, $\mathcal{A}^{\text{train}} = \{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N\}$ denotes a set of N pre-defined corruption functions $\mathbf{A}_n()$ that are used during training. The corresponding corrupted image is computed as

$$\tilde{\mathbf{x}} = \mathbf{A}_n(\mathbf{x}, \Psi) \in \mathbb{I}^{H \times W \times C}, \tag{3}$$

where $\mathbf{A}_n(\mathbf{x}, \Psi)$ is the image corruption function and Ψ is a parameter controlling the strength of the applied augmentation. This random sampling and augmentation operation is repeated consequently $R = 4$ times within each of the data augmenter sub-blocks.

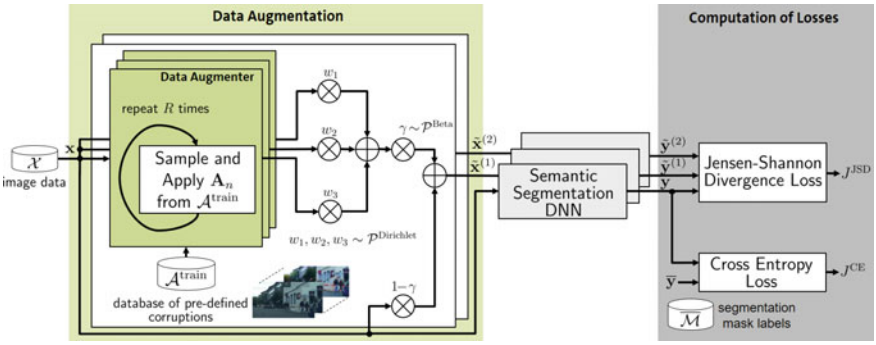


Fig. 2 Overview of the data augmentation strategy (left) and loss construction (right) for a semantic segmentation DNN

The output of each of the N data augmenter sub-blocks is, therefore, an augmented image

$$\tilde{\mathbf{x}}_n = \sum_{r=1}^{R} \tilde{\mathbf{x}}_n^{(r)} = \sum_{r=1}^{R} \mathbf{A}_n^{(r)}(\mathbf{x}, \Psi), \quad n \in \{1, 2, \ldots, N\}, \tag{4}$$

that is a combination of R applications of corruptions from $\mathcal{A}^{\text{train}}$. Choosing $N=3$, these outputs are first passed to multipliers with weights w_1, w_2, and w_3, which are sampled from a Dirichlet distribution $\mathcal{P}^{\text{Dirichlet}}$ and then added. Thereafter, the added output is multiplied by a factor γ that is sampled from a beta distribution $\mathcal{P}^{\text{Beta}}$ with parameters α=1 and β=1. Further, this is added to the input image $\mathbf{x}$, which is multiplied by a factor $1-\gamma$ to obtain the augmented image $\tilde{\mathbf{x}}^{(b)}$, with $b \in \mathcal{B} = \{1, 2, \ldots, B\}$ denoting the index among the B final augmented images being used in our proposed training method. Note that $B+1$ is our minibatch size, where one original image and B augmented images are being employed.

Construction of losses: In Fig. 2, the gray block on the right shows the strategy to construct losses from the predictions of the semantic segmentation network. Following (1), $\tilde{\mathbf{y}}^{(b)}$ denotes the class scores for an augmented input image $\tilde{\mathbf{x}}^{(b)}$. In addition to the aforementioned data augmentation strategy in the pre-processing stage, a loss function with an auxiliary loss term is introduced to enforce regularization between the responses of the semantic segmentation network to clean and augmented images in the training stage. The total loss is defined as

$$J = J^{\text{CE}} + \lambda J^{\text{JSD}}, \tag{5}$$

where J^{CE} is the cross-entropy loss and J^{JSD} is the auxiliary loss, also called the Jenson-Shannon divergence (JSD) loss [MS99]. The λ term is a hyper-parameter introduced to adjust the influence of J^{JSD} on the total loss J. The cross-entropy loss J^{CE} is computed between the posterior probabilities $\mathbf{y}$ of the network conditioned on input $\mathbf{x}$ and its corresponding labels $\overline{\mathbf{y}}$. It is defined as

$$J^{\text{CE}} = -\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{s \in \mathcal{S}} \alpha_s \overline{y}_{i,s} \cdot \log(y_{i,s}), \tag{6}$$

by taking a mean over all pixels for the posterior probability $\mathbf{y}$, where α_s are the weights assigned to each class during training, following [WSC+20]. The auxiliary loss, or the Jenson-Shannon Divergence (JSD) loss, is defined as

$$J^{\text{JSD}} = \frac{1}{B+1} \cdot (\text{KL}(\mathbf{y}, \mathring{\mathbf{y}}) + \sum_{b \in \mathcal{B}} \text{KL}(\tilde{\mathbf{y}}^{(b)}, \mathring{\mathbf{y}})). \tag{7}$$

It is computed between the posterior probabilities $\mathbf{y}$ and $\mathring{\mathbf{y}}$ or $\tilde{\mathbf{y}}^{(b)}$, where $b \in \mathcal{B} = \{1, \ldots, B\}$. Note that $\mathring{\mathbf{y}} = \frac{1}{B+1} \cdot (\mathbf{y} + \sum_{b \in \mathcal{B}} \tilde{\mathbf{y}}^{(b)})$ being the mixtures of the probabilities, and $\tilde{\mathbf{y}}^{(b)} = \mathbf{F}(\tilde{\mathbf{x}}^{(b)}, \boldsymbol{\theta})$. The auxiliary JSD loss is introduced to reduce the

variation in the probability distributions of the predictions between a clean input and an augmented input. To do this, two Kullback-Leibler (KL) divergence terms are introduced in (7), e.g.,

$$\mathrm{KL}(\tilde{\mathbf{y}}^{(b)}, \mathring{\mathbf{y}}) = \sum_{i \in \mathcal{I}} \tilde{\mathbf{y}}_i^{(b)} \log \left(\frac{\tilde{\mathbf{y}}_i^{(b)}}{\mathring{\mathbf{y}}_i} \right), \tag{8}$$

defining a distribution-wise measure of how one probability distribution (here: $\tilde{\mathbf{y}}^{(b)}$) differs from the reference mixture distribution (here: $\mathring{\mathbf{y}}_i$).

3.3 Compression Objective

Network pruning: We define a neural network as a particular parameterization of an architecture, i.e., $\mathbf{F}(\mathbf{x}, \boldsymbol{\theta})$ for specific parameters $\boldsymbol{\theta}$. Neural network pruning entails taking as input a model $\mathbf{F}(\mathbf{x}, \boldsymbol{\theta})$ and producing a new network $\mathbf{F}(\mathbf{x}, \mathbf{M} \odot \tilde{\boldsymbol{\theta}})$. Here $\tilde{\boldsymbol{\theta}}$ is the set of parameter values that may be different from $\boldsymbol{\theta}$, but both sets are of the same size $|\boldsymbol{\theta}| = |\tilde{\boldsymbol{\theta}}|$, and $\mathbf{M} \in \{0, 1\}^{|\tilde{\boldsymbol{\theta}}|}$ is a binary mask that forces certain parameters to be 0, while $\odot$ is the element-wise product operator. In practice, rather than using an explicit mask, pruned parameters of $\boldsymbol{\theta}$ are fixed to zero and are removed entirely.

We focus on producing a pruned network $\mathbf{F}(\mathbf{x}, \mathbf{M} \odot \tilde{\boldsymbol{\theta}})$ from a network $\mathbf{F}(\mathbf{x}, \boldsymbol{\theta}_0)$, where $\boldsymbol{\theta}_0$ is either sampled from an initialization distribution, or retrieved from a network pretrained on a particular task. Most neural network pruning strategies build upon [HPTD15], where each parameter or structural element in the network is issued a score, and the network is pruned based on these scores. Afterward, as pruning reduces the accuracy of the network, it is trained further (known as fine-tuning) to recover this lost accuracy. The process of pruning and fine-tuning is often iterated several times (iterative pruning) or performed only once (one-shot pruning).

In this chapter, we adopt the magnitude-based pruning approach [HPTD15] that is described in Algorithm 1. Although there exists a large body of more sophisticated scoring algorithms, the gain with such algorithms is marginal, if at all existing [MBKR18]. Based on the number of fine-tuning iterations F^{iter}, the number of filter weights to be pruned F^{pruned} (see Algorithm 1), the total number of prunable filter weights F^{total}, and the type of pruning (see Algorithm 1, Iterative Pruning), the function returns a sparser network $\boldsymbol{\theta}^{\text{pruned}}$ and the binary weight mask $\mathbf{M}$.

Algorithm 1 Magnitude-Based Filter Pruning and Fine-Tuning (Iterative Pruning)

Input: F^{iter}, the number of iterations of fine-tuning,
$\mathcal{X}^{\text{train}}$, the dataset to train and fine-tune,
F^{total}, the total number of prunable filter weights,
F^{pruned}, the number of filter weights to be pruned, and
`IterativePruning`, a boolean. If `true`: iterative pruning, if `false`: one-shot pruning.
$\boldsymbol{\theta} \leftarrow$ `initialize()` ▷ Random/ImageNet pretrained weights initialization
$\boldsymbol{\theta} \leftarrow$ `trainToConvergence`$(\mathbf{F}(\mathbf{x}, \boldsymbol{\theta}))$ ▷ Standard network training
$\mathbf{M} \leftarrow$ `rank`$(\boldsymbol{\theta}, F^{\text{pruned}})$ ▷ Filter weight rank computation (one-shot pruning)
for i in 1 to F^{iter} **do**
  if `IterativePruning` **then**
    $F^{\text{current}} \leftarrow 1 - (F^{\text{pruned}}/F^{\text{total}})^{i/F^{\text{iter}}}$ ▷ Adapt rule of pruning to iterative pruning
    $\mathbf{M} \leftarrow$ `rank`$(\boldsymbol{\theta}^{\text{pruned}}, F^{\text{current}})$ ▷ Updating $\mathbf{M}$
  end if
  $\boldsymbol{\theta}^{\text{pruned}} \leftarrow$ `fineTune`$(\mathbf{F}(\mathbf{x}, \mathbf{M} \odot \boldsymbol{\theta}))$ ▷ Network fine-tuning with sparsed weights
end for
Output: $\boldsymbol{\theta}^{\text{pruned}}$, the pruned network,
$\mathbf{M}$, the binary weight mask vector

Quantization: Low precision fixed-point representations replace floating-point number representations in fixed-point scalar quantization methods. Fixed-point scalar quantization operates on single weights $\theta_{\ell,k,j}$ of the network parameters, where the floating-point format weights are generally replaced by Q-bit fixed-point words [GAGN15], with the extreme case of binarization ($Q = 1$) [CBD15]. We focus on this uniform rounding scheme instead of other non-uniform schemes because it allows for fixed-point arithmetic with implementations in `PyTorch`. Quantization of network weights contributes to a large reduction in the model size and gives possibilities for acceleration on target hardware. In-training quantization (ITQ) refers to training a network by introducing quantization errors (12) in the network weights, and post-training quantization (PTQ) refers to quantizing the weights of a network after the training process by calibrating on the $\mathcal{X}^{\text{train}}$ and/or the $\mathcal{X}^{\text{val}}$ set. Figure 3 gives an overview of the in-training quantization (ITQ) and post-training quantization (PTQ) methods that are used in this chapter. Here, $\boldsymbol{\theta}$ corresponds to the neural network which is the input to the quantization methods, and $\boldsymbol{\theta}^{\text{quant}}$ refers to the fixed-point quantized codewords with lower precision. To do so, in the first block, the statistics for the scale factor

$$\rho_{\ell,k} = \frac{\max_j \theta_{\ell,k,j} - \min_j \theta_{\ell,k,j}}{2^Q - 1}, \tag{9}$$

which defines the spacing between bins, and the bias

$$\delta_{\ell,k} = \text{round}\left(\frac{\min_j \theta_{\ell,k,j}}{\rho_{\ell,k}}\right) \tag{10}$$

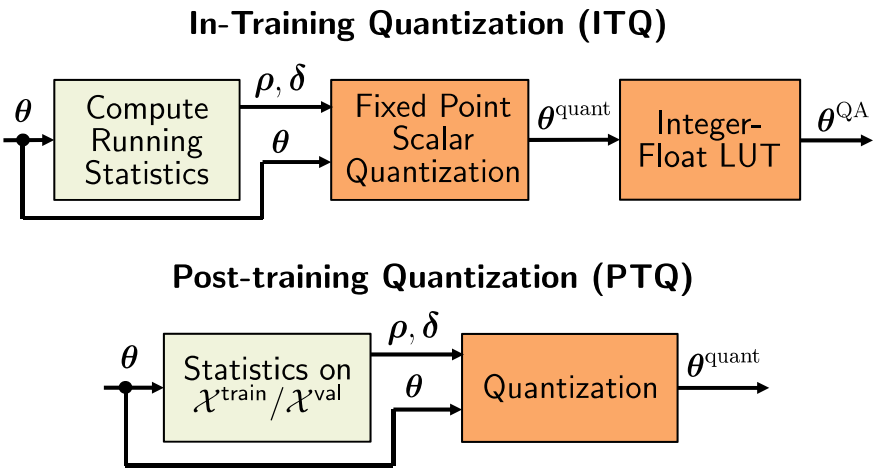


Fig. 3 Overview of quantization methods used in this work. Top: Within the in-training quantization (ITQ), for each iteration, based on the computed scale $\boldsymbol{\rho} = (\rho_{\ell,k})$ (9) and bias $\boldsymbol{\delta} = (\delta_{\ell,k})$ (10) terms, the network parameters $\boldsymbol{\theta}$ are first quantized to $\boldsymbol{\theta}^{\text{quant}}$ (11). Thereafter, each element in $\theta^{\text{quant}}_{\ell,k,j}$ is converted back to its floating-point representation based on a LUT (12). Bottom: Within the post-training quantization (PTQ), similar statistics $(\boldsymbol{\rho}, \boldsymbol{\delta})$ for a trained network are computed on the training and validation set and the network is quantized to $\boldsymbol{\theta}^{\text{quant}}$ (11)

by which the codewords are shifted, are computed. Here, $j \in \mathcal{J}_{\ell,k}$, with $\mathcal{J}_{\ell,k}$ is the set of parameter indices of kernel k in layer ℓ. Thereafter, for both ITQ and PTQ, each weight $\theta_{\ell,k,j}$ is mapped to its closest codeword $\theta^{\text{quant}}_{\ell,k,j}$ by quantizing $\theta_{\ell,k,j}$ using

$$\theta^{\text{quant}}_{\ell,k,j} = \min(\mathrm{q}^{\max}, \max(\mathrm{q}^{\min}, \mathrm{round}(\theta_{\ell,\mathrm{k},\mathrm{j}}/\rho_{\ell,\mathrm{k}} + \delta_{\ell,\mathrm{k}}))). \tag{11}$$

Here, $q^{\min}$ and $q^{\max}$ correspond to the minimum and maximum of the range of quantization levels depending on the chosen Q- bit quantization. For example, for an 8-bit quantization, $q^{\min} = 0$ and $q^{\max} = 255$. For ITQ training, the quantized parameters $\theta^{\text{quant}}_{\ell,k,j}$ are converted back to floating-point representation $\theta^{\text{QA}}_{\ell,k,j}$ based on an integer-float lookup table (LUT) following

$$\theta^{\text{QA}}_{\ell,k,j} = (\theta^{\text{quant}}_{\ell,k,i} - \delta_{\ell,k}) \cdot \rho_{\ell,k}. \tag{12}$$

This means that quantization errors are introduced within the network parameters $\boldsymbol{\theta}^{\text{QA}}$, which are then used within the training process. For PTQ, the quantized parameters $\boldsymbol{\theta}^{\text{quant}}$ are directly used in the evaluation of the semantic segmentation network.

In this chapter, we focus on the uniform rounding scheme instead of other non-uniform schemes, because it allows for fixed-point arithmetic with implementations in `PyTorch`. Throughout this chapter, we use a strong quantization of $Q = 8$ bits to enable higher acceleration on edge devices.

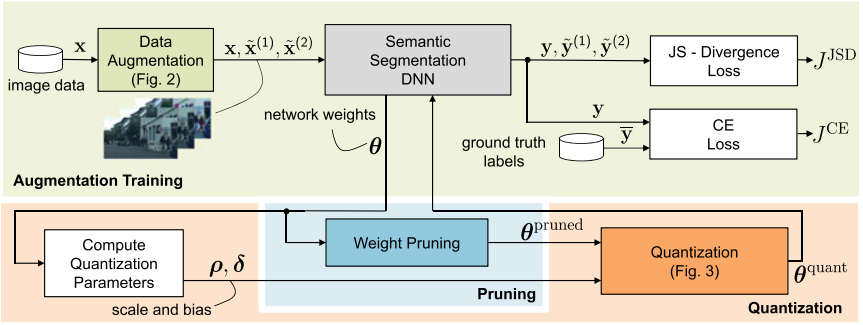


Fig. 4 Overview of our training strategy to co-optimize for corruption robustness along with network compression by the use of augmentation training, weight pruning, and quantization methods

3.4 HCRC Core Method

Within the HCRC framework, we systematically combine the robustness (Sect. 3.2), pruning, and quantization (Sect. 3.3) methods to co-optimize both robustness and compression objectives. Figure 4 gives an overview of our training strategy. The green block on the top depicts the augmentation strategy of the input image data and the consequent construction of losses. For each input image $\mathbf{x}$, the augmented images $\tilde{\mathbf{x}}^{(b)}$ are initially computed and passed to the semantic segmentation network. The total loss (5) is then computed based on the clean and corrupted image predictions. The orange block on the bottom depicts the quantization of the network weights and activations and the blue block contains the pruning module. We start by initializing the network parameters. In each training iteration, the scale factor (9) and the bias (10) are computed, and the network parameters are quantized (11). Additionally, in each training epoch, we use iterative pruning which continually prunes a certain percentage of the weights of the network (see blue block in Fig. 4).

4 Experimental Setup

In this section, the details of the semantic segmentation network, the road-scenes datasets, and the semantic segmentation networks that have been used in this chapter are initially described. The image corruptions that are applied in both phases, training and evaluation, are then introduced. Finally, the evaluation metrics are described.

4.1 Datasets and Semantic Segmentation Network

Our dataset splits are summarized in Table 1. For Cityscapes [COR+16], the baseline networks are trained with the 2,975 images of the training set $\mathcal{X}_{\text{CS}}^{\text{train}}$. Due to the Cityscapes test set upload restrictions, we split the official validation set into two sets-a mini validation set $\mathcal{X}_{\text{CS}}^{\text{val}}$ (Lindau, 59 images) and a mini test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ (Frankfurt and Münster, 441 images). The images have a resolution of 2,048×1,024. The Sim KI-A dataset is an artificially generated dataset with 4,257 training ($\mathcal{X}_{\text{Sim}}^{\text{train}}$), 387 validation ($\mathcal{X}_{\text{Sim}}^{\text{val}}$) and 387 test ($\mathcal{X}_{\text{Sim}}^{\text{test}}$) images. The images have a resolution of 1,920×1,080.

In this chapter, we use the `DeepLabv3+` [CBLR18] semantic segmentation network with `ResNet-101` backbone [HZRS16]. For both datasets, the baseline network, that is the network without any augmentation or compression, is trained with a crop size of 513 × 513 and a batch size of 4 on an `Nvidia Tesla V100` GPU. The class frequency-weighted cross-entropy loss J^{CE} (6) in combination with stochastic gradient descent (SGD) are used as optimization criterion and optimizer, respectively. During training, a polynomial learning rate scheme with an initial learning rate of 0.01 and a power of 0.9 is applied. The network is trained to convergence for 100 epochs on the Cityscapes dataset and 50 epochs for the Sim KI-A dataset. For a fair comparison, all the networks are evaluated on an `Intel(R) Xeon(R) Gold 6148` CPU.

4.2 Image Corruptions

The images corruptions used in this chapter are described in Table 2. These corruptions are split into two different categories depending on their usage, i.e., either

Table 1 Details of the road-scenes datasets used in the experiments. The image resolution of the dataset images and split into training, validation, and test sets are described

Dataset	Resolution	$\mathcal{X}^{\text{train}}$	$\mathcal{X}^{\text{val}}$	$\mathcal{X}^{\text{test}}$
Cityscapes [COR+16]	2,048 × 1,048	2,975	59	441
Sim KI-A	1,920 × 1,080	4,257	387	387

Table 2 Types of image corruptions used in this work that are arranged in two categories, based on their usage in either the training $\mathcal{A}^{\text{train}}$ or test $\mathcal{A}^{\text{test}}$ phases

Category	Corruption type
$\mathcal{A}^{\text{train}}$	Auto-contrast, equalize, posterize, color, sharpness, Gaussian blur, spatter, saturation
$\mathcal{A}^{\text{test}}$	Gaussian noise, shot noise, impulse noise, defocus blur, frosted glass blur, motion blur, zoom blur

Table 3 Corruptions and their parameterization used during training are listed. A dash (-) indicates that the corruption function is image-dependent and does not need any parameterization. An interval [a, b] indicates that the respective parameter is a real number $\mathbb{R}$ sampled uniformly from this interval

Corruption type	Auto-contrast	Equalize	Posterize	Color
Parameterization	–	–	[0, 4.0]	[0.25, 4.0]
Corruption type	Sharpness	Gaussian blur	Spatter	Saturation
Parameterization	[0.25, 4.0]	8	{0.65, 0.5, 0.3, 0.7, 0.65}	{1.5, 0.1}

during training or during test. The corruptions $\mathcal{A}^{\text{test}}$ in the test process are adopted from the neural network robustness benchmark[1] from [HD19]. The corruptions $\mathcal{A}^{\text{train}}$ used in the training process are adopted following a large body of work [HMC+20, CZM+19, TPL+19, SK19] that use these corruptions in different ways for training with data augmentation. Table 3 gives an overview of the parameterization of each corruption within $\mathcal{A}^{\text{train}}$. For spatter corruption, the list of parameters corresponds to the location, scale, two sigma, and threshold values, respectively. For saturation corruption, the list of parameters corresponds to the amount of saturation and the scale. For posterize, color, and sharpness corruptions, the parameter is sampled from within the given interval.

Definition of severities: Various kinds of data augmentations exist, and it is rather difficult to compare between different augmentation types, although first attempts are known [KBFs20]. Let us take an example of brightness and contrast augmentations. We can increase or decrease the brightness and contrast values for a given input image by manipulating the image pixel values. An increase in the brightness and an increase in the contrast do not necessarily correspond to the same effect on the input image. To standardize the method of measuring the strength of augmentations irrespective of the augmentation type, the structural similarity (SSIM) metric [WBSS04] is used. To do this, SSIM is computed between the clean input image $\mathbf{x}$ and the augmented image $\tilde{\mathbf{x}}^{(b)}$. Here, $\text{SSIM}(\mathbf{x}, \tilde{\mathbf{x}}^{(b)}) = 0$ indicates that the image $\mathbf{x}$ and the corresponding augmented image $\tilde{\mathbf{x}}^{(b)}$ are completely dissimilar. Similarly, $\text{SSIM}(\mathbf{x}, \tilde{\mathbf{x}}^{(b)}) = 1$ indicates that the image $\mathbf{x}$ and the corresponding augmented image $\tilde{\mathbf{x}}^{(b)}$ are identical, or no augmentation is applied. We define severity levels (V) to indicate the strength of the augmentation. Severity level $V = 0$ indicates that no type of augmentation is applied to the input image and severity level $V = 10$ indicates that the input image is completely dissimilar after the augmentation. This means that for every increase in level in V, the SSIM between the clean input image and the augmented image reduces by 0.1. To control the severity of the data augmentation during training, the parameters (α, β) of the γ-function are varied (see Fig. 2) by keeping the corruption parameters constant following Table 2.

[1] https://github.com/hendrycks/robustness.

4.3 Metrics

Mean intersection-over-union (mIoU) between the predictions of the semantic segmentation network and the human-annotated ground truth labels is commonly used for evaluating semantic segmentation networks. The mIoU is defined as

$$\text{mIoU} = \frac{1}{S} \sum_{s \in \mathcal{S}} \frac{\text{TP(s)}}{\text{TP(s)} + \text{FP(s)} + \text{FN(s)}} = \text{mIoU}(\mathbf{y}, \bar{\mathbf{y}}), \tag{13}$$

where TP(s), FP(s), and FN(s) are the class-specific true positives, false positives, and false negatives, respectively, computed between segmentation output $\mathbf{y}$ and ground truth one-hot encoded segmentation $\bar{\mathbf{y}}$.

Mean performance under corruption (mPC) has been introduced by [HD19] for evaluating the robustness of neural networks under varying corruptions and varying strengths. For this purpose, the individual augmentations $\mathbf{A}_n \in \mathcal{A}^{\text{test}}$ are further sub-divided with respect to the strength of the augmentations. We use the augmentations $\mathcal{A}^{\text{test}}$ (see Table 2) for the computation of mPC. This is computed by

$$\text{mPC} = \frac{1}{|\mathcal{A}^{\text{test}}|} \sum_{c=1}^{|\mathcal{A}^{\text{test}}|} \frac{1}{N_c} \sum_{V=1}^{N_c} \text{mIoU}_{c,V}, \tag{14}$$

with corruption index c and N_c denoting the amount of severity conditions for a corruption c. Here, $\text{mIoU}_{c,V}$ denotes the mIoU (13) of the model under the corruption c and severity V. The key factor here is choosing the severities, as this can vary on different datasets, and even models, depending on the selection criteria. In this chapter, we use the SSIM metric [HD19] as a means of finding severity thresholds. Using this metric allows for standardized severities across a dataset, as it is task- and model-agnostic. Thus, different robustness improvement methods can be benchmarked and compared easily using the mPC metric.

Relative performance under corruption (rPC) is simply the ratio of the mPC and the mIoU of the semantic segmentation under the corruptions during evaluation, and is defined as

$$\text{rPC} = \frac{\text{mPC}}{\text{mIoU}}. \tag{15}$$

4.4 Training Framework

For the task of achieving robust and compressed semantic segmentation networks, one can envision various different ways to approach it. An overview of all possible approaches is given in Fig. 5. For all the reference models, we start from the pre-trained checkpoint weights of the `ResNet-101` backbone for the `DeepLabv3+` architecture.

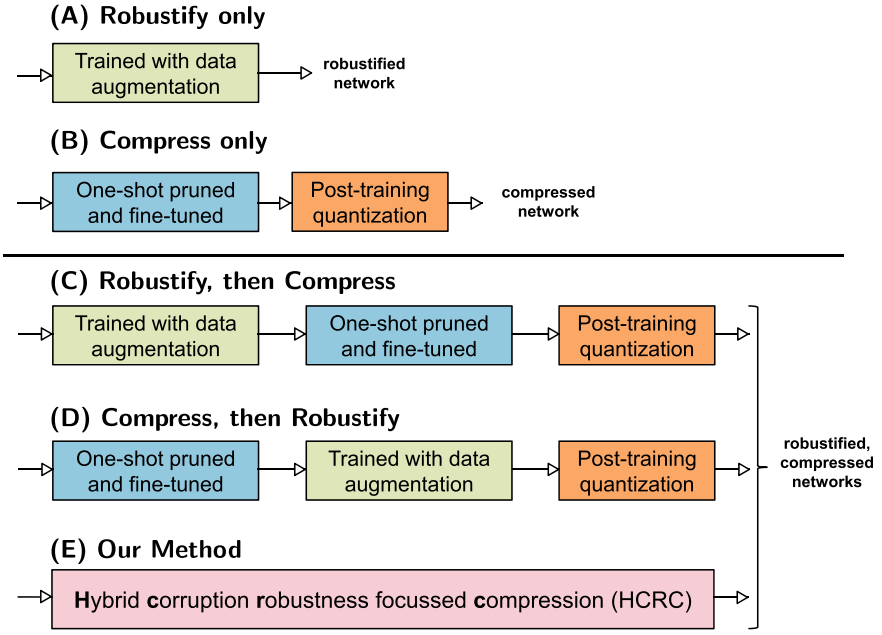


Fig. 5 The training approaches used in this work are depicted. In addition to the simple baselines (Reference A, and Reference B), we compare our HCRC approach against sequential applications (Reference C, and Reference D) of the individual steps in the training framework

Reference A: In this approach, the `DeepLabv3+` network with the `ResNet-101` backbone is trained for improving its corruption robustness. Here, no compression techniques are applied. The network is trained using the protocol defined in Sect. 4.1 with the total loss (5) and $\lambda = 10^{-6}$.

Reference B: Here, the `DeepLabv3+` undergoes one-shot pruning (see Algorithm 1). First, the network is trained using the protocol defined in Sect. 4.1 with the class frequency weighted cross-entropy loss (6). Next, the statistics (9), (10) are computed on the $\mathcal{X}^{\text{train}}$ and $\mathcal{X}^{\text{val}}$ set and the network undergoes PTQ (see Fig. 3). No robustness-related training is enforced.

Reference C: In this configuration, we perform sequential application of the robustness and compression goals. In the first step, the `DeepLabv3+` is trained using the protocol defined in Sect. 4.1 with the total loss (5) and $\lambda = 10^{-6}$ (see also Reference A) in combination with the data augmentation strategy described in Sect. 3.2 (see Fig. 2). Next, the network undergoes iterative pruning (see Algorithm 1) following again the protocol defined in Sect. 4.1, this time with the class frequency weighted cross-entropy loss (6). Finally, statistics (9), (10) are computed on the $\mathcal{X}^{\text{train}}$ and $\mathcal{X}^{\text{val}}$ set and the network undergoes PTQ (see Fig. 3).

Reference D: In this setup, the `DeepLabv3+`, the network is first one-shot pruned (see Algorithm 1) and then fine-tuned following the protocol defined in Sect. 4.1 using

the class frequency weighted cross-entropy loss (6). In the next step, the network is trained with the data augmentation strategy described in Sect. 3.2 (see Fig. 2) following again the protocol defined in Sect. 4.1, however, this time the total loss J (5) is used as the optimization criterion. Finally, statistics (9, 10) are computed on the $\mathcal{X}^{\text{train}}$ and $\mathcal{X}^{\text{val}}$ set and the network undergoes PTQ (see Fig. 3).

5 Experimental Results and Discussion

5.1 Ablation Studies

In-training quantization (ITQ) vs. post-training quantization (PTQ) We hypothesize that training the network with quantization errors is better than quantizing the network after training.

In Table 4, we compare these two approaches of achieving quantized networks. The baseline `DeepLabv3+` network has an mIoU of 69.78% and mPC of 44.03%. On one hand, we observe that mIoU drops by 5.35% and mPC by 2.76% (both: absolute) after PTQ. On the other hand, the drop in mIoU is only 1.8% after ITQ, within no change in mPC. This result supports the abovementioned hypothesis on quantization, that in-training quantization is superior to post-training quantization. Additionally, we increased the size of the calibration set used within PTQ by also including $\mathcal{X}^{\text{val}}$ along with $\mathcal{X}^{\text{train}}$. This, however, resulted in no significant changes in the performance of the quantized networks.

Controlled severity training: The semantic segmentation network is evaluated over various corruptions and various severity levels. From our initial experiments, we observed that the data corruptions used during the training process [HMC+20] have a mean severity level of $V = 1$. It is intuitive that a network trained on data augmentations of higher severity should be, in theory, more robust to higher severity corruptions during test. To study the effect of the training severity on the robustness of the trained semantic segmentation network, we train the `DeepLabv3+` network with three different severities of corruption. To do this, we vary the parameters of the

Table 4 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation of mIoU, mPC, and rPC comparing the non-quantized `DeepLabv3+` network, and the corresponding PTQ and ITQ networks that are trained to convergence for 100 epochs. Note that the inference times are computed on the `Intel(R) Xeon(R) Gold 6148` CPU. Best numbers reported in **bold**

Method	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)	Time (s)
`DeepLabv3+`	69.78	44.03	63.09	5.23
with PTQ	64.43	41.27	64.05	**2.66**
with ITQ	**67.98**	**44.04**	**64.78**	**2.66**

Table 5 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation based on mIoU, mPC, and rPC comparing the three DeepLabv3+ networks trained on augmentations of three different severity levels. Note that the networks are not subjected to any kind of compression. Best numbers reported in **bold**

Method	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)
DeepLabv3+	69.78	44.03	63.09
Trained with severity level $V = 1$	**69.98**	53.65	76.66
Trained with severity level $V = 2$	69.54	56.14	80.73
Trained with severity level $V = 3$	69.44	**56.27**	**81.03**

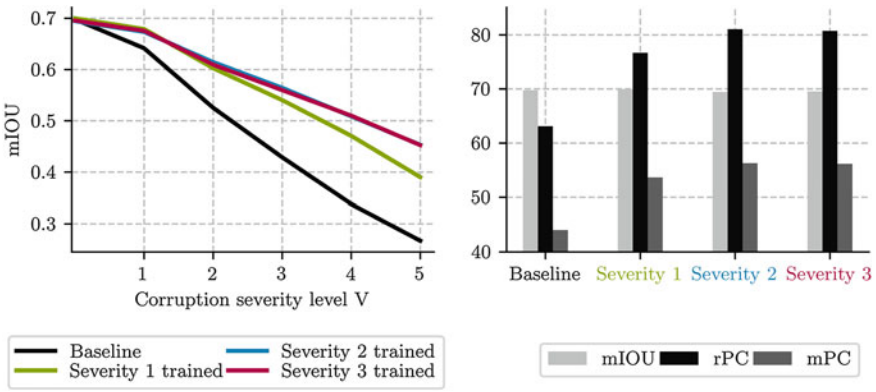


Fig. 6 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation for the DeepLabv3+ network trained with different corruption severities. Left: The four networks are evaluated over the augmentations $\mathcal{A}^{\text{test}}$ with six severity levels (x-axis, severities $V = 0, 1, \ldots, 5$). Right: The bar chart shows the mIoU on clean data ($V = 0$), as well as mPC and rPC, computed over the same six severity levels ($V = 0, 1, \ldots, 5$), for four networks trained with severity $V = 0$ (baseline), 1, 2, and 3

beta distribution to increase the influence of the individual corruptions. The results are shown in Table 5.

We generally observe that training with higher severities leads to higher robustness in terms of the mPC. The DeepLabv3+ network trained with a severity level of 3 has an increase of 2.62% absolute mPC and 4.36% absolute rPC when evaluated on $\mathcal{X}_{\text{CS}}^{\text{test}}$. In Fig. 6, we show the results of evaluating these networks on six different severity values. The networks trained with higher severities show higher robustness, especially when evaluated on higher severities ($V \geq 3$). Training with a higher severity ($V \geq 4$) did not show any further improvements. A drop in the mIoU indicates a certain trade-off between an increase in the generalization (to unseen corruptions) capability of the network to a decrease in its performance on the clean (or vanilla) input.

Table 6 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation based on mIoU, mPC, and rPC comparing the one-shot and iterative pruning approaches for the `DeepLabv3+` network. All the networks are trained with augmentations of severity level 2. A $Q = 8$ bits quantization is applied to all the HCRC trainings. Best numbers reported in **bold**

Method	Pruning Ratio [%]	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)
`DeepLabv3+`	0	69.78	44.03	63.09
HCRC with one-shot pruning	30	66.06	50.80	76.89
HCRC with iterative pruning	30	**68.12**	**52.50**	**77.07**
HCRC with one-shot pruning	50	64.58	49.45	76.57
HCRC with iterative pruning	50	**66.84**	**51.95**	**77.72**

Sensitivity of the pruning algorithm: We perform an ablation study to analyze the effect of the types of pruning methodology within our HCRC approach. To this end, we train the `DeepLabv3+` network in a combined fashion (see Sect. 3.4) with two different types of pruning, namely, one-shot pruning and iterative pruning. In Table 6, we provide the evaluation results of this study over two different pruning ratios (30% and 50%). A pruning ratio of 30% indicates that 30% of the prunable weights are removed from the network, while 70% are remaining. For quantization, within all our experiments, we have used a strong quantization of $Q = 8$ bits.

For 30% pruning ratio, we observe that the iterative pruning method shows an (absolute) increase in mIoU (2.06%), mPC (1.7%), and rPC (0.18%), when compared to the one-shot pruning method, evaluated on $\mathcal{X}_{\text{CS}}^{\text{test}}$. For 50% pruning ratio, we observe similar (absolute) improvements for iterative pruning in its mIoU (2.26%), mPC (2.5%), and rPC (1.15%), computed over $\mathcal{X}_{\text{CS}}^{\text{test}}$.

5.2 Comparison With Reference Baselines

In this section, we compare our HCRC method (with iterative pruning) with the reference methods (see Sect. 4.4). In particular, we compare our HCRC method against Reference C and Reference D, which also aim to achieve robust and compressed segmentation networks.

For the Cityscapes dataset, the results of the evaluation are shown in Table 7. We observe that our HCRC method outperforms all the relevant reference methods for both pruning ratios. The reference A network with a pruning ratio of 0% shows an improvement of 11.62% absolute mPC over the `DeepLabv3+` baseline network with a slight improvement in the mIoU. For 30% pruning, the HCRC shows significant improvements over the reference methods B, C, and D. The HCRC method shows an improvement of 3.29% absolute mIoU and 9.14% absolute mPC over the best reference (Reference D). Additionally, HCRC improves the robustness of the `DeepLabv3+` network by 8.47% absolute mPC with a 77.67% reduction in the

Table 7 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation comparing HCRC to reference methods A–D. A quantization with $Q = 8$ bits quantization is applied to all trainings where compression is applied. Best numbers reported in **bold**

Pruning Ratio [%]	Method	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)	Model Size (MB)
0	`DeepLabv3+`	69.78	44.03	63.10	237.38
	Reference A	**69.98**	**53.65**	**76.67**	237.38
30	Reference B	62.56	35.79	57.21	52.33
	Reference C	64.28	37.13	57.76	52.33
	Reference D	64.83	43.36	66.88	52.33
	HCRC (ours)	**68.12**	**52.50**	**77.07**	52.33
50	Reference B	64.48	37.50	58.16	47.47
	Reference C	66.05	36.52	55.29	47.47
	Reference D	66.03	48.52	73.48	47.47
	HCRC (ours)	**66.16**	**51.95**	**77.72**	47.47

model size. For 50% pruning ratio, we observe similar improvements in HCRC over the reference methods B, C, and D. The HCRC method shows an improvement in mIoU (2.13%) and mPC (2.91%) when evaluated on $\mathcal{X}_{\text{CS}}^{\text{test}}$ and when compared to the best reference (reference D). Overall, HCRC with pruning ratio of 50% improves the robustness of the `DeepLabv3+` network by 7.92% absolute mPC with an almost 80% reduction in the model size.

For the Sim KI-A dataset, we similarly observe that our HCRC method outperforms all the relevant reference baselines for both the pruning ratios, see Table 8. The reference A network with a pruning ratio of 0% shows an improvement of 12.98% absolute mPC over the `DeepLabv3+` baseline network with a slight improvement in the mIoU. For 30% pruning, the HCRC shows significant improvements over the reference methods B, C, and D. The HCRC method shows an improvement of 2.33% absolute mIoU and 5.09% absolute mPC over the best reference (Reference D). For 50% pruning ratio, we observe similar improvements in HCRC over the reference methods B, C, and D. The HCRC method shows an improvement in mIoU (1.04%) and mPC (3.68%) when evaluated on $\mathcal{X}_{\text{Sim}}^{\text{test}}$ and when compared to the best reference (reference D). Overall, HCRC with pruning ratio of 50% improves the robustness of the `DeepLabv3+` network by 7.60% absolute mPC with an almost 80% reduction in the model size.

Interestingly, the clean performance of our compressed HCRC network is nearly the same as the uncompressed `DeepLabv3+` baseline, albeit with much improved robustness. We also show qualitative results in Fig. 7 for impulse noise of severity level $V = 3$, where we observe a significant improvement over the simpler reference B baseline. In summary, our proposed HCRC approach to co-optimize for corruption robustness and model compression outperforms all possible reference baselines and produces a network that is heavily compressed and robust to unseen and commonly occurring image corruptions.

Table 8 Test set $\mathcal{X}_{\text{Sim}}^{\text{test}}$ evaluation comparing HCRC to reference methods A–D. A quantization with $Q = 8$ bits is applied to all trainings where compression is applied. Best numbers reported in **bold**

Pruning Ratio [%]	Method	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)	Model Size (MB)
0	`DeepLabv3+`	**77.57**	54.42	70.16	237.38
	Reference A	77.05	**67.40**	**87.48**	237.38
30	Reference B	74.19	49.78	67.10	52.33
	Reference C	72.44	51.56	71.17	52.33
	Reference D	73.90	58.70	79.43	52.33
	HCRC (ours)	**76.23**	**63.79**	**83.68**	52.33
50	Reference B	75.65	47.60	62.92	47.47
	Reference C	75.08	47.82	63.69	47.47
	Reference D	74.38	58.34	78.43	47.47
	HCRC (ours)	**76.12**	**62.02**	**81.48**	47.47

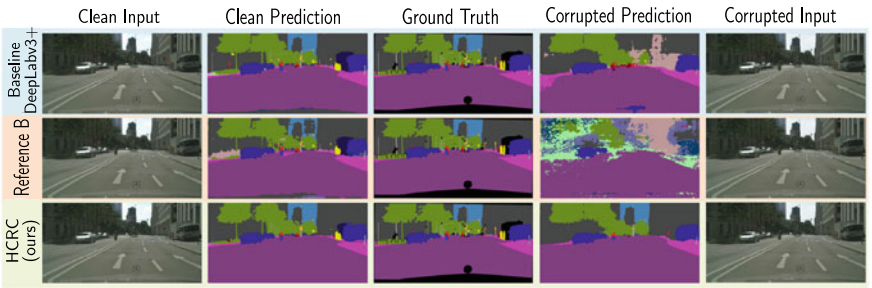


Fig. 7 Example segmentations on the Cityscapes dataset. We show a snippet from $\mathcal{X}_{\text{CS}}^{\text{test}}$, where the differences in the robustness of the compressed networks are more pronounced. We observe that our HCRC method is compressed and has superior robustness to the `DeepLabv3+` baseline and the compressed network of reference B, in this example, for impulse noise corruption

6 Conclusions

In this chapter, we introduce hybrid corruption-robustness focused compression (HCRC), an approach to jointly optimize a neural network for achieving network compression along with improvement in corruption robustness, such as noise and blurring artifacts, which are commonly observed. For this study, we consider the task of semantic segmentation for automated driving and look at the interactions between robustness and compression of networks. HCRC improves the robustness of the `DeepLabv3+` network by 8.47% absolute mean performance under corruption (mPC) on the Cityscapes dataset and 7.60% absolute mPC on the Sim KI-A dataset and generalizes even to augmentations not seen by the network in the training process. This is achieved with only minor degradations on undisturbed data.

Our approach is evaluated over two strong compression ratios and consistently outperforms all considered baseline approaches. Additionally, we perform extensive ablation studies to further leverage and extend existing state-of-the-art methods.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)". The authors would like to thank the consortium for the successful cooperation.

References

[AMT18] A. Arnab, O. Miksik, P.H.S. Torr, On the robustness of semantic segmentation models to adversarial attacks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT, USA, 2018), pp. 888–897

[BHSFs19] A. Bär, F. Hüger, P. Schlicht, T. Fingscheidt, On the robustness of redundant teacher-student frameworks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (Long Beach, CA, USA, 2019), pp. 1380–1388

[BKV+20] A. Bär, M. Klingner, S. Varghese, F. Hüger, P. Schlicht, T. Fingscheidt, Robust semantic segmentation by redundant networks with a layer-specific loss contribution and majority vote, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, virtual conference (2020), pp. 1348–1358

[CBD15] M. Courbariaux, Y. Bengio, J-P. David, Binary connect: training deep neural networks with binary weights during propagations, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, 2015), pp. 3123–3133

[CBLR18] Z. Chen, V. Badrinarayanan, C-Y. Lee, A. Rabinovich, GradNorm: gradient normalization for adaptive loss balancing in deep multitask networks, in *Proceedings of the International Conference on Machine Learning (ICML)* (Stockholm, Sweden, 2018), pp. 794–803

[CLW+16] Y. Cao, M. Long, J. Wang, H. Zhu, Q. Wen, Deep quantization network for efficient image retrieval, in *Proceedings of the AAAI Conference on Artificial Intelligence* (Phoenix, AZ, USA, 2016), pp. 3457–3463

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The Cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 3213–3223

[CPK+18] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **40**(4), 834–848 (2018)

[CZM+19] E.D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q.V. Le, AutoAugment: learning augmentation strategies from data, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, 2019), pp. 113–123

[DFY+20] Y. Dong, Q-A. Fu, X. Yang, T. Pang, H. Su, Z. Xiao, J. Zhu, Benchmarking adversarial robustness on image classification, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2020), pp. 1322–1330

[GAGN15] S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan, Deep learning with limited numerical precision, in *Proceedings of the International Conference on Machine Learning (ICML)* (Lille, France, 2015), pp. 1737–1746

[GSS15] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in *Proceedings of the International Conference on Learning Representations (ICLR)* (San Diego, CA, USA, 2015), pp. 1–11

[GWY+19] S. Gui, H. Wang, H. Yang, C. Yu, Z. Wang, J. Liu, Model compression with adversarial robustness: a unified optimization framework, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Vancouver, BC, Canada, 2019), pp. 1283–1294

[HD19] D. Hendrycks, T. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–15

[HMC+20] D. Hendrycks, N. Mu, E.D. Cubuk, B. Zoph, J. Gilmer, B. Lakshminarayanan, Augmix: a simple data processing method to improve robustness and uncertainty, in *Proceedings of the International Conference on Learning Representations (ICLR)*, virtual conference (2020), pp. 1–15

[HMD16] S. Han, H. Mao, W.J. Dally, Deep compression: compressing deep neural network with pruning, trained quantization and Huffman coding, in *proceedings of the international conference on learning representations (ICLR)* (2016), pp. 1–14

[HPTD15] S. Han, J. Pool, J. Tran, W.J. Dally, Learning both weights and connections for efficient neural networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, 2015), pp. 1135–1143

[HVD14] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops (Montréal, QC, Canada, 2014)*, pp. 1–9

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 770–778,

[HZS17] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Venice, Italy, 2017), pp. 1398–1406

[JGWD19] S. Jain, A. Gural, Mi. Wu, C. Dick, Trained uniform quantization for accurate and efficient neural network inference on fixed-point hardware (2019), pp. 1–17, arXiv:1903.08066

[JKC+18] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A.G. Howard, H. Adam, D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT, USA, 2018), pp. 2704–2713

[KBFs20] M. Klingner, A. Bär, T. Fingscheidt, Improved noise and attack robustness for semantic segmentation by using multi-task training with self-supervised depth estimation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* , virtual conference (2020), pp. 1299–1309

[KBK20] I. Kim, W. Baek, S. Kim, Spatially attentive output layer for image classification, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2020), pp. 9533–9542

[Kri09] A. Krizhevsky, *Object Classification Experiments* (Technical report, Canadian Institute for Advanced Research, April 2009)

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. IEEE **86**(11), 2278–2324 (1998)

[LGH19] J. Lin, C.Gan, S. Han, Defensive quantization: when efficiency meets robustness, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–15

[MBKR18] D. Mittal, S. Bhardwaj, M. Khapra, B. Ravindran, Recovering from random pruning: on the plasticity of deep convolutional neural networks, in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)* (Lake Tahoe, NV, USA, 2018), pp. 848–857

[MMS+18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Vancouver, BC, Canada, 2018), pp. 1–10

[MS99] D. Christopher, *Manning and Hinrich Schütze* (MIT Press, Foundations of Statistical Natural Language Processing, 1999)

[MTK+17] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Toulon, France, 2017), pp. 1–17

[NWC+11] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops* (Granada, Spain, 2011), pp. 1–9

[SK19] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning. J. Big Data **60**(6), 1–48 (2019)

[SVS19] J. Su, D.V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks. IEEE Trans. Evolut. Comput. (TEVC) **23**(5), 828–841 (2019)

[SWMJ20] V. Sehwag, S. Wang, P. Mittal, S. Jana, HYDRA: pruning adversarially robust neural networks (2020), pp. 1–22. arXiv:2002.10509

[TKTH18] L. Theis, I. Korshunova, A. Tejani, F. Huszár, Faster Gaze Prediction With Dense Networks and Fisher Pruning (2018), pp. 1–18. arXiv:1801.05787

[TPL+19] Z. Tang, X. Peng, T. Li, Y. Zhu, D. Metaxas, Adatrans form: adaptive data transformation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, 2019), pp. 2998–3006

[WBSS04] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)

[WLX+20] S. Wang, N. Liao, L. Xiang, N. Ye, Q. Zhang, Achieving Adversarial Robustness via Sparsity (2020), pp. 1–9, arXiv:2009.05423

[WSC+20] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, M. Yadong, M. Tan, X. Wang et al., Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **43**(10), 3349–3364 (2020)

[XLZ+18] C. Xiao, B. Li, J-Y. Zhu, W. He, M. Liu, D. Song, Generating adversarial examples with adversarial networks (2018), pp. 1–8, arXiv:1801.02610

[XQXL20] H. Xie, L. Qian, X.g Xiang, N. Liu, Blind adversarial pruning: balance accuracy, efficiency and robustness (2020), pp. 1–12. arXiv: 2004.05913

[XWZ+17] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, A. Yuille, Adversarial examples for semantic segmentation and object detection, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Venice, Italy, 2017), pp. 1369–1378

[YXC20] M. Ye, S. Xu, T. Cao, HVNet: hybrid voxel network for LiDAR based 3d object detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2020), pp. 1631–1640

[YXL+19] S. Ye, K. Xu, S. Liu, H. Cheng, J.H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, X. Lin, Adversarial robustness vs. model compression, or both? in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, October 2019), pp. 111–120

[ZHZ11] M. Zhou, M. Huang, X. Zhu, Robust reading comprehension with linguistic constraints via posterior regularization. IEEE/ACM Trans. Audio, Speech, Lang. Process. **20**(4), 1096–1108 (2011)

[ZS18] Z. Zhang, M.R. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, 2018), pp. 8792–8802

[ZSMA19] Y. Zhao, I. Shumailov, R. Mullins, R. Anderson, To compress or not to compress: understanding the interactions between adversarial attacks and neural network compression, in *Proceedings of the Conference on Machine Learning and Systems (MLSys)* (Stanford, CA, USA, 2019), pp. 230–240