ROBUST AND INTERPRETABLE VISUAL PERCEPTION USING DEEP NEURAL NETWORKS

DISSERTATION
zur Erlangung des Doktorgrades (Dr. rer. nat.)
der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

JÖRG WAGNER

aus Simmern (Hunsrück)

Bonn, 2022

Angefertigt mit Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Sven Behnke
2. Gutachter:	Prof. Dr. Jürgen Gall

Tag der Promotion:	18.11.2022
Erscheinungsjahr:	2023

Abstract

Autonomous vehicles promise to revolutionize the transportation of people and goods by increasing road safety, reducing resource consumption, and improving quality of life. To achieve an unrestricted and large-scale deployment in the real world without any human supervision, many challenges still need to be solved. A key challenge is the robust perception and interpretation of the surroundings. Deep learning-based approaches have significantly advanced the creation of robust environment representations in recent years. However, further improvements are required, for example, to cope with difficult environment conditions (adverse weather, low lighting conditions, ...). In the first part of this thesis, we investigate approaches to improve the robustness of vision-based perception models. One promising approach is to fuse data of multiple complementary sensors. Building on previous deep learning-based pedestrian detectors operating on visible images, we develop a multispectral detector. Our detector combines the data of a visible and a thermal camera using a deep fusion network and provides significantly better results than comparable single sensor models. To the best of our knowledge, this is the first work to use a deep learning-based approach for multispectral pedestrian detection. A complementary method for improving perception performance is the temporal filtering of information. The filtering task can be divided into a prediction and an update step. Initially, we explore the prediction step and propose an approach for generating semantic forecasting models by transforming trained non-predictive feed-forward networks. The predictive transformation is based on a structural extension of the network using a recurrent predictive module and a teacher-student training strategy. The resulting semantic forecasting architecture models the dynamics of the scene, enabling meaningful predictions. Building on the knowledge gained, we design a parameter efficient approach to temporally filter the representations of *Fully Convolutional DenseNet* (FC-DenseNet) in a hierarchical manner. Based on a simulated dataset with significant disturbances (*e.g.* noise, occlusions, missing information), we show the advantages of temporal filtering and compare our architecture with similar temporal networks.

A disadvantage of many modern perception models is their black-box character. Especially in safety-critical applications, a high degree of transparency and interpretability is beneficial, as it facilitates model development, debugging, failure analysis, and validation. In the second part of this thesis, we study two approaches to increase transparency and interpretability of deep learning-based models. First, we consider the concept of *interpretability by design* and propose a modular and interpretable representation filter which divides the filtering task into multiple, less complex subtasks. Additional insights into its functioning are gained by introducing intermediate representations which are interpretable to humans. These representations also enable the integration of domain knowledge. Using our proposed filter, we increase the robustness of a semantic segmentation model. As an alternative to designing more interpretable architectures, post-hoc explanation methods can be used to gain insights into the decision-making process of a model. We present such a method, which creates visual explanations in the images space by successively removing either relevant or irrelevant pixels from the input image. A core component of our approach is a novel technique to defend against adversarial evidence (*i.e.* faulty evidence due to artifacts). Using a multitude of experiments, we show the ability of our method to create fine-grained and class-discriminative explanations which are faithful to the model.

Zusammenfassung

Autonome Fahrzeuge versprechen den Verkehrssektor zu revolutionieren, indem sie die Verkehrssicherheit erhöhen, den Ressourcenverbrauch verringern und die Lebensqualität verbessern. Um den uneingeschränkten Einsatz solcher Fahrzeuge im öffentlichen Verkehr zu erreichen, sind noch viele Problemstellungen zu lösen. Ein wesentlicher Forschungsschwerpunkt ist hier die robuste Wahrnehmung und Interpretation des Umfeldes. Mit Deep-Learning-basierten Modellen wurden in den letzten Jahren große Fortschritte in diesem Bereich erzielt. Es bestehen jedoch weiterhin noch viele ungelöste Herausforderungen, wie zum Beispiel die zuverlässige Umfelderfassung unter schwierigen Umgebungsbedingungen (schlechte Lichtverhältnisse, widrige Wetterbedingungen, …). Im ersten Teil dieser Dissertation untersuchen wir Methoden zur Erhöhung der Robustheit von bild-basierten Wahrnehmungsmodellen. Ein vielversprechender Ansatz ist die Fusion der Daten mehrerer komplementärer Sensoren. Aufbauend auf vorherigen Arbeiten im Bereich der Personendetektion entwerfen wir einen multispektralen Detektor, der Bilddaten aus dem sichtbaren Spektrum mit Wärmebilddaten fusioniert. Kern unseres Detektors ist ein neuronales Netz, das Bildausschnitte der Sensordaten kombiniert und klassifiziert. Ein komplementärer Ansatz zur Erhöhung der Leistungsfähigkeit von Wahrnehmungsmodellen ist die zeitliche Filterung von Informationen. Die Filteraufgabe kann in zwei Teile zerlegt werden: Prädiktion und Korrektur. Zunächst untersuchen wir den Aspekt der Prädiktion und schlagen einen Ansatz zur Erzeugung semantischer Vorhersagemodelle vor. Ausgangspunkt unseres Ansatzes ist ein trainiertes Einzelbildmodell, welches durch Erweiterung um ein rekurrentes Modul strukturell transformiert und anschließend selbst-überwacht trainiert wird. Das resultierende prädiktive Modell ist in der Lage, die Bewegung und Interaktion von Objekten zu modellieren und Vorhersagen zu treffen. Aufbauend auf den gewonnenen Erkenntnissen, wird ein parametereffizienter, hierarchischer Ansatz zur zeitlichen Filterung der Repräsentationen des *Fully Convolutional DenseNet* (FC-DenseNet) entworfen. Unter Verwendung von simulierten Videosequenzen mit signifikanten Störungen (z. B. Rauschen, Verdeckungen, fehlende Informationen), zeigen wir die Vorteile der zeitlichen Integration von Informationen auf.

Ein Nachteil vieler aktueller Wahrnehmungsmodelle ist deren Black-Box-Charakter. Insbesondere in sicherheitskritischen Anwendungen ist ein hohes Maß an Transparenz und Interpretierbarkeit vorteilhaft, da es die Modellentwicklung, Fehleranalyse, Validierung und Zertifizierung erleichtert. Im zweiten Teil der Dissertation untersuchen wir zwei Ansätze zur Erhöhung der Transparenz und Interpretierbarkeit von Deep-Learning-basierten

Modellen. Zunächst betrachten wir das Konzept der *Interpretierbarkeit durch Design* und schlagen einen zeitlichen Filter vor, der die Filteraufgabe in mehrere Teilaufgaben zerlegt. Zur Einblicknahme in das Filterverhalten, werden interpretierbare Zwischenrepräsentationen eingefügt, diese ermöglichen zudem die Integrationen von Domänenwissen. Der Filter kann in eine Vielzahl von Einzelbildmodellen integriert werden, um deren Robustheit zu erhöhen. Eine Alternative zum Design von interpretierbaren Architekturen sind Post-hoc-Erklärungsmethoden, die Einblicke in den Entscheidungsprozess von Modellen gewähren. Abschließend stellen wir eine solche Methode vor, die visuelle Erklärungen im Bildraum erzeugt, indem sie schrittweise aus dem Eingangsbild entweder relevante oder irrelevante Pixel entfernt. Eine Kernkomponente unseres Ansatzes ist eine neuartige Technik zur Abwehr fehlerhafter Evidenz. Unsere Methode ist in der Lage, detaillierte, klassendifferenzierende und modellgetreue visuelle Erklärungen zu erzeugen.

Acknowledgments

First and foremost, I would like to thank my doctoral advisor, Prof. Dr. Sven Behnke, for his excellent support, constant feedback, and scientific guidance. My sincere gratitude also goes to Prof. Dr. Jürgen Gall, Prof. Dr. Thomas Schultz, and Prof. Dr.-Ing. Ribana Roscher, for agreeing to serve on my examination committee.

In addition, I sincerely thank my colleagues and friends at the Bosch Center for Artificial Intelligence (BCAI) for their mutual support, fruitful discussions, and scientific inspiration. Our many social activities, kicker games, and music sessions during my time as a Ph.D. student were also a pleasure and relaxing change of pace. I especially want to express my appreciation to Volker Fischer, Michael Herman, Tobias Gindele, Jan Köhler, Johannes Döllinger, Leon Hetzel, Thaddaeus Wiedemer, Sebastian Ziesche, and Markus Spies for their guidance and support in my scientific endeavors.

I am particularly grateful for the guidance, encouragement, and advice of my supervisor at Bosch, Volker Fischer. I will fondly look back on our many discussions, whiteboard sessions, and coffee breaks.

Last but not least, I would like to thank my family and friends for their unwavering support and encouragement on the journey to completing this thesis.

Contents

1

Introduction

Autonomous vehicles, such as self-driving cars or mobile robots, promise to revolutionize the transportation of people and goods (Fagnant and Kockelman, 2015). A majority of current traffic accidents can be attributed to driver-related factors, such as driver distraction, misjudgment of the driving situation, aggressive driving behavior, illegal maneuvers, or inattention (NHTSA, 2008; Singh, 2015). Autonomous vehicles are capable of preventing a large number of these accidents or reducing their severity, assuming they are designed with a safety-first principle (Mueller *et al.*, 2020). Besides safety, further benefits of autonomous vehicles are an increased comfort, an optimized traffic flow, and a lower environmental impact through fuel savings as well as improved usage patterns (Hao and Yamamoto, 2018). Self-driving cars can additionally improve the mobility of elderly people and teenagers (Fagnant and Kockelman, 2015), especially in areas with poor public transport. Furthermore, mobile autonomous robots promise significant cost savings in the transportation and delivery of goods.

The performance of autonomous vehicles depends highly on their ability to produce a robust representation of their surroundings. Such vehicles must be able to reliably perceive and interpret both the static environment (*e.g.* lanes, traffic signs, drivable area) and the dynamic environment (*e.g.* vehicles, pedestrians, animals) in order to navigate safely in an unconstrained and changing world without any human supervision. Errors in the environment representation are passed to all subsequent processing steps and can hardly be detected. Thus, the performance of the perception component is crucial for the functional safety of an autonomous vehicle.

Deep learning-based approaches have greatly advanced the generation of robust environment representations. They became the standard for the majority of current perception models (Grigorescu *et al.*, 2020), especially in the visual domain. Object detectors such as *Regions with CNN features* (R-CNN) (Girshick *et al.*, 2014) and Faster R-CNN (Ren *et al.*, 2015) are used to detect vehicles and pedestrians, semantic segmentation models (Long *et al.*, 2015; Jégou *et al.*, 2017; Zhao *et al.*, 2017) identify the drivable area as well as lanes, and models like the *Multi-Scale Depth Network* of Eigen *et al.* (2014) estimate the geometry of the scene. In addition to new models and methods, decisive factors for the progress of perception systems were more powerful computers and the availability of large benchmark datasets (*e.g.* *CityScapes* (Cordts *et al.*, 2016) or *Caltech* (Dollár *et al.*, 2009b)).

A majority of the previous work in the field of deep learning-based visual perception uses models that make predictions conditioned on a single image and focuses on reducing epistemic failures. We denote failures of a perception system as epistemic if they can be remedied by using more training data or a more advanced model (*e.g.* by introducing task-specific model structures). A drawback of many of these models is their poor performance in challenging scenarios, such as adverse weather conditions, difficult lighting conditions, or short sensor failures (Pfeuffer and Dietmayer, 2020). Such challenging scenarios are highly underrepresented in common large-scale benchmark datasets (Cordts *et al.*, 2016), as these are mostly recorded during daytime in good environment conditions. We refer to a second class of failures, which originate from such data-inherent perturbations, as aleatoric failures in the following. Compared to epistemic failures, these failures cannot be eliminated by using a more advanced model or additional training data. To solve aleatoric failures, one has to enhance the information provided to the perception system. This can be achieved by fusing the information of multiple sensors, by utilizing a more advanced sensor, by integrating context information, and / or by considering temporal information. The categorization into epistemic and aleatoric failures is based on the classification of uncertainties (Kiureghian and Ditlevsen, 2009; Kendall and Gal, 2017).

In this thesis, we focus on increasing the robustness of vision-based perception systems by reducing aleatoric failures. In Chapter 2, we first evaluate the advantages of fusing the information of a camera operating in the visible spectrum with the data of a thermal camera. Given the complementary strengths of the two camera types, a fusion of their sensor data is a promising approach for improving robustness of perception systems. Building upon the R-CNN detection framework (Girshick *et al.*, 2014), we propose a multispectral pedestrian detection model. To fuse the information of the sensor modalities, we evaluate two deep fusion architectures — an early-fusion *Convolutional Neural Network* (CNN) which fuses the information at pixel-level and a late-fusion CNN which performs fusion at feature-level. We overcome the constraint of limited multispectral training data by using a multi-step pre-training procedure that solely relies on training data from the visible spectrum. Our late-fusion-based pedestrian detector significantly outperforms prior detectors on the *KAIST* benchmark (Hwang *et al.*, 2015). To the best of our knowledge, this is the first work to use a deep learning-based approach for multispectral pedestrian detection.

In a second step, we consider the use of temporal information to increase the robustness of semantic segmentation models. Using recurrent representation filters, information of several time steps is temporally integrated. The filter task can be divided into two subtasks: a prediction step and an update step. In the prediction step, information of the previous time step is propagated into the future. The predicted information is fused in the update step with information of the new measurement. To approach this problem systematically, we first focus on the prediction step. In Chapter 3, we propose a method to create semantic forecasting models — *i.e.* models which predict the pixel-wise semantic segmentation of the next time step conditioned on previous observed image frames. Such forecasting models enable an autonomous vehicle to reason about the future environment state and thus provide valuable information for a predictive planner (Rudenko *et al.*, 2017). We create semantic forecasting models by transforming non-predictive feed-forward networks. Our proposed predictive transformation extends a given feed-forward network with a recurrent predictive module, while reusing its original structure and encoded task-specific knowledge. We suggest self-supervised training using a teacher-student-like strategy to eliminate the need for costly labeled data. In addition, we consider the applicability of our approach in a semi-supervised setting, when no appropriate feed-forward model is available.

Building on the experience we have gained with our predictive transformation approach, Chapter 4 focuses on using temporal information to reduce aleatoric failures of a single-frame segmentation network (FC-DenseNet (Jégou *et al.*, 2017)). We present a parameter efficient approach to temporally filter the representations of FC-DenseNet in a hierarchical and recurrent manner. The resulting architecture, *Recurrent Fully Convolutional DenseNet* (RFC-DenseNet), utilizes temporal correlations on all abstraction levels and conceptually decouples temporal dependencies from scene representation. By using information from multiple time steps, our proposed model is more capable in suppressing noise, inferring additional object properties, and resolving missing information as well as ambiguities compared to single-image models. It also shows an improved performance compared to other common temporal semantic segmentation models.

One drawback of the models presented so far is their black-box character. Compared to classical approaches, these deep learning models have a low level of transparency and interpretability (Arrieta *et al.*, 2020). Especially in safety-critical applications, such as autonomous driving, it is advantageous to have insights into the decision-making process of a model. Such insights facilitate debugging, they can help to better understand shortcomings and limitations of a model, and provide information which might be required to certify or verify models (Arrieta *et al.*, 2020; Fan *et al.*, 2021; Zablocki *et al.*, 2021). Additionally, they may simplify incorporating prior knowledge and can be used to increase the trust of a user (McAllister *et al.*, 2017).

As a second focus of this thesis, we examine two approaches to increase the transparency and interpretability of models: *interpretability by design* and a post-hoc explanation method. In Chapter 5, similar to Chapter 4, we use a recurrent representation filter to increase the robustness of a semantic segmentation model. However, we design the filter to be more transparent and interpretable, by modularizing functionalities, using explicit physical models as subcomponents, and introducing human interpretable intermediate representations. The modular structure of our proposed architecture also allows for pre-training, evaluation, and easy replacement of subcomponents. Our experiments especially highlight the advantages introduced by an interpretable and explicit filter structure.

In some cases, it is not possible to consider the interpretability requirement already in the design phase of a model. Post-hoc explanation methods can then be used to examine the behavior of resulting black-box models. The most common form of such methods are local ones, which explain the prediction of a model (*e.g.* CNN) for a specific input (*e.g.* image). A popular form of local explanations are visual, image-like representations, which depict the pixels or image regions that significantly determine the output of a model. In Chapter 6, we propose a post-hoc, local explanation method, named *Fine-Grained Visual Explanation Method* (FGVis), which generates fine-grained and class-discriminative visual explanations for CNNs. To prevent the generation of adversarial evidence (*i.e.* faulty evidence due to artifacts), we propose a novel adversarial defense which does not depend on human-tuned parameters and imposes no constraints (*e.g.* a reduced resolution) on explanations. Visual explanations created by FGVis are intuitively interpretable, preserve the characteristics of images (*e.g.* edges and colors), and can be tested as they are valid model inputs. FGVis is thus a useful tool to identify and analyze failure cases and shortcomings of CNNs. The obtained insights can be used to improve the robustness of a CNN.

1.1 Contributions

This thesis contributes to improving the robustness of visual perception systems, which are critical for the realization of autonomous vehicles. To increase robustness, we investigate different models and methods that fuse data from multiple sensors or integrate information over time.

Considering the safety critical nature of an autonomous vehicle, it is beneficial to gain insights into the decision-making process of used models. As a second focus of this thesis, we investigate two approaches to increase the transparency and interpretability of models: *interpretability by design* and a post-hoc explanation method.

In particular, the main contributions of this thesis are:

- *Multispectral Pedestrian Detection.* We propose a multispectral pedestrian detector which is based on the R-CNN detection framework. To fuse the information of a visible and a thermal camera, we evaluate a late- and an early-fusion CNN. The limited availability of multispectral training data is address by a multi-step pre-training procedure that exclusively relies on visible images. Our best model significantly outperforms prior detectors on the *KAIST* benchmark (Hwang *et al.*, 2015). To the best of our knowledge, this is the first work to use a deep learning-based approach for multispectral pedestrian detection.
- *Semantic Forecasting Models.* Autonomous vehicles have to act in an anticipatory manner in order to facilitate a safe deployment. Consequently, they must be able to reason about the future state of the environment. We propose a method for creating semantic forecasting models, *i.e.* models which predict the semantic segmentation of the next time-step conditioned on previous observed images, by transforming non-predictive feed-forward networks. The predictive transformation

is based on a structural extension of the feed-forward network using a recurrent predictive module and a teacher-student training strategy. Training is performed in a fully self-supervised fashion, eliminating the need for costly labeled data. Furthermore, we show that our approach is applicable in a semi-supervised setting, when no appropriate feed-forward model is available.

- *Hierarchical and Recurrent Filtering.* We present a parameter efficient temporal filtering concept, which extends the *Fully Convolutional DenseNet* (FC-DenseNet) of Jégou *et al.* (2017) to multiple video frames. Temporal integration is achieved by recurrently filtering the representations of FC-DenseNet on all abstraction levels in a hierarchical manner. The resulting *Recurrent Fully Convolutional DenseNet* (RFC-DenseNet) conceptually decouples temporal dependencies from scene representation. By integrating the information of previous time steps, RFC-DenseNet is able to produce a more robust pixel-wise semantic segmentation compared to single-image models. It also shows improved performance compared to other common temporal semantic segmentation models.
- *Modular and Interpretable Filtering.* We propose a temporal representation filter to reduce aleatoric failures of a single-frame semantic segmentation model. The filter is designed to be inherently more transparent and interpretable, by modularizing functionalities, using explicit physical models as subcomponents, and introducing human interpretable intermediate representations. It consists of multiple submodules, which can be pre-trained, debugged, and evaluated independently. Compared to many other temporal architectures, our filter is less susceptible to missing information. The additional insights provided by the interpretable structure can be used to understand shortcomings of the model and to derive new concepts for its improvement.
- *Explaining Model Predictions.* We propose a post-hoc, optimization-based explanation method (FGVis) to explain the prediction of a CNN for an input image. The main contribution of our method is a novel adversarial defense, which prevents the generation of adversarial evidence (*i.e.* faulty evidence due to artifacts). Compared to other adversarial defense techniques, ours does not depend on human-tuned hyperparameters and imposes no further constraints (*e.g.* a reduced resolution) on explanations. FGVis creates fine-grained and class-discriminative visual explanations in the image space. In addition, explanations are intuitively interpretable,

preserve the characteristics of images (*e.g.* edges and colors), and can be tested as they are valid model inputs.

1.2 Publications

This thesis is based on work published in the following conference proceedings:

- Wagner, J., V. Fischer, M. Herman, and S. Behnke (2016). “Multispectral Pedestrian Detection using Deep Fusion Convolutional Neural Networks”. In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 509–514
- Wagner, J., V. Fischer, M. Herman, and S. Behnke (2017). “Learning Semantic Prediction using Pretrained Deep Feedforward Networks”. In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 565–570
- Wagner, J., V. Fischer, M. Herman, and S. Behnke (2018a). “Functionally Modular and Interpretable Temporal Filtering for Robust Segmentation”. In: *Proceedings of the British Machine Vision Conference (BMVC)*
- Wagner, J., V. Fischer, M. Herman, and S. Behnke (2018b). “Hierarchical Recurrent Filtering for Fully Convolutional DenseNets”. In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 49–54
- Wagner, J., J. M. Köhler, T. Gindele, L. Hetzel, J. T. Wiedemer, and S. Behnke (2019). “Interpretable and Fine-Grained Visual Explanations for Convolutional Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9089–9099. DOI: 10.1109/CVPR.2019.00931; © 2019 IEEE

1.3 Outline

This thesis is organized in seven chapters: Chapter 1 introduces the context of the thesis, outlines the bigger picture, and summarizes the scientific contributions. In Chapter 2 to Chapter 6, the scientific contributions are presented in detail. Finally, Chapter 7 recapitulates the proposed methods and gives an outlook on possible future fields of research.

The five chapters covering scientific contributions are based on work published in conference proceedings (see Section 1.2) and are written in a self-contained manner. Each chapter includes an introduction, a discussion of related work, a method section with accompanying experiments, and a conclusion. Relevant related work released after the publication of the content of each chapter is discussed in each chapter's conclusion. Although the chapters are self-contained, we still recommend reading them consecutively, as ideas and approaches from previous chapters are revisited in subsequent ones.

In Chapter 2 to Chapter 6, we propose, discuss, and evaluate deep learning-based methods and models for visual perception. In particular, we focus on improving the robustness of visual perception systems (Chapter 2 – 5) as well as approaches for increased model transparency and interpretability (Chapter 5 – 6). The topics covered by each chapter are:

- Chapter 2: *Multispectral Pedestrian Detection*

  Sensor Fusion, Deep Fusion CNN, Early-Fusion vs. *Late-Fusion, R-CNN Detection Framework, Unimodal Pre-Training, Multispectral Sensor System,* KAIST *Benchmark*

- Chapter 3: *Semantic Forecasting Models*

  Future Prediction, Predictive Transformation, Teacher-Student Training, Recurrent Predictive Module, Self-Supervised / Semi-Supervised Learning, Semantic Forecasting

- Chapter 4: *Hierarchical and Recurrent Filtering*

  Temporal Filtering, Hierarchical Filter Concept, Recurrent Filter Modules, Dense Connection Pattern, Robust Semantic Segmentation, Parameter Efficient Structure

- Chapter 5: *Modular and Interpretable Filtering*

  Temporal Filtering, Interpretability by Design, Modular Filter Structure, Human Interpretable Intermediate Representations, Robust Semantic Segmentation, Multi-Task Learning, Inspecting Model Behavior, Physical Constraints / Subcomponents, Geometric Projection, Transparent Structures

- Chapter 6: *Explaining Model Predictions*

  Post-hoc / Optimization-based Visual Explanation Method, Novel Defense Against Adversarial Evidence, Inspecting Model Behavior, Fine-Grained and Class-Discriminative Visual Explanations, Faithful Explanations

2 Multispectral Pedestrian Detection

A robust, vision-based pedestrian detection system is an important building block of future autonomous vehicles, such as self-driving cars or mobile robots. Current detection systems mainly rely on cameras operating in the visible spectrum due to their low price, high resolution, and ability to provide rich color and texture information. However, under low or rapidly changing lighting conditions, the information content of a visible image is significantly reduced, leading to failures of current detectors. To reduce these aleatoric failures, one can fuse the information of the visible spectrum with the data of a thermal camera. The complementary strengths of the two sensors lead to a more robust detection system.
In this chapter, we study the benefits of using a multispectral pedestrian detection system. In contrast to prior multispectral pedestrian detectors, we investigate the potential of using *Convolutional Neural Network*s to fuse the information of the two sensor modalities. We evaluate two deep fusion architectures and analyze their performance on the *KAIST* multispectral pedestrian detection benchmark. To address the limited availability of multispectral training data, we propose a multi-step pre-training procedure using only visible images. Our best performing detector uses a deep late-fusion architecture and outperforms the current state-of-the-art pedestrian detector on the *KAIST* benchmark significantly.

2.1 Introduction

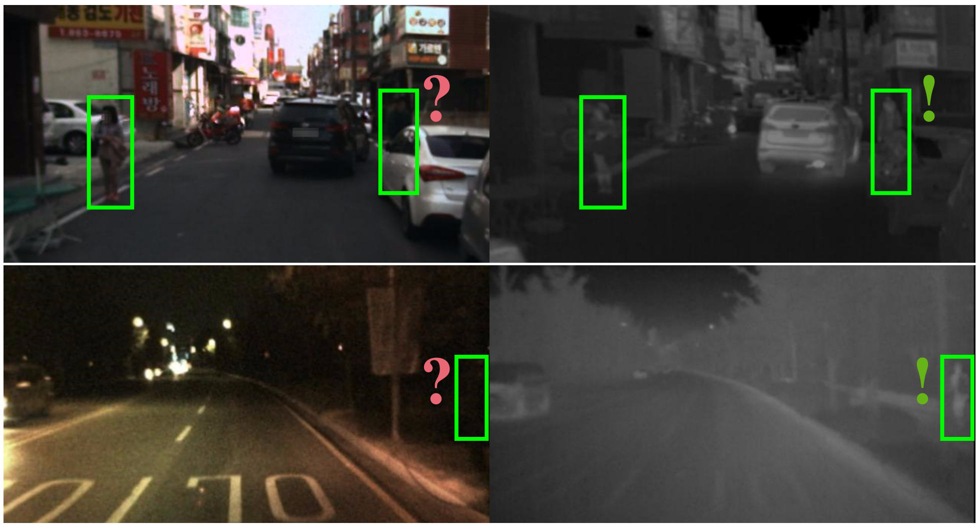
Vision-based pedestrian detection is a critical capability of future autonomous vehicles, such as mobile robots or self-driving cars. Especially for self-driving cars, a robust detection of pedestrians is a mandatory requirement to ensure a safe deployment in populated environments like city centers. Although the topic has been intensively researched in the last decade (Benenson *et al.*, 2014; Dollár *et al.*, 2012), it is still a challenging task due to the variability of the pedestrian's shape, clothes, pose, occlusion, and illumination as well as the variability of the environment. Typical instances of these challenges are highlighted in Figure 2.1a and Figure 2.1b using images of the *KAIST* multispectral pedestrian detection dataset (Hwang *et al.*, 2015).

The majority of past research focused on the detection of pedestrians in images of the visible spectrum, where multiple benchmark datasets with comparatively large amounts of annotated pedestrians are available (Benenson *et al.*, 2014). For a long period of time, approaches using hand-crafted features dominated these benchmarks. With the recent interest of the vision community in deep learning-based approaches, an increasing number of top performing detectors utilize *Convolutional Neural Network*s (CNNs).

A major drawback of pedestrian detectors operating on images of the visible spectrum (visible images) is their poor performance in low lighting conditions (*e.g.* at night or in the shade), as well as their sensitivity to illumination changes. To overcome these drawbacks, it is helpful to fuse the information of a visible camera with the information provided by a long-wavelength infrared (thermal) camera (Gade and Moeslund, 2014). Due to the spectral band in which a thermal camera operates, it does not only omit the need for an external light source, but is also less affected by bad weather conditions. In comparison to visible cameras, thermal cameras therefore perform particularly well at night, in rapidly changing lighting conditions, and can also observe pedestrians in the shade. Example images of such situations are depicted in Figure 2.1c. On the other hand, thermal cameras often exhibit a decrease in image quality during daytime due to a high background temperature. Additionally, thermal images are usually of lower resolution and do not provide rich texture information. Due to their complementary strengths, a fusion of both sensors is a promising approach to construct a more robust detection system. In the past, multispectral detectors (*i.e.* detectors which utilize the information of a visible and a thermal camera) were mainly used in the military and

(a) Pedestrian detectors have to cope with a variety of environments and illumination conditions.

(b) The appearance of pedestrians can differ drastically due to variability in clothing, pose, occlusion and lighting.



(c) For two scenes we show the visible image on the left side and the corresponding thermal image on the right side. Thermal cameras measure the long-wavelength infrared radiation. They are therefore well suited for detecting pedestrians at night or in low lighting conditions. Although the pedestrians are poorly illuminated in both scenes, their appearance can be clearly perceived in the thermal image.

Figure 2.1: Images of the *KAIST* multispectral pedestrian detection dataset (Hwang *et al.*, 2015), highlighting the variability of the environment and of pedestrians as well as the benefits of thermal imaging.

surveillance domain. With the recent decline in the price of thermal cameras, these detectors are becoming increasingly attractive for other domains, such as self-driving cars or mobile robots.

In this chapter, we propose a CNN-based multispectral pedestrian detection method. Our detector builds upon the R-CNN detection framework (Girshick *et al.*, 2014) and extends it to the multispectral domain by using CNNs to fuse the information of a visible and a thermal camera. To the best of our knowledge, this is the first work to use a deep learning-based approach for multispectral pedestrian detection. We evaluate an early- and a late-fusion approach and analyze their performance on the *KAIST* multispectral pedestrian detection benchmark (Hwang *et al.*, 2015). To overcome the constraint of limited multispectral training data, we propose a multi-step pre-training procedure using only visible images. We show that our late-fusion-based deep model, which is additionally pre-trained on auxiliary datasets, outperforms the state-of-the-art baseline on the *KAIST* benchmark significantly.

The approach of this chapter was first presented and published at the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (Wagner *et al.*, 2016).

2.2 Related Work

Vision-based pedestrian detection has been intensively researched in the last decade (Benenson *et al.*, 2014; Dollár *et al.*, 2012). The majority of past research focused on the detection of pedestrians in images of the visible spectrum. Early top performing detectors mainly relied on the sliding window paradigm (Dollár *et al.*, 2010) and used hand-crafted features, such as *Histogram of Oriented Gradients* (HOGs) (Dalal and Triggs, 2005; Felzenszwalb *et al.*, 2010), *Haar-like* features (Papageorgiou *et al.*, 1998; Viola and Jones, 2001), and channel features (Dollár *et al.*, 2009a; Dollár *et al.*, 2014). Common choices for the classifier were *Support Vector Machines* (SVMs) (Dalal and Triggs, 2005; Felzenszwalb *et al.*, 2008) and boosted decision trees (Benenson *et al.*, 2012; Benenson *et al.*, 2013). Cascade architectures were often used to quickly reject negative examples, resulting in a decrease in detector runtime (Viola and Jones, 2001; Bourdev and Brandt, 2005).

More recent works propose CNN-based pedestrian detection methods. Yang *et al.* (2015) extend the channel features framework (Dollár *et al.*, 2009a; Dollár *et al.*, 2014) by integrating features extracted from a CNN pre-trained on *ImageNet* (Russakovsky *et al.*, 2015). Sermanet *et al.* (2013) use a multi-scale convolutional network for pedestrian detection, which is pre-trained in an unsupervised manner. A multitude of detectors build upon the R-CNN detection framework of Girshick *et al.* (2014). Using a proposal generator, a set of candidate bounding boxes is generated for an input image. These proposals are evaluated in a second stage by a CNN-based classifier. A detailed investigation of various aspects of this framework in the context of pedestrian detection is conducted by Hosang *et al.* (2015). A variety of papers propose approaches to improve the second stage of the R-CNN framework. Verma *et al.* (2015) use a mixture of CNN experts in the second stage. Tian *et al.* (2015) propose to use additional auxiliary tasks (*e.g.* pedestrian and scene attribute prediction) and datasets to train a more robust CNN classifier. Similar detection methods using more complex cascade structures are introduced by Angelova *et al.* (2015) and Cai *et al.* (2015). These methods focus on increasing the accuracy of detectors by using CNNs while ensuring a short processing time. Other approaches such as *JointDeep* (Ouyang and Wang, 2013) or *Switchable Deep Network* (Luo *et al.*, 2014) explicitly model concepts like deformable body parts or occlusion into the network architecture. Further improvements to the R-CNN detection framework, like Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren *et al.*, 2015), and new detection concepts (Redmon *et al.*, 2016) were developed concurrently to the preparation of the publication covering the content of this chapter.

Variations of the introduced approaches for detecting pedestrians in visible images were adopted to the thermal domain. Suard *et al.* (2006) use HOG features in combination with a SVM classifier to detect pedestrians in thermal images. To reduce the runtime of the detector they propose a heuristic which generates bounding box proposals based on the temperature of image regions. Zhang *et al.* (2007) evaluate the usability of various feature-classifier combinations developed for visible images in the thermal domain. They conclude that methods developed for visible images can achieve comparable detection accuracy on thermal images. A larger number of features and classifiers developed for visible images are compared by Teutsch *et al.* (2014) using four thermal datasets. On the basis of their assessment, they propose *Discrete Cosine Transform*-based features in combination with a modified *Random Naïve Bayes classifier* for pedestrian classification. Their overall pedestrian detection system uses a two-stage approach, similar to the R-CNN framework, consisting of a hot spot-based proposal generator and the pedestrian classifier.

Multispectral pedestrian detectors can be divided into three categories based on the level of abstraction at which the fusion takes place — pixel-level fusion, feature-level fusion, and decision- or score-level fusion (Liggins *et al.*, 2008; Zin *et al.*, 2011). Choi and Park (2010) use a joint bilinear filter to fuse the information of a thermal- and a visible image at pixel-level. To detect pedestrians in the fused image they use a background subtraction method. Such a detector is limited to surveillance applications as it assumes a stationary camera and moving pedestrians. Additionally, the pedestrian temperature is generally assumed to be higher than the background temperature. Similar assumptions are made by Leykin and Hammoud (2006) to develop a multispectral pedestrian detection and tracking system. Fusion takes place on pixel-level within a visible-thermal scene background model. To better distinguish pedestrians from other moving objects, they propose an additional verification stage using a pedestrian classifier based on movement patterns in a subsequent work (Leykin *et al.*, 2007). Further related works, which focus on fusing the two modalities in a visible-thermal scene background model, are proposed by Conaire *et al.* (2005) and St-Laurent *et al.* (2007). A method for multispectral pedestrian detection that fuses thermal and visible information at feature-level is proposed by Krotosky and Trivedi (2008). They combine HOG features extracted from a visible and thermal image and employ a SVM classifier. The result of the HOG-based classifier is fused with the result of a disparity-based classifier in a second processing step at decision-level. The disparity image is calculated using a third visible or thermal camera. They report a significant increase in performance by using a multispectral detector compared to a detector that uses either the visible or the thermal image only. Another method that performs a feature-level fusion is introduced by Hwang *et al.* (2015). Their pedestrian detector is an extension of the *Aggregated Channel Features* (ACF) detector of Dollár *et al.* (2014) using additional feature channels computed from the thermal image. To evaluate the detector they introduce a large-scale multispectral pedestrian detection benchmark dataset — the *KAIST benchmark* (Hwang *et al.*, 2015). If there is a poor spatial registration of the two sensor modalities, a fusion at decision level may be appropriate. However, this also requires that each modality on its own contains sufficient information to enable an independent detection of pedestrians. Such a decision-level fusion approach is proposed by Bertozzi *et al.* (2006) to fuse the bounding boxes of two independent pedestrian detectors. Each detector independently processes the data from a stereo camera, of which one operates in the visible spectrum and one in the thermal spectrum. Similar approaches are proposed by Lee *et al.* (2015) and Torresan *et al.* (2004) to construct multispectral pedestrian detectors.

Utilizing the multispectral dataset of Hwang *et al.* (2015), we investigate the potential of using deep neural networks for multispectral pedestrian detection. Our proposed detector is based on the R-CNN framework, which was successfully used to develop promising detectors in the visible domain. To extend these ideas to the multispectral setting, we propose two deep fusion architectures, an early-fusion CNN that fuses information at pixel level and a late-fusion CNN that fuses information at feature level. Additionally, we propose a multi-step pre-training procedure using only visible images, to overcome the

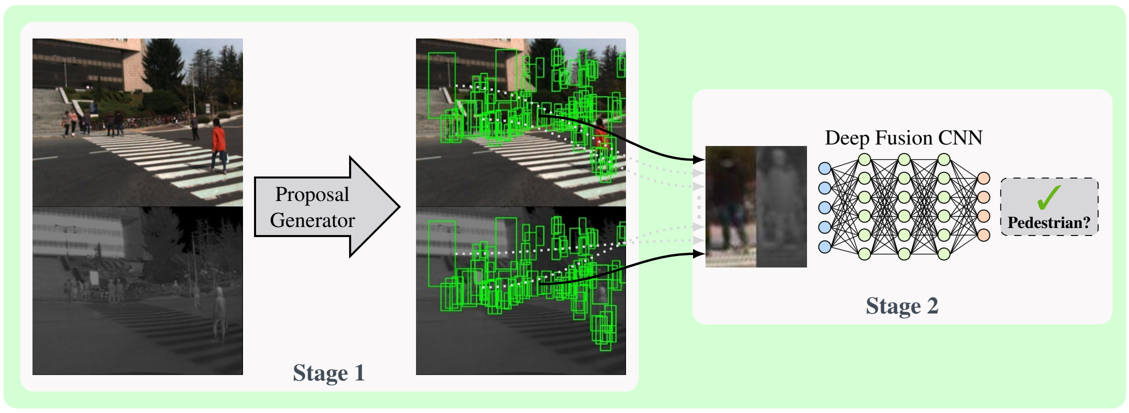


Figure 2.2: Detection framework employed by our multispectral pedestrian detector. Our detector follows the R-CNN framework consisting of a proposal generator (**Stage 1**) and a deep fusion CNN (**Stage 2**). We evaluate two options for fusion, an early- and a late-fusion CNN architecture. The example images are taken from the *KAIST* multispectral pedestrian detection benchmark (Hwang *et al.*, 2015). All bounding box proposals of the first stage are shown as green rectangles in both the visible and thermal image. For visualization purposes, the visual and thermal image are displayed side by side (**Stage 2**) or one above the other (**Stage 1**). The actual input representation used for the two fusion CNNs differs and is described in detail in Section 2.3.2.

constraint of limited multispectral training data. To the best of our knowledge, this is the first work to use a deep learning-based approach for multispectral pedestrian detection. Our detector is well suited for autonomous vehicles, like self-driving cars, as it does not impose constraints on the geometry and dynamics of the scene. In this respect, we differ in particular from the mentioned pixel-level detectors, which assume a static camera and moving pedestrians. In comparison to detection methods which fuse information on decision-level, our models should be better able to use inherent relations between the sensor modalities.

2.3 Multispectral Pedestrian Detector

2.3.1 Detection Framework

Our multispectral pedestrian detector is based on the R-CNN detection framework of Girshick *et al.* (2014), which is first applied by Hosang *et al.* (2015) to detect pedestrians in visible images. According to the R-CNN framework, we utilize a proposal generator to generate a set of candidate detections (*i.e.* candidate bounding boxes). The multispectral image data of the proposals is transformed into a standardized size and classified using a deep fusion CNN. The input size of the used CNN is set to 128×64 pixels containing

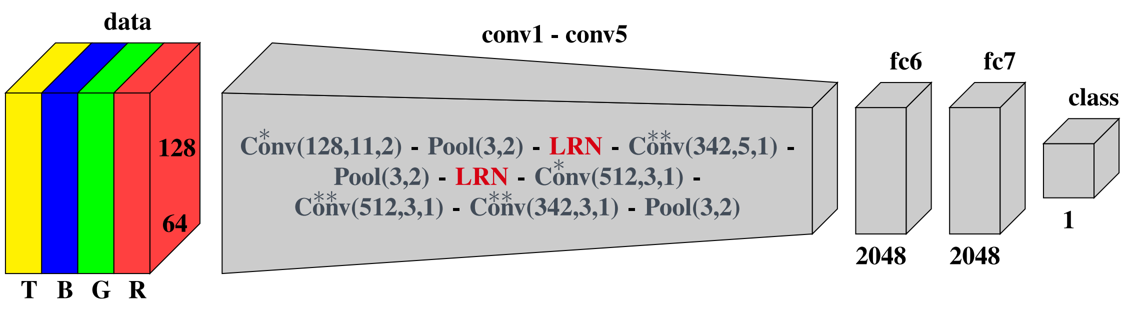

data
128
64
T B G R
conv1 - conv5
$\overset{*}{\text{Conv}}$(128,11,2) - Pool(3,2) - LRN - $\overset{**}{\text{Conv}}$(342,5,1) -
Pool(3,2) - LRN - $\overset{*}{\text{Conv}}$(512,3,1) -
$\overset{**}{\text{Conv}}$(512,3,1) - $\overset{**}{\text{Conv}}$(342,3,1) - Pool(3,2)
fc6
fc7
class
2048
2048
1

Figure 2.3: Early-fusion architecture which fuses the visible and thermal image on pixel-level. The model receives the thermal and visible image concatenated along the feature dimension as input. $\overset{*}{\textbf{Conv}}$**(f,k,s)**: convolutional layer with **f** filters, a kernel size of **k**, a stride of **s**, and a ReLU nonlinearity. A second asterisk in the superscript indicates the use of two input groups. **Pool(k,s)**: max-pooling layer with a kernel size of **k** and a stride of **s**. **LRN**: local response normalization layer. We apply dropout and use ReLU nonlinearities in the first two fully connected layers, followed by a binary classification layer.

the content of the proposal bounding box as well as additional surrounding context. The proposal occupies 100×41 pixels centered in the input image the remainder is padded context. As depicted in Figure 2.2, the resulting detector thus consists of a two-stage cascade, whose first stage is the proposal generator and whose second stage a neural network. By limiting the search space of potential pedestrians using the proposal generator, the more complex and computationally more expensive CNN only has to evaluate a drastically reduced set of pedestrian candidates.

To generate the proposals, we use the *multispectral ACF* detector of Hwang *et al.* (2015), which is described in detail in Section 2.4.2. The proposal generator is tuned to achieve a high detection rate while keeping the false alarm rate fairly low. To further reduce the number of proposals and to decrease redundancy, we apply non-maximum suppression (Viola and Jones, 2004) in a greedy fashion to the proposal bounding boxes using an *Intersection over Union* (IoU) threshold of 0.65. In total, we investigate two deep fusion CNNs for the second stage of the multispectral pedestrian detector, which are presented in the next subsection.

2.3.2 Deep Fusion Architectures

We investigate an early- and a late-fusion-based CNN architecture to combine and classify the information of the visible and thermal image. The early-fusion architecture (*EarlyFusion*) combines the information of the two modalities at pixel-level. In contrast, the late-fusion CNN (*LateFusion*) uses separate subnetworks to generate a feature representation for each modality. These representations are combined in an additional fully connected layer. Similar late-fusion architectures have been used by Eitel *et al.* (2015) to perform *RGB-D* object recognition and by Simonyan and Zisserman (2014) for action recognition in videos. The structure of the early-fusion architecture as well as the subnetworks of the late-fusion architecture are based on the *CaffeNet* model (Jia *et al.*, 2014).

Early-Fusion Architecture

For the early-fusion architecture (*EarlyFusion*), we use *CaffeNet* and increase the number of filters per convolutional layer by a factor of $^4/_3$, to compensate for the fourth input channel. Additionally, we decrease the number of neurons in the fully connected layers to 2048 and replace the 1000 class classification layer by a binary classification layer. Furthermore, the stride of the first convolutional layer is reduced to two, to obtain a sufficient spatial resolution after the last convolutional layer. The network receives images of size $4 \times 128 \times 64$ as input, containing a concatenated version of the visible-thermal image pair. By combining the two modalities at pixel-level, we expect a better utilization of inherent relations between the sensor modalities. We use *Rectified Linear Unit* (ReLU) nonlinearities (Nair and Hinton, 2010) and apply dropout (Srivastava *et al.*, 2014) after the first and second fully connected layer. The overall model architecture is depicted in Figure 2.3.

Late-Fusion Architecture

The late-fusion architecture (*LateFusion*) processes the data of the two modalities separately in subnetworks and fuses the resulting feature representations in a fully connected layer. The subnetwork which processes visible images will be referred to as *CaffeNet-RGB* and the thermal subnetwork as *CaffeNet-T*. Both subnetworks are based on *CaffeNet* without its classification layer and use, analogous to the early-fusion CNN, 2048 neurons in their fully connected layers, as well as a stride of two in their first convolutional layer. We use ReLU nonlinearities in both subnetworks and apply dropout after the two fully

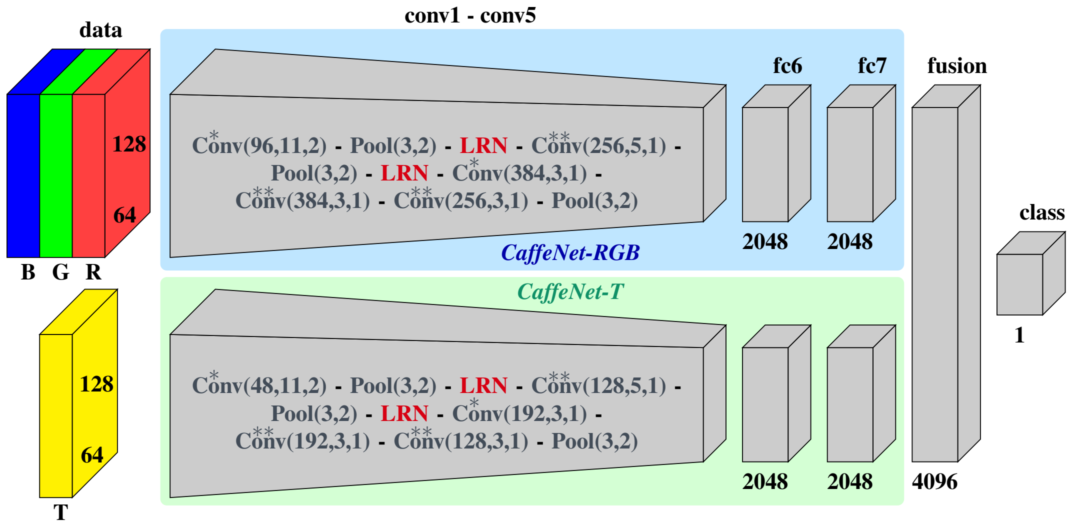

data
B G R
128
64
T
128
64
conv1 - conv5
Conv(96,11,2) - Pool(3,2) - LRN - Conv(256,5,1) - Pool(3,2) - LRN - Conv(384,3,1) - Conv(384,3,1) - Conv(256,3,1) - Pool(3,2)
CaffeNet-RGB
CaffeNet-T
Conv(48,11,2) - Pool(3,2) - LRN - Conv(128,5,1) - Pool(3,2) - LRN - Conv(192,3,1) - Conv(192,3,1) - Conv(128,3,1) - Pool(3,2)
fc6
fc7
fusion
2048
2048
2048
2048
4096
class
1

Figure 2.4: Late-fusion architecture consisting of two subnetworks (*CaffeNet-T* and *CaffeNet-RGB*) which preprocess the two sensor modalities independently. $\overset{*}{\textbf{Conv}}\textbf{(f,k,s)}$: convolutional layer with **f** filters, a kernel size of **k**, a stride of **s**, and a ReLU nonlinearity. A second asterisk in the superscript indicates the usage of two input groups. **Pool(k,s)**: max-pooling layer with a kernel size of **k** and a stride of **s**. **LRN**: local response normalization layer. We apply dropout and use ReLU nonlinearities in the fusion layer as well as the fully connected layers of the subnetworks.

connected layers. In subnetwork *CaffeNet-T*, we additionally halve the number of filters per convolutional layer. The factor 0.5 is derived based on the number of grayscale filters in the first convolutional layer of *CaffeNet* (Jia *et al.*, 2014). The resulting activations of the two subnetworks are concatenated along the feature dimension and fused in a fully connected layer using 4096 filters. The fusion layer uses a ReLU nonlinearity and is followed by a dropout layer and a binary classification layer. A detailed depiction of the architecture is shown in Figure 2.4.

2.3.3 Training Procedure

The availability of a sufficiently large dataset is a crucial requirement when training deep neural networks. However, due to the cost associated with data generation and labeling, the amount of available training data is limited in many applications. One popular approach to overcome this problem is to pre-train a network on a large auxiliary dataset. The *ImageNet* dataset (Russakovsky *et al.*, 2015) has become the de facto standard dataset for model pre-training in the computer vision domain. To evaluate the benefits of pre-training in our multispectral setting, we train our networks twice. Both by only using the training data provided by the multispectral benchmark dataset and by additionally pre-training the networks on auxiliary datasets. Due to the lack of available large visible-thermal image datasets, we pre-train our models using visible data only. The red channel is used as an approximation of the thermal channel in our pre-training procedure. From a physical point of view, this approximation is rather crude given the differences between the two sensor concepts. Nevertheless, the visual representation of both sensor modalities has numerous similarities (*e.g.* edges at object borders), which should enable a transfer of pre-trained filters. During pre-training, the images of the early-fusion architecture contain the red channel twice. It is thus unclear whether the early-fusion architecture can learn complementary features which utilize the substituted thermal channel. We therefore expect a greater benefit from pre-training for the late-fusion architecture.

Our pre-training procedure on auxiliary datasets consists of the following two steps: In the first step, the convolutional layers of *CaffeNet-T*, *CaffeNet-RGB* and the *Early-Fusion* network are pre-trained on the task of image classification using the *ImageNet* dataset (Russakovsky *et al.*, 2015). To adapt the three networks to the *ImageNet* benchmark, we add a 1000 class classification layer to the two subnetworks of the late-fusion CNN and replace the binary classification layer of the early-fusion model with a multi-class classification layer (*i.e.* a fully connected layer with 1000 filters, followed by a soft-max function). In the second step, we fine-tune these networks using the *Caltech* (Dollár *et al.*, 2009b) dataset which was created for the development and benchmarking of pedestrian detectors based on visible images. We use the *Caltech10x* sampling strategy proposed by Hosang *et al.* (2015) and use ground truth bounding boxes to extract positive training samples. Negative training samples are created using bounding box proposals generated by an ACF pedestrian detector (Dollár *et al.*, 2014). A proposal is counted as a negative sample if its *Intersection over Union* (IoU) overlap with all ground truth bounding boxes does not exceed a threshold of 0.5. To be consistent with the cascaded processing scheme of our proposed multispectral detector (see Section 2.3.1), we extend the positive and negative samples to contain additional image context and transform them into a standardized size of 128×64 pixels. Using these training samples, we further pre-train the three networks of step one. To make the models compatible with the new data, we replace the 1000 class classification layer with a randomly initialized binary classification layer and re-initialize the weights of the fully connected layers with random values. During both pre-training steps, the two subnetworks of the late-fusion architecture are solely trained separately.

Finally, we fine-tune the models on the multispectral data of the *KAIST* benchmark dataset (Hwang *et al.*, 2015). Analogously to Hosang *et al.* (2015), we use every second frame of the training data. Additionally, we split the original training data into a training set containing $92\,\%$ of the images and a validation set containing the remaining $8\,\%$ of the images. We create positive and negative samples similar to the procedure used to create the *Caltech* pre-training data. Instead of using the ACF detector to extract negative samples, we use the *multispectral ACF* detector (Hwang *et al.*, 2015) of our two-stage detection cascade (see Section 2.3.1). The fine-tuning of the late-fusion model occurs in two steps. First, we separately optimize the two subnetworks (*CaffeNet-T* and *CaffeNet-RGB*) by adding a binary classification layer at the end. Depending on whether we use pre-training or not, the weights of the two subnetworks are initialized with either pre-trained weights, or random values. The second step encompasses a fine-tuning of the full late-fusion architecture on the *KAIST* data. As suggested by Eitel *et al.* (2015), the best fusion results can be achieved when the weights of the subnetworks are fixed and only the subsequent layers are trained. Due to its simple structure, the early-fusion network is trained or fine-tuned, depending on the usage of pre-training, in a regular fashion.

2.4 Experiments

2.4.1 Dataset

To train and evaluate our multispectral pedestrian detectors we use the *KAIST* multispectral pedestrian detection benchmark dataset of Hwang *et al.* (2015). This dataset consists of temporally and spatially aligned visible and thermal image pairs with a resolution of 640×512 pixels. The images are captured at a frame rate of $20\,\mathrm{Hz}$ from a sensor system mounted at the rooftop of a car driving at different times of day in regular traffic. In total, the dataset contains $95.3\mathrm{k}$ images pairs, split into a training set of $50.2\mathrm{k}$ images with $41.5\mathrm{k}$ labeled pedestrians and a test set of $45.1\mathrm{k}$ images with $44.7\mathrm{k}$ labeled pedestrians. Example images with corresponding ground truth pedestrian bounding boxes are shown in Figure 2.5.

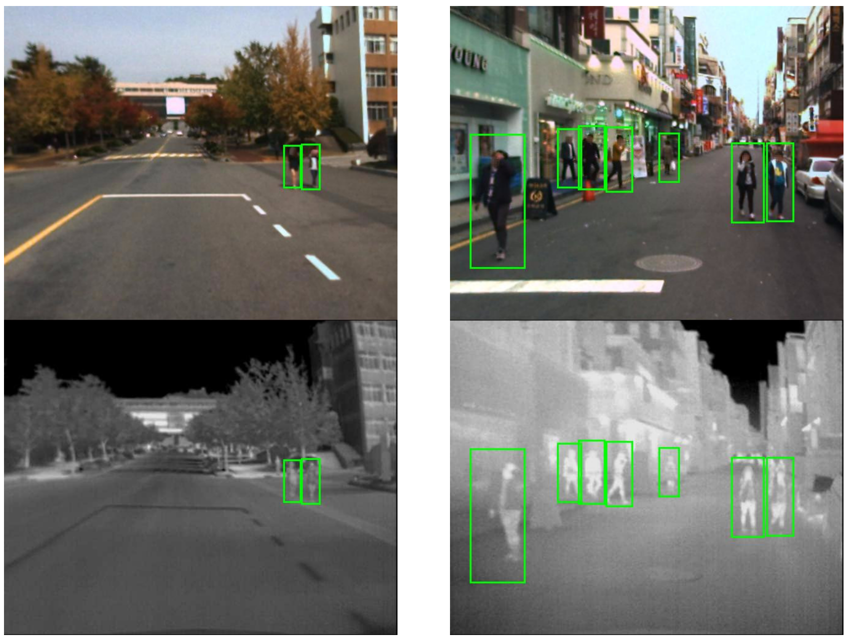


Figure 2.5: Example images of the *KAIST* multispectral pedestrian detection dataset (Hwang *et al.*, 2015) containing ground truth pedestrian bounding boxes. For each scene we show the visible image at the top and the corresponding thermal image at the bottom.

The evaluation of the detectors is performed on the *reasonable all* subset of the *KAIST* test dataset (Hwang *et al.*, 2015). This subset consists of pedestrians which are unoccluded or partially occluded and have a pixel height of 55 or taller. Additionally, we report the performance of the detectors on the *reasonable day* and *reasonable night* subset. These two subsets each contain images recorded either only during the day or only during the night. The *reasonable all* subset is thus the union of the *reasonable day* and *reasonable night* evaluation settings.

2.4.2 Baseline

The best performing approach on the *KAIST* benchmark is an extension of the *Aggregated Channel Features* (ACF) detector of Dollár *et al.* (2014). Given an image, the ACF detector computes ten channel features. These channels are the components of the *CIELUV* color space, the normalized gradient magnitude, and the histogram of oriented gradients using six orientation bins (Dollár *et al.*, 2009a). After computing all channels, they are downsampled by a factor of four using average pooling and smoothed by means of a binomial filter. The pixels of the subsampled channels form the feature pool used to train a boosted decision tree classifier. The resulting boosted classifier is transformed into a soft-cascade classifier (Bourdev and Brandt, 2005), enabling a quicker rejection of negative samples. To detect all pedestrians in an image, the classifier is applied in a sliding window manner over multiple scales using the fast pyramid construction technique of Dollár *et al.* (2014).

The ACF detector was extended to the multispectral domain by Hwang *et al.* (2015). In the remainder of this work, we will refer to the multispectral extension of the ACF detector as *multispectral ACF*. The design of the ACF detector allows an easy extension to the multispectral input data of the *KAIST* benchmark by adding additional feature channels encoding the thermal image. These thermal feature channels are a contrast-enhanced version of the thermal image as well as HOG features (Dalal and Triggs, 2005) of the thermal image. The *multispectral ACF* detector used in our experiments is a retrained version of the detector provided by the toolbox of the *KAIST* benchmark (Hwang *et al.*, 2015). As suggested by Dollár *et al.* (2012), we additionally standardize all bounding boxes to an aspect-ratio of 0.41 by keeping the height and center fixed and only adjusting the width.

2.4.3 Results

To compare the different detectors, we report their performance on the *reasonable all*, *reasonable day*, and *reasonable night* subset of the *KAIST* test data (see Section 2.4.1). We follow the evaluation protocol defined by Hwang *et al.* (2015) and additionally standardize in accordance with Dollár *et al.* (2012) the aspect ratio of bounding boxes to a fixed value of 0.41, by means of width adjustment. In Figure 2.6, we visualize the miss rate against false positives per image of the detectors for all three evaluation subsets. Table 2.1 reports the corresponding log-average miss rates. Further details and discussion of the metrics can be found in Dollár *et al.* (2012).

We report the performance of four versions of our multispectral pedestrian detector as well as the performance of the baseline detector (*multispectral ACF*). All deep learning-based detectors are named after the used CNN. *LateFusion*, for example, references a detector according to the description in Section 2.3.1 which uses a late-fusion-based CNN

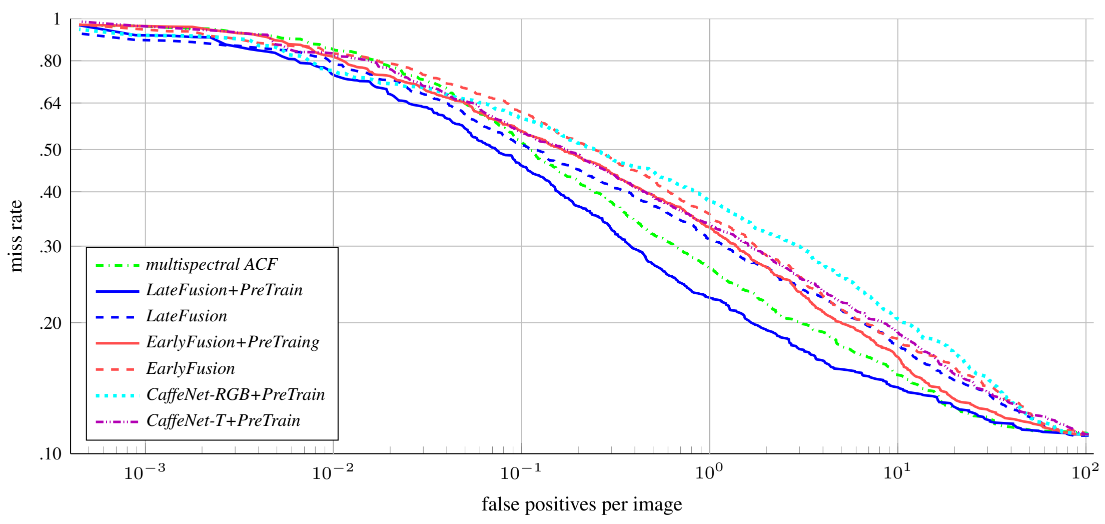
Detector	**log-average miss rate**		
	all	*day*	*night*
Multispectral ACF	$50.48\,\%$	$51.27\,\%$	$47.40\,\%$
LateFusion+PreTrain	$\mathbf{43.80\,\%}$	$\mathbf{46.15\,\%}$	$\mathbf{37.00\,\%}$
LateFusion	$51.30\,\%$	$55.27\,\%$	$41.58\,\%$
EarlyFusion+PreTrain	$53.94\,\%$	$50.90\,\%$	$51.76\,\%$
EarlyFusion	$57.96\,\%$	$58.22\,\%$	$57.78\,\%$
CaffeNet-RGB+PreTrain	$56.52\,\%$	$53.51\,\%$	$63.40\,\%$
CaffeNet-T+PreTrain	$54.67\,\%$	$59.77\,\%$	$42.09\,\%$

Table 2.1: Log-average miss rate on the *reasonable all*, *reasonable day*, and *reasonable night* evaluation split. All deep learning-based detectors, which are additionally pre-trained on auxiliary data, are indicated by the suffix *PreTrain*. All detectors besides *CaffeNet-RGB+PreTrain* and *CaffeNet-T+PreTrain* operate on multispectral input data. The CNN of the *CaffeNet-RGB+PreTrain* detector only uses visible images, the CNN of the *CaffeNet-T+PreTrain* detector only thermal images. The *LateFusion+PreTrain* detector outperforms all other detectors on all three evaluation subsets.

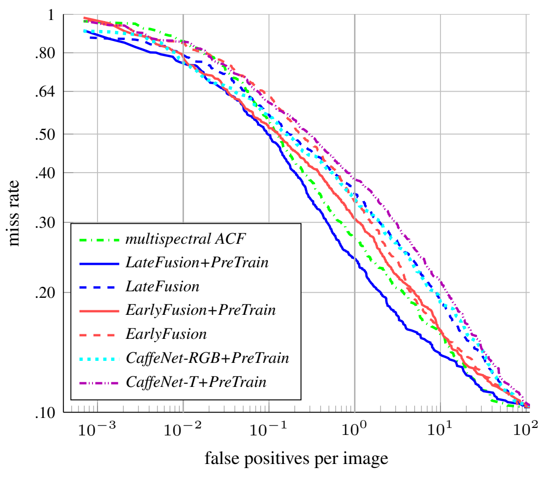
architecture to classify proposals. The additional suffix *PreTrain* marks detectors which utilize a CNN that is pre-trained on auxiliary datasets (see Section 2.3.3). In addition, we evaluate the performance of two detectors whose CNN only uses one sensor modality. These detectors use the two late-fusion subnetworks *CaffeNet-RGB* and *CaffeNet-T*, respectively, which are pre-trained on the *ImageNet* as well as *Caltech* dataset and then fine-tuned on the *KAIST* dataset. Following our naming convention, they are referenced by *CaffeNet-RGB+PreTrain* and *CaffeNet-T+PreTrain*.

Comparing the performance of *CaffeNet-RGB+PreTrain* and *CaffeNet-T+PreTrain* on the *reasonable day* and *reasonable night* subset highlights the complementary strength of the two sensor modalities. At night, when the lighting conditions are rather poor, the detector using the thermal CNN (*CaffeNet-T+PreTrain*) achieves a log-average miss rate of $42.09\,\%$, outperforming *CaffeNet-RGB+PreTrain* by $21.31\,\%$. However, on the *reasonable day* subset which was recorded at daytime, the model using the visible image (*CaffeNet-RGB+PreTrain*) yields a superior performance. *CaffeNet-RGB+PreTrain* achieves a log-average miss rate of $53.51\,\%$, while *CaffeNet-T+PreTrain* only achieves a rate of $59.77\,\%$. Due to their complementary strengths, a fusion of both modalities is thus a promising approach to construct a more robust detector.

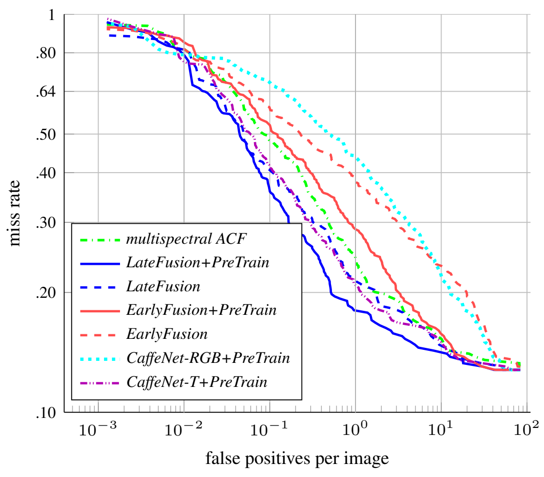
Even though during pre-training the substitution of the thermal channel with the red channel is a very crude approximation, it leads to a significant performance increase for all detectors. The early-fusion-based detector increases its performance by $4.02\,\%$ on



(a) *Reasonable all* evaluation setting



(b) *Reasonable day* evaluation setting



(c) *Reasonable night* evaluation setting

Figure 2.6: Miss rate against false positives per image for the *reasonable all*, *reasonable day*, and *reasonable night* evaluation split. All deep learning-based detectors, which are additionally pre-trained on auxiliary data, are indicated by the suffix *PreTrain*. All detectors besides *CaffeNet-RGB+PreTrain* and *CaffeNet-T+PreTrain* operate on multispectral input data. The CNN of the *CaffeNet-RGB+PreTrain* detector only uses visible images, the CNN of the *CaffeNet-T+PreTrain* detector only thermal images.

the *reasonable all* evaluation split and the late-fusion-based detector even achieves an increase of $7.5\,\%$. In the remainder of the evaluation, we will therefore primarily focus on the detectors using pre-trained architectures.

The detector using a pre-trained late-fusion-based CNN (*LateFusion+PreTrain*) significantly outperforms the state-of-the-art *multispectral ACF* baseline, as well as all other evaluated detectors in all three test set splits. During the day, the performance of the

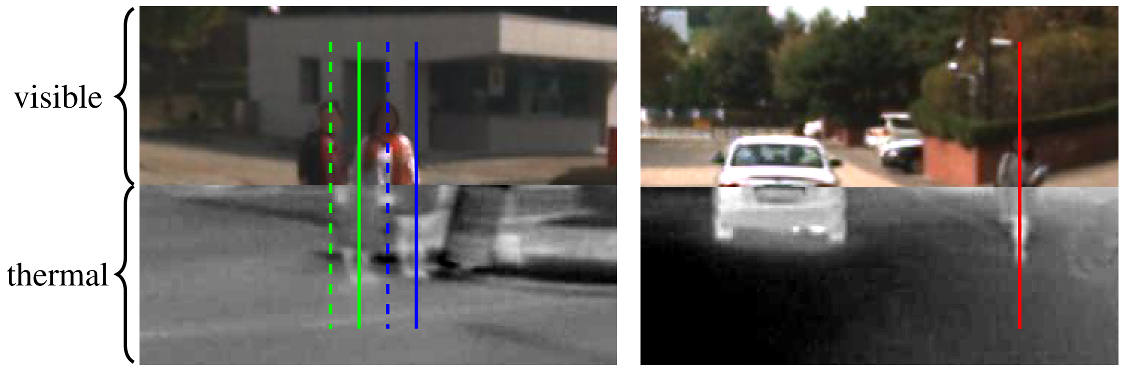


Figure 2.7: Non-systematic alignment error occurring in a subset of the multispectral *Kaist* data (Hwang *et al.*, 2015). For two images, we visualize in the upper half the data of the visible camera and in the lower half the data of the thermal camera. For each pedestrian two vertical lines indicate the position of the center of the pedestrian. A dashed line is aligned according on the position of the pedestrian in the visible image, a solid line indicates the center of the pedestrian in the thermal image. Both modalities are perfectly aligned in the right image. In the left image there is a large horizontal offset between both modalities.

LateFusion+PreTrain architecture is $5.12\,\%$ better than the performance of the baseline, and at night it is $10.4\,\%$ better. The performance of our best multispectral detector is particularly superior to the two single modality detectors, which underlines the advantage of sensor data fusion.

For most of the evaluation settings, the detectors using the early-fusion architecture are not able to reach state-of-the-art *multispectral ACF* detector performance. In our opinion, there are at least three reasons for this observation: The poor performance of the early-fusion model without pre-training can most likely be traced back to the limited amount of data available in the *KAIST* dataset. Additionally, we suspect that the early-fusion network did not learn meaningful multimodal features during the pre-training process, due to the absence of complementary information in the chosen approximation of the thermal channel. The architecture of the late-fusion CNN is due to its two subnetworks better suited for pre-training on only visible data. A third reason for the poor performance of the early-fusion architectures is a small non-systematic alignment error, which is noticeable in some images of the dataset. This non-systematic offset complicates learning of meaningful multimodal low-level features. In Figure 2.7, we exemplary show two images of the *KAIST* dataset, one in which the modalities are perfectly aligned and one with a relatively large alignment error. The late-fusion network is better able to cope with such alignment errors, because it fuses information at a stage where spatial information is less relevant.

2.5 Conclusion

A robust detection of pedestrians is a crucial capability of autonomous vehicles. Pedestrian detectors which rely only on images of the visible spectrum suffer from low or rapidly changing lighting conditions. By using the information of multiple sensors with complementary strengths, the robustness of a detection system can be significantly increased. In this chapter, we investigated the benefits of multispectral pedestrian detectors which utilize the information of a visible and thermal camera. To fuse the information of both modalities, we proposed two deep fusion architectures. An early-fusion architecture which fuses the information at pixel-level and a late-fusion architecture which performs fusion at feature-level. We proposed a multi-step pre-training procedure that relies solely on visible images to cope with the limited availability of multispectral training data. As proposed by the R-CNN framework (Girshick *et al.*, 2014), the deep fusion architectures were combined with a proposal generator to construct a multispectral detector. To the best of our knowledge, this is the first work to use a deep learning-based approach for multispectral pedestrian detection.

Our analysis on the *KAIST* multispectral pedestrian detection benchmark dataset (Hwang *et al.*, 2015) showed that a CNN-based detector using both the visible and thermal image can significantly outperform models relying solely on one of the two modalities. Our best detector, which uses a pre-trained late-fusion architecture, outperformed the current state-of-the-art multispectral detector (Hwang *et al.*, 2015) on the *KAIST* benchmark dataset by more than $6.5\,\%$. Pre-training on auxiliary datasets was an important factor in achieving a good performance. In our experiments, the late-fusion CNN has proven to be superior to the early-fusion CNN. This is most likely due to the inability of the early-fusion network to learn meaningful multimodal features in the given setting as well as due to the poor spatial alignment of several of the benchmark images.

Following the publication of the contents of this chapter (Wagner *et al.*, 2016), several improvements have been proposed to increase the performance of multispectral pedestrian detectors. Numerous works investigate new fusion approaches and detection concepts. Liu *et al.* (2016a) use the Faster R-CNN detection framework, which is an extension of the R-CNN framework used in this work. In total they evaluate four fusion architectures and show that their *Halfway Fusion* architecture achieves the best performance on the *KAIST* benchmark. Similarly to our *LateFusion* model, their best architecture uses two subnetworks that process the visible and thermal image separately. In contrast to our *LateFusion* model, fusion of the modalities already takes place on mid-level convolutional features (*i.e.* after the fourth block of convolutional layers of the used *VGG16* (Simonyan and Zisserman, 2015) network). To combine the feature maps of the visible and thermal image they utilize a fusion module consisting of a concatenation layer and a 1×1 convolutional layer. König *et al.* (2017) use the *Region Proposal Network* of the Faster R-CNN framework and combine it with a *Boosted Decision Trees* classifier to construct a multispectral detector. They adopt the *Halfway Fusion* architecture of Liu *et al.* (2016a) and show an improved performance when fusing the features of both modalities after the third block of convolutional layers. Inspired by our work, they employ a multi-stage training procedure using auxiliary datasets. Related multispectral detection approaches are introduced by Lee *et al.* (2018) and Li *et al.* (2018). Lee *et al.* (2018) adopt the *Halfway Fusion* approach, enrich it by additional multi-spectral features and integrate it into a *Deconvolutional Single Shot Detector* (Fu *et al.*, 2017). Utilizing findings from previous multi-spectral and visible detectors, Li *et al.* (2018) propose the *Multispectral Simultaneous Detection and Segmentation R-CNN* (MSDS-RCNN). This model uses a two-stage cascade consisting of a multi-spectral proposal generator and a multi-spectral classification network. The architecture of both networks is similar to the CNN proposed by König *et al.* (2017). To further improve the performance of the model, they propose to simultaneously solve a semantic segmentation task and to integrate additional single-modality predictions. In total, their model generates four classification scores for each bounding box proposal, which are fused at score level. More explicit fusion approaches were introduced by Guan *et al.* (2019) and Li *et al.* (2019) using an illumination-aware weighting mechanism to fuse the information of two subnetworks. The illumination value used for weighting is predicted by a neural network which is trained in a supervised manner using the coarse day / night labels of the *KAIST* dataset. To better cope with the alignment error shown in Section 2.4.3, Zhang *et al.* (2019) and Zhou *et al.* (2020) introduce modules which estimate the error and adaptively align the feature maps of the sensor modalities. Zhang *et al.* (2019) regionally align feature maps of both modalities by predicting and correcting a horizontal and vertical shift for each proposal. To increase the robustness of the module they additionally introduce an augmentation strategy which randomly shifts the modalities during training. Zhou *et al.* (2020), on the other hand, predict a horizontal and vertical offset for each pixel and use bilinear interpolation to align the modalities.

In addition to sensor data fusion, temporal integration of information is a promising approach for improving the robustness of perception systems. We examine corresponding methods in Chapter 4 and Chapter 5.

3 Semantic Forecasting Models

The ability to predict future environment states is essential for an anticipatory behavior of autonomous agents, such as self-driving cars or mobile robots. Deep learning-based methods have shown impressive results on key perception challenges but currently mainly operate in a non-predictive fashion. We bridge this gap by proposing an approach to transform trained non-predictive feed-forward networks into predictive ones via a combination of a recurrent predictive module with a teacher-student training strategy. Our proposed predictive transformation can be conducted without the need of labeled video data in a fully self-supervised fashion. This is especially beneficial for models which generate dense and detailed representations, like a pixel-wise semantic segmentation, as manual labeling is quite time-consuming and expensive.
Using simulated video data and focusing on the use case of semantic segmentation, we evaluate the ability of our approach to generate models which forecast the future environment representation conditioned on previous observations. Our results show the ability of the resulting predictive network to model the dynamics of the world. Additionally, we show the benefits of using our approach in a semi-supervised setting, when no appropriate non-predictive feed-forward model is available.

3.1 Introduction

Deep learning-based methods yield impressive results in application domains such as speech recognition, computer vision, and machine translation. Especially in visual perception, deep convolutional neural networks dominate the majority of benchmarks (Dollár *et al.*, 2009b; Geiger *et al.*, 2012; Cordts *et al.*, 2016). Due to their ability to model complex dependencies and to generalize to unseen examples, they have the potential to solve key challenges of autonomous vehicles, such as self-driving cars or mobile robots. Among these challenges are the robust perception and interpretation of the environment as well as the prediction of future environment states in order to enable acting in an anticipatory way. A pixel-wise semantic segmentation constitutes a detailed and rich representation of the visually observed environment. Such a semantic representation contains useful information for the planning and decision-making component of an autonomous vehicle, such as the presence of other traffic participants, the position of lane markings, or the extent of the drivable area. Deep learning-based models for semantic segmentation gained increasing interest in recent years (Garcia-Garcia *et al.*, 2018), leading to a rapid improvement of segmentation performance on common benchmark datasets (Brostow *et al.*, 2009; Cordts *et al.*, 2016). These advances bring the autonomous vehicle community closer to a robust and reliable perception of the environment. However, to cope with the dynamic nature of the world (*e.g.* moving traffic participants), a planner of a safety-critical system must take into account not only the current but also the future environment state (Bahram, 2017). For example, in the situation depicted in Figure 3.1, the autonomous vehicle has to decide whether to break or to drive on. The semantic representation of the current time step provides the information that a pedestrian is in the vicinity of a crosswalk located on the current path of the autonomous vehicle. By predicting the environment state into the future, the system can infer the intent of the pedestrian to cross the street in front of the vehicle and initiate a deceleration.

In this chapter, we focus on the prediction of the future environment representation conditioned on previous observations. We build upon the recent success of deep convolutional neural networks for visual perception and propose an approach to transform these non-predictive feed-forward networks[1] into ones which predict the future. Based on the

[1]For simplicity, we will refer to non-predictive feed-forward networks — *i.e.* models which receive a measurement $\mathbf{x}_t$ of the current time step t and generate a corresponding application specific target $\mathbf{y}_t = f(\mathbf{x}_t; \theta)$ (see Section 3.3.1 for more details) — as feed-forward networks in the remainder of this chapter.

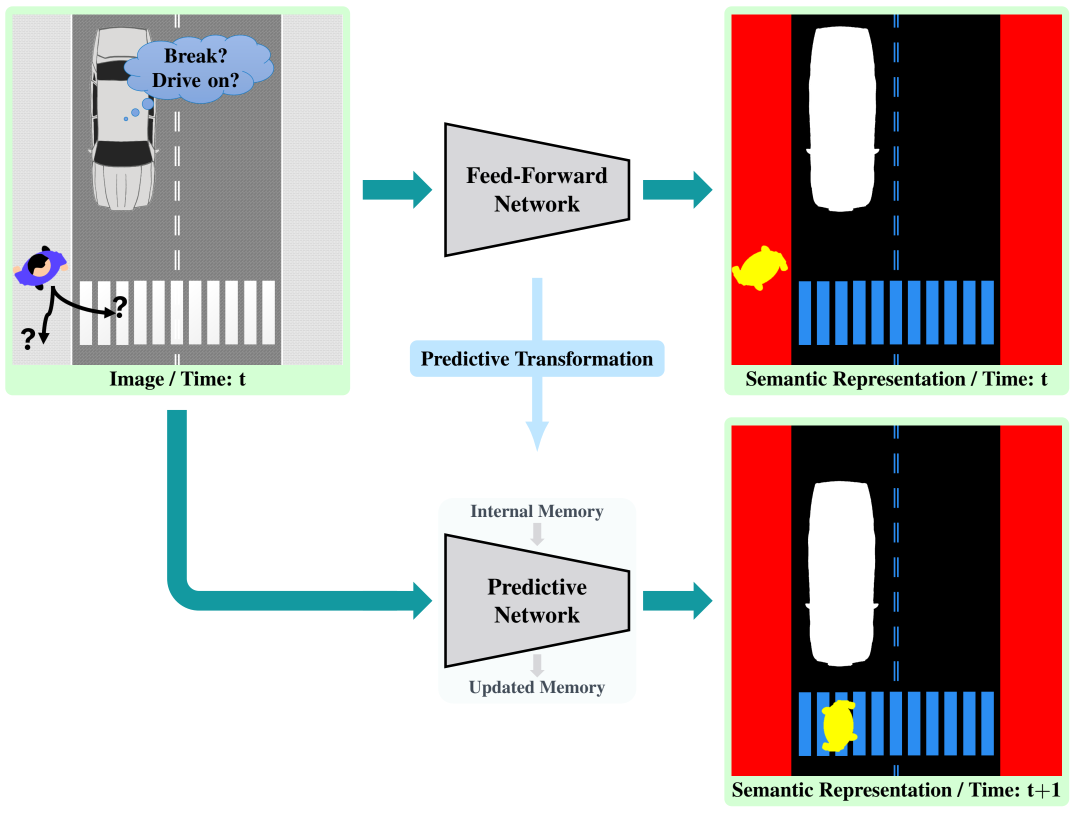


Figure 3.1: Future autonomous vehicles have to act in an anticipatory manner in order to facilitate a safe deployment in open-world scenarios. To enable a proactive planning of actions, we propose an approach to transform a given feed-forward network into a predictive one. The predictive transformation is achieved by introducing an additional recurrent module and applying a teacher-student-like training strategy. The predictive transformation can thus be conducted without the need of labeled data in a fully self-supervised fashion. In our experiments, we use a pixel-wise semantic segmentation to represent the observed environment in a semantic manner.

use case of pixel-wise semantic segmentation, we present and evaluate our predictive transformation approach. The resulting semantic forecasting model predicts the semantic segmentation of the next time step conditioned on previous observed image frames. Since our proposed method is applicable to a wide range of feed-forward models, we describe the transformation in general terms and highlight the specifics that apply to pixel-wise semantic segmentation models.

The predictive network generated by our approach reuses the original structure of the provided feed-forward model an extends it by a recurrent predictive module. The resulting predictive network conceptually decouples temporal prediction from scene representation. We thus can reuse the already well-trained weights and the encoded task-specific knowledge of the original feed-forward model. Training of the predictive model is conducted in a self-supervised fashion using a teacher-student-like training strategy. This eliminates the need for labeling video data, which is time-consuming and costly, especially in the case of semantic segmentation. In total, we propose four knowledge transfer variations and systematically evaluate their performance using simulated video data. Our experiments quantitatively and qualitatively demonstrate the ability of our approach to create semantic forecasting models using unlabeled video data which capture the dynamics of the world (*e.g.* the interaction and motion of objects). We additionally demonstrate that our approach is beneficial in cases when no feed-forward network is available and one is tasked with training a predictive model based on video data of which only a small portion is labeled (*i.e.* a semi-supervised setting (Ouali *et al.*, 2020)). In such a setting, we first train a non-predictive feed-forward model using the labeled data and then transform it to a predictive one using the unlabeled data. A final fine-tuning of the predictive model based on the labeled video data can further improve model performance.

The method presented in this chapter was first published at the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (Wagner *et al.*, 2017).

3.2 Related Work

The majority of approaches which predict a representation of the future environment state either focus on the prediction of raw sensory data (Ranzato *et al.*, 2014; Michalski *et al.*, 2014; Srivastava *et al.*, 2015; Mathieu *et al.*, 2016) or model the world in an object-centric manner (Kruse *et al.*, 2013; Lefèvre *et al.*, 2014; Brouwer *et al.*, 2016).

Object-centric approaches predict the future state or the future trajectory of an object (*e.g.* a pedestrian or car) conditioned on observations of previous states. Elnagar (2001), for example, uses a physics-based motion model (Lefèvre *et al.*, 2014) to predict the future position and orientation of an object. To better account for different behavior patterns, Schneider and Gavrila (2013) combine several physics-based motion models using the *Interacting Multiple Model* (IMM) (Blom, 1984) algorithm. Other methods use observed data to learn models which approximate object dynamics (Goldhammer *et al.*, 2014) or apply a planner-based prediction approach (Ziebart *et al.*, 2009). By taking into account additional factors which influence the complex dynamics of real-world objects, the prediction performance can be further improved. Alahi *et al.* (2016) propose the *Social-LSTM* model which additionally takes into account the interaction of pedestrians. They model each pedestrian using a *Long Short-Term Memory* (LSTM) cell and use a social pooling strategy to enable information exchange between cells. Their qualitative evaluation highlights the ability of the *Social-LSTM* model to create predictions with plausible interaction patterns. In addition to object-object interactions, modeling the interactions between objects and the static environment is an additional crucial factor. The static environment is usually represented either as individual static objects (Pellegrini *et al.*, 2009) or as a grid-like, potentially semantic map (Rehder and Kloeden, 2015; Karasev *et al.*, 2016). The already mentioned planning-based prediction approach of Ziebart *et al.* (2009), for example, utilizes features derived from a map of the static environment.

Compared to these object-centric prediction approaches, which are usually limited to one class of objects (*e.g.* pedestrians), our approach predicts a more dense and semantically rich representation of the environment. Furthermore, since our method directly operates on image data, there is no need for a separate perception component to extract objects. Our model covers the perception and prediction step of a classical processing pipeline and optimizes them jointly, in an end-to-end fashion[2]. Interactions between objects as well as objects and the static environment are directly learned from data, without having to design hand-crafted interaction features or an explicit interaction model (Luber *et al.*, 2010; Yi *et al.*, 2015).

Deep learning-based models that predict raw sensory data have become increasingly popular in recent years. These models can be trained without requiring additional object detection methods or labeled data. They make use of the sequential nature of sensor streams and directly predict the next measurement or the future sequence of measurements given the history of previous measurements. Such models have been used to predict the next image in a video (Ranzato *et al.*, 2014; Michalski *et al.*, 2014; Srivastava *et al.*, 2015; Lotter *et al.*, 2016; Mathieu *et al.*, 2016), to predict radar measurements (Xingjian *et al.*, 2015), and to predict and filter future processed laser scans (Ondruska and Posner, 2016). Many predictive models incorporate recurrent structures to process sequential data. Ranzato *et al.* (2014) draw inspiration from concepts used in language model-

[2]We propose and evaluate several knowledge transfer versions. Not all of them jointly optimize all weights of the predictive network. For more details, see Section 3.3.3.

ing and propose the *Recurrent Convolutional Neural Network* (rCNN) to predict future video frames. Srivastava *et al.* (2015) use LSTM cells to construct an encoder-decoder architecture which predicts the sequence of future frames. To improve prediction performance, they additionally add a second LSTM decoder which reconstructs the input sequence. Furthermore, they condition the LSTM decoder on the last generated frame. Xingjian *et al.* (2015) propose the convolutional LSTM, an extension of the classical LSTM with a convolutional structure, and use it to build an encoder-decoder architecture similar to Srivastava *et al.* (2015). Non-recurrent model architectures have been employed by Mathieu *et al.* (2016) and Michalski *et al.* (2014). The model by Mathieu *et al.* (2016) predicts several concatenated future frames conditioned on a concatenated version of previous frames using a convolutional multi-scale architecture. To optimize the model weights, they use an adversarial training method combined with an image gradient difference loss. Michalski *et al.* (2014) use bi-linear models to infer transformations (*i.e.* relational features) between two consecutive observations and utilize the inferred transformations to forward propagate the second observation in time. Building on this concept, they construct the multi-layer *Predictive Gating Pyramid* (PGP) by leveraging higher-order relational features and applying the prediction in a recursive manner to forecast multiple steps into the future.

Compared to these approaches, our forecasting network predicts a more abstract, high-level representation of the environment (*i.e.* a pixel-wise semantic segmentation), which is well suited for autonomous driving tasks. In most applications, one is more interested in predicting such an application-specific representation of the world rather than the next raw measurement. An autonomous robot tasked with avoiding the collision with people, for example, has no additional benefit from knowing the future raw measurement compared to the future position of all people. To implement a planner for an autonomous vehicle, a second perception model is needed that derives an application-specific representation from the predicted raw measurement. The overall number of parameters and the computational cost of such a two-model approach is almost certainly much higher compared to a model which directly predicts an application-specific representation. The prediction task is additionally more complex, as the model has to reconstruct the whole measurement of the next time step, which includes additional information not required to derive a task-specific environment representation.

Vondrick *et al.* (2016) propose a model which predicts a more abstract, high-level representation. They learn a general-purpose predictive feature extractor in a self-supervised manner and use an off-the-shelf or an adapted classifier to predict future actions or object categories. To train the adapted classifier, labeled data is required. Our method, on the other hand, creates a task-specific predictive encoder and we evaluate options to jointly optimize the overall predictive network in a self-supervised manner using unlabeled data. Their method assumes that a general-purpose feature extractor network and labeled data are available, our methods is based on the availability of a task-specific feed-forward model. An additional difference is the prediction target of the models. Vondrick *et al.* (2016) focus on predicting a single class (*e.g.* action class), which differs from the spatially detailed representation of the environment created by our model. These dissimilarities also manifest in the different design of the recurrent structures of the models.

A related field of research focuses on temporally stabilizing (temporally filtering) the prediction of a model using information of previous time steps (Zhang *et al.*, 2014; Pavel *et al.*, 2015; Patraucean *et al.*, 2015; Fayyaz *et al.*, 2016; Kundu *et al.*, 2016). Such methods perform a prediction for the current time step conditioned on the measurements of the current as well as all previous time steps. In Chapter 4 and Chapter 5, we propose two such models and additionally highlight related approaches from literature.

3.3 Predictive Transformation

To convert a trained feed-forward neural network into a predictive one, we transform it into a recurrent network by introducing an additional recurrent predictive module. Our proposed predictive module is based on a convolutional LSTM cell (Xingjian *et al.*, 2015) and propagates an abstract feature representation of the feed-forward model one time step into the future. The weights of the newly generated predictive model are optimized using a teacher-student-like (Bucilua *et al.*, 2006; Ba and Caruana, 2014; Hinton *et al.*, 2014) training approach. In total, we evaluate two loss options for training, a loss which enforces a similarity between the predicted feature representation of the student and the corresponding feature representation of the teacher network and a loss that enforces a similarity between the task-specific output of the teacher and student network.

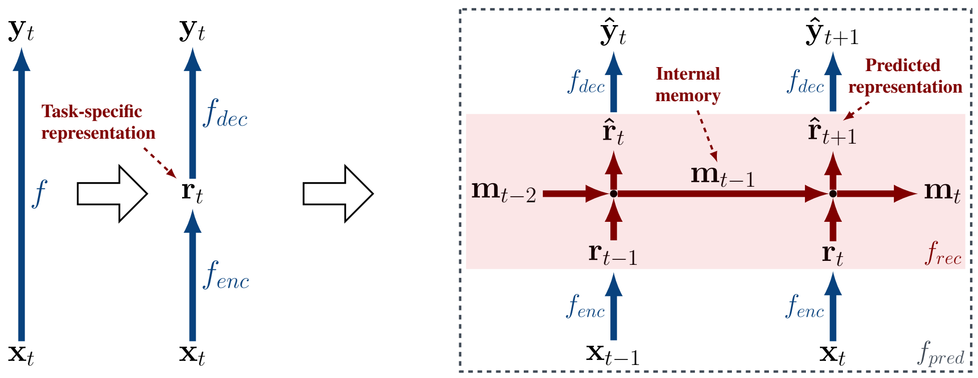


Figure 3.2: Proposed predictive transformation used to convert a trained feed-forward neural network f into a predictive network f_{pred}. In a first step, we split the model into two parts, an encoder network f_{enc} and a decoder network f_{dec}. To temporally propagate the feature representation $\mathbf{r}_t$ of the encoder, a recurrent predictive module f_{rec} is introduced. The resulting predictive network architecture f_{pred} thus consists of the original feed-forward model (f_{enc} and f_{dec}) depicted in blue and the inserted recurrent predictive module (f_{rec}), which is highlighted in red. The predicted feature representations and targets are each marked by a hat accent above the variable name.

In Section 3.3.1, we describe the structural transformation of the feed-forward neural network into a predictive model in general terms. Section 3.3.2, further elaborates on the recurrent module that has been inserted to temporally predict an abstract feature representation of the feed-forward model. In Section 3.3.3, we present our employed learning strategy which operates in a self-supervised fashion without requiring hand-labeled training data.

3.3.1 Predictive Network Architecture

We assume that a well-trained feed-forward neural network f with parameters θ is given, which receives a measurement $\mathbf{x}_t$ of the current time step t and generates a corresponding application specific target $\mathbf{y}_t = f(\mathbf{x}_t; \theta)$. Although this chapter focuses on networks which produce a pixel-wise semantic segmentation $\mathbf{y}_t$ using the information of an image $\mathbf{x}_t$, our approach is viable for feed-forward networks in general[3]. The measurement $\mathbf{x}_t$ can originate from different sensors such as cameras, laser scanners or even a group of sensors (see Section 2), and the target can be chosen arbitrarily. However, the target representation may influence the choice of training loss. A more detailed discussion of loss options is given in Section 3.3.3.

[3]The proposed approach is viable for any feed-forward neural network that allows the structural transformation described in this section — *i.e.* the model has to be divisible into two parts to create an abstract representation that can be predicted in time.

To transform a model f into a predictive one, we first split it into two parts (see Figure 3.2, left sides) — a lower network part f_{enc}, which we will refer to as feature encoder, and an upper part f_{dec}, the so-called decoder. The encoder network f_{enc} receives the measurement $\mathbf{x}_t$ and generates an abstract task-specific feature representation $\mathbf{r}_t = f_{enc}(\mathbf{x}_t; \theta_{enc})$. The decoder network f_{dec} computes the target $\mathbf{y}_t = f_{dec}(\mathbf{r}_t; \theta_{dec})$, given the representation $\mathbf{r}_t$ generated by the encoder. All parameters of the encoder and decoder are specified by θ_{enc} and θ_{dec}, respectively. To propagate the abstract feature representation $\mathbf{r}_t$ one time step into the future we utilize a recurrent predictive module f_{rec} with parameters θ_{rec}:

$$\begin{aligned}\hat{\mathbf{r}}_{t+1} &= f_{rec}(\mathbf{r}_t, \mathbf{m}_{t-1}; \theta_{rec}) \\ &= f_{rec}(f_{enc}(\mathbf{x}_t; \theta_{enc}), \mathbf{m}_{t-1}; \theta_{rec}). \end{aligned} \tag{3.1}$$

This module predicts the expected representation of the next time step $\hat{\mathbf{r}}_{t+1}$ based on the current representation $\mathbf{r}_t$ as well as a hidden state $\mathbf{m}_{t-1}$. The hidden state represents the knowledge aggregated from representations of previous time steps. To generate the predicted target $\hat{\mathbf{y}}_{t+1}$ of time step $t+1$, the predicted representation $\hat{\mathbf{r}}_{t+1}$ is passed through the decoder network f_{dec}:

$$\begin{aligned}\hat{\mathbf{y}}_{t+1} &= f_{dec}(\hat{\mathbf{r}}_{t+1}; \theta_{dec}) \\ &= f_{dec}(f_{rec}(f_{enc}(\mathbf{x}_t; \theta_{enc}), \mathbf{m}_{t-1}; \theta_{rec}); \theta_{dec}). \end{aligned} \tag{3.2}$$

The split of the network has to be chosen task-specifically and is a hyperparameter of our approach. Based on our experience, a split at a layer producing higher-level (more abstract) features is a favorable choice, considering that the temporal prediction of abstract scene properties is generally easier compared to the prediction of detailed low level features. The right side of Figure 3.2 depicts the overall predictive network architecture f_{pred}:

$$\hat{\mathbf{y}}_{t+1} = f_{pred}(\mathbf{x}_t, \mathbf{m}_{t-1}; \theta_{enc}, \theta_{rec}, \theta_{dec}), \tag{3.3}$$

consisting of the original feed-forward network drawn in blue as well as the inserted recurrent module highlighted in red.

The resulting predictive network conceptually decouples temporal prediction (subnetwork f_{rec}) from scene representation (subnetwork f_{enc} and f_{dec}). This property makes it possible to reuse the trained weights $\theta = \theta_{enc} \cup \theta_{dec}$ of the original feed-forward model f.

3.3.2 Recurrent Predictive Module

The recurrent predictive module is tasked with predicting a representation of a feed-forward network one time step into the future. Thus, it has to learn the dynamics of the representation based on a sequence of previous representations to predict the expected next state. A natural model choice to implement such a behavior are *Recurrent Neural Network*s (RNNs). *Long Short-Term Memory* (LSTM) cells as a variant of RNNs have proven to produce state-of-the-art results in a variety of sequence modeling tasks (Graves, 2014). Due to the spatial and local nature of the data used in this work, we use a convolutional LSTM as suggested by Xingjian *et al.* (2015) to implement the predictive module f_{rec}. The recurrent part of the predictive module is thus defined by:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ri} * \mathbf{r}_t + \mathbf{W}_{hi} * \mathbf{h}_{t-1} + \mathbf{W}_{ci} \circ \mathbf{c}_{t-1} + \mathbf{b}_i), && (3.4)\\
\mathbf{f}_t &= \sigma(\mathbf{W}_{rf} * \mathbf{r}_t + \mathbf{W}_{hf} * \mathbf{h}_{t-1} + \mathbf{W}_{cf} \circ \mathbf{c}_{t-1} + \mathbf{b}_f), && (3.5)\\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ro} * \mathbf{r}_t + \mathbf{W}_{ho} * \mathbf{h}_{t-1} + \mathbf{W}_{co} \circ \mathbf{c}_t + \mathbf{b}_o), && (3.6)\\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbf{W}_{rc} * \mathbf{r}_t + \mathbf{W}_{hc} * \mathbf{h}_{t-1} + \mathbf{b}_c), && (3.7)\\
\mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t), && (3.8)
\end{aligned}$$

where $*$ is the convolutional operator and $\circ$ the *Hadamard* product. The input, forget, and output gate are denoted by $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$, respectively. For brevity, we will summarize the cell state $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$ by $\mathbf{m}_t$ in the remainder of this chapter. To obtain the predicted representation $\hat{\mathbf{r}}_{t+1}$, we pass the cell state $\mathbf{c}_t$ through an additional convolutional layer:

$$\hat{\mathbf{r}}_{t+1} = f_{rec}(\mathbf{r}_t, \mathbf{m}_{t-1}; \theta_{rec}) = \phi(\mathbf{W}_{cr} * \mathbf{c}_t + \mathbf{b}_{cr}). \tag{3.9}$$

All trainable weights of the predictive module f_{rec} are summarized by the variable θ_{rec}. As nonlinearity ϕ, we use a leaky ReLU (Maas *et al.*, 2013) with a negative slope coefficient of 0.01. The weight tensor $\mathbf{W}_{cr}$ of the final convolutional layer has to be chosen

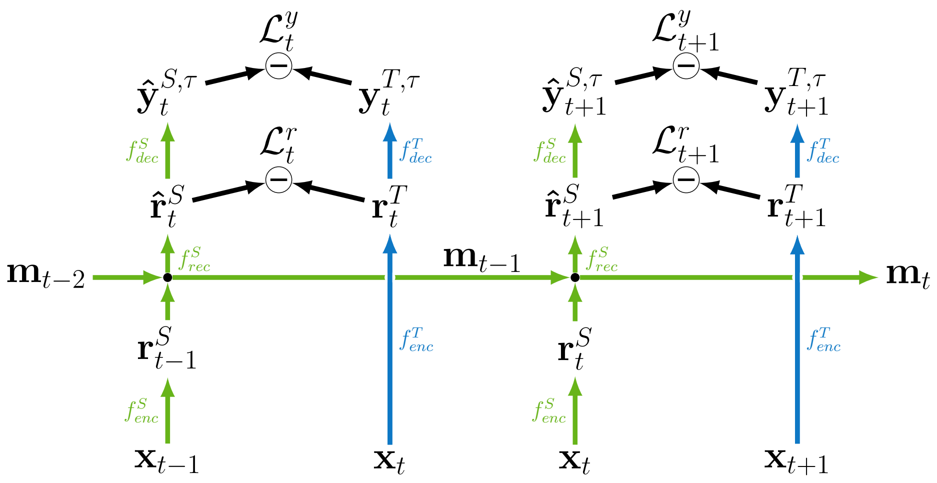


Figure 3.3: Teacher-student structure used for self-supervised training. The teacher network is depicted in blue and marked by a superscript *T*. The student network is drawn in green and indicated by the superscript *S*. We use the trained feed-forward model as the teacher network and apply it to the measurement of the next time step. The student network mimics the teacher, given the current measurement as well as the internal memory. We evaluate two loss options, a loss $\mathcal{L}^r$ defined on representation level and a loss $\mathcal{L}^y$ defined on target level.

in such a way that the predicted representation $\hat{\mathbf{r}}_{t+1}$ and the input representation $\mathbf{r}_t$ have equal dimensions (*i.e.* the number of convolutional filters has to be identical to the number of feature maps in $\mathbf{r}_t$). The prediction capabilities of the proposed recurrent module can easily be scaled up by stacking multiple convolutional LSTM cells and / or by using an encoder-decoder-like structure as proposed by Xingjian *et al.* (2015).

3.3.3 Predictive Knowledge Transfer

We use the teacher-student paradigm (Bucilua *et al.*, 2006; Ba and Caruana, 2014; Hinton *et al.*, 2014) to train the predictive network architecture in a self-supervised manner without the need of labeled sequence data. This paradigm is commonly used to compress a well-trained deep neural network into a smaller network by using the prediction of the teacher as a supervision signal for the student. In our setting, the well-trained feed-forward network f serves as the teacher to train the more complex predictive model f_{pred} — the student network. A similar setting, where a simple teacher model is used to train a more complex student model, was investigated by Tang *et al.* (2016).

To generate a predictive supervision signal, we provide the teacher with the measurement of the next time step and force the student network to mimic the teacher given the current measurement as well as the internal memory. The resulting teacher-student training structure is depicted in Figure 3.3, with the teacher network illustrated in blue and the student network in green. The trained weights of the teacher, which we will denote as $\theta^T = \{\theta^T_{enc}, \theta^T_{dec}\}$, are fixed during the training procedure. The weights of the student $\theta^S_{pred} = \{\theta^S_{enc}, \theta^S_{rec}, \theta^S_{dec}\}$ are optimized using the supervision signal of the teacher model. In addition, we initialize all weights of the student which are not part of the predictive module (*i.e.* θ^S_{enc} and θ^S_{dec}) with the corresponding weights of the teacher. Since the initial weights θ^S_{enc} and θ^S_{dec} are thus already well-trained, one could also fix them and only optimize the weights θ^S_{rec} of the recurrent module. Both training options are evaluated in the experiments of Section 3.4. To compute the gradient with respect to the weights, we use *Backpropagation Through Time* (BPTT) (Werbos, 1990) and unroll the recurrent network for N time steps.

We evaluate two losses to train the predictive student network: a loss $\mathcal{L}^r$ defined on representation level and a loss $\mathcal{L}^y$ defined on target level (*i.e.* using the semantic prediction of both networks). The representation loss $\mathcal{L}^r$ enforces a similarity between the predicted representation $\hat{\mathbf{r}}^S_{t+1}$ of the student network and the representation $\mathbf{r}^T_{t+1}$ of the teacher network for the last N time steps:

$$\mathcal{L}^r(\theta^S_{enc}, \theta^S_{rec}) = \sum_{t=1}^{N} \lambda_t \tfrac{1}{2} \left\| f_{rec}(f_{enc}(\mathbf{x}_t; \theta^S_{enc}), \mathbf{m}_{t-1}; \theta^S_{rec}) - f_{enc}(\mathbf{x}_{t+1}; \theta^T_{enc}) \right\|_2^2, \quad (3.10)$$

where λ_t denotes a time-dependent weighting factor. This loss only optimizes the weights of the encoder network θ^S_{enc} as well as the weights of the recurrent module θ^S_{rec}. The weights of the decoder θ^S_{dec} are fixed to the corresponding weights of the teacher network. The representation loss is independent of the target y_t of the model and thus usable in a variety of applications. A disadvantage of this loss is its inability to adapt the weights of the decoder subnetwork to slight changes in the distribution of the abstract feature representation.

The loss $\mathcal{L}^y$ on the other hand enforces a similarity between the task-specific output of the two networks. Given our focus on a classification task (pixel-wise semantic segmentation), we adopt the knowledge distillation loss of Hinton *et al.* (2014). To increase clarity, we will omit the spatial pixel coordinates in the following equations (*i.e.* assume a classical classification task). The specified loss function is extended to the use case of semantic segmentation by aggregating the loss values over pixels. Let $\mathbf{e}^{T}_{t+1} = f'(\mathbf{x}_{t+1}; \theta^{T})$ and $\hat{\mathbf{e}}^{S}_{t+1} = f'_{pred}(\mathbf{x}_t, \mathbf{m}_{t-1}; \theta^{S}_{pred})$ be the pre-softmax activations (logits) of the teacher and student network, respectively. Using the softmax function, the logit e_i of class i can be transformed to a probability:

$$y_i^{\tau} = \frac{\exp(e_i/\tau)}{\sum_j \exp(e_j/\tau)}. \tag{3.11}$$

The temperature parameter τ scales the logit and is usually set to 1. A target y without superscript τ indicates the use of a temperature of 1. For the sake of brevity we will use the notation $\mathbf{y}^{\tau} = \mathrm{softmax}(\mathbf{e}/\tau)$ to indicate the full probability distribution over classes computed for the logits $\mathbf{e}$ using a temperature of τ. To compute the knowledge distillation objective function, Hinton *et al.* (2014) propose to use a cross-entropy loss in combination with soft targets (*i.e.* a softened probability distribution computed using a softmax temperature of greater than 1):

$$\mathcal{L}^{y}(\theta^{S}_{pred}) = \sum_{t=1}^{N} \lambda_t \mathcal{H}\left(\mathbf{y}^{T,\tau}_{t+1}, \hat{\mathbf{y}}^{S,\tau}_{t+1}\right), \tag{3.12}$$

where $\mathcal{H}$ denotes the cross-entropy and $\tau > 1$. During training the same temperature value τ is used to compute the probability distribution over classes of the teacher model:

$$\mathbf{y}^{T,\tau}_{t+1} = \mathrm{softmax}\left(\frac{\mathbf{e}^{T}_{t+1}}{\tau}\right) = \mathrm{softmax}\left(\frac{f'(\mathbf{x}_{t+1}; \theta^{T})}{\tau}\right) \tag{3.13}$$

as well as the student model:

$$\hat{\mathbf{y}}^{S,\tau}_{t+1} = \mathrm{softmax}\left(\frac{\hat{\mathbf{e}}^{S}_{t+1}}{\tau}\right) = \mathrm{softmax}\left(\frac{f'_{pred}(\mathbf{x}_t, \mathbf{m}_{t-1}; \theta^{S}_{pred})}{\tau}\right). \tag{3.14}$$

Using soft targets instead of hard targets leads to a richer transfer of information from the teacher network to the student network. A detailed discussion and experimental evaluation of the benefits of soft targets is given by Hinton *et al.* (2014).

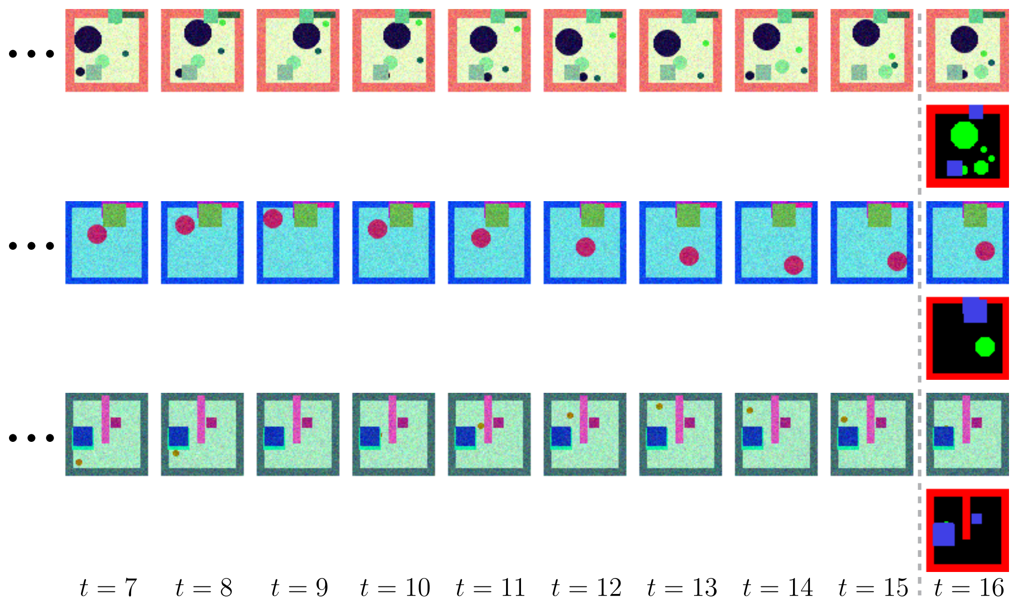


Figure 3.4: Simulated video dataset for semantic segmentation, emulating a 2D environment in which rectangles represent walls and borders, circles represent moving objects, and squares represent static foreground objects. Circles elastically collide with other circles, walls as well as borders and are occluded by squares. A semantic label is only available for the last image of each validation and test sequence as well as for 1000 training sequences. The four label classes are: ⬛ background pixels; 🟥 wall and border pixels; 🟩 pixels of circles; 🟦 pixels of squares. We only depict the last 10 frames of three example sequences of length 16.

3.4 Experiments

3.4.1 Dataset

To systematically evaluate different aspects of the proposed approach, we use a simulated video dataset for semantic segmentation. The data emulates a 2D environment of 64×64 pixels, in which rectangles represent walls and environment borders, circles represent moving objects, and squares represent static foreground objects. The circles elastically collide with other circles, walls, and borders. Squares occlude all other objects as well as each other. The color and size of the objects, the number of walls, squares, and circles as well as the velocity of circles are randomly sampled for each sequence. Additionally, we add independent *Gaussian* noise with zero mean and a variance of 0.005 to each pixel and clip the resulting pixel values to the interval $[0, 1]$ to obtain a valid sequence. The dataset thus mimics common phenomena of the real world (*e.g.* occlusion, interaction of objects, noisy observations, $\ldots$). In total, our dataset contains $10{,}000$ sequences of length 16, which are split into $6{,}000$ training sequences and $2{,}000$ validation and test sequences, respectively. A pixel-wise semantic label is available for the last image (*i.e.* frame 16)

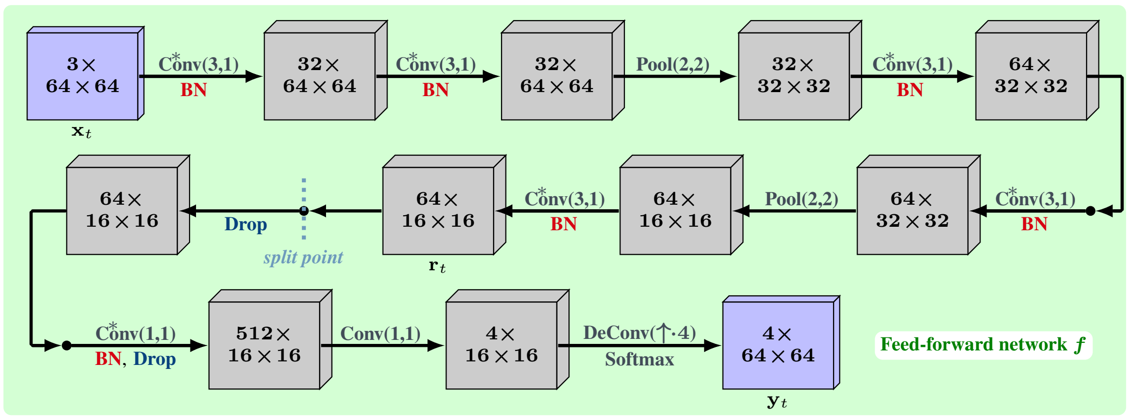


Figure 3.5: Convolutional feed-forward model f which receives an image $\mathbf{x}_t$ and infers a pixel-wise semantic segmentation $\mathbf{y}_t$. $\overset{*}{\textbf{Conv}}$**(f,s)**: convolutional layer with **f** filters and a stride of **s**. An additional asterix indicates the usage of a leaky ReLU nonlinearity with a negative slope coefficient of 0.01. **Pool(k,s)**: Max-pooling layer using a kernel size and stride of **k**. The usage of dropout and batch normalization is respectively indicated by **Drop** and **BN**. To obtain a semantic segmentation $\mathbf{y}_t$ with the same resolution as the input image $\mathbf{x}_t$, we use a deconvolutional layer **DeConv(↑·4)** which performs a fourfold upsampling, followed by a pixel-wise softmax function.

of each validation and test sequence. In addition, we assume that $1{,}000$ sequences of the training dataset are labeled in the same way. The pixels are labeled according to the four classes: background, walls and borders, circles, and squares. Figure 3.4 depicts example sequences of the dataset with corresponding ground-truth labels.

3.4.2 Implementation Details

To evaluate our proposed predictive transformation approach, we use the convolutional feed-forward model f depicted in Figure 3.5. This model computes a pixel-wise segmentation $\mathbf{y}_t$ given an image $\mathbf{x}_t$. Structurally the model is based on the *FCN-32s* architecture of Long *et al.* (2015) with a reduced number and size of layers. In contrast to the *FCN-32s* model, we use leaky ReLU nonlinearities (Maas *et al.*, 2013) and apply batch normalization (Ioffe and Szegedy, 2015). The model is trained using the $1{,}000$ labeled images of the training dataset and a pixel-wise cross-entropy loss. The parameters of the optimizer as well as the dropout probabilities are derived using an exhaustive random search. The feed-forward model f achieves a mean *Intersection over Union* (IoU) (Everingham *et al.*, 2015) of $89.02\,\%$ on the labeled images of the test dataset.

The predictive network is constructed in accordance to Section 3.3, by splitting the feed-forward model f before the first dropout layer. A vertical dotted line indicates the split

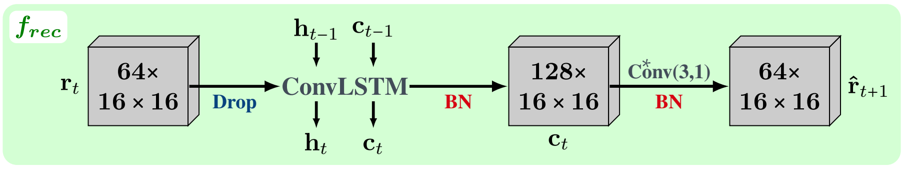


Figure 3.6: Recurrent predictive module f_{rec} consisting of a convolutional LSTM layer followed by a convolutional layer (see Section 3.6). $\overset{*}{\textbf{Conv}}$**(f,s)**: convolutional layer with **f** filters and a stride of **s**. The usage of dropout and batch normalization is respectively indicated by **Drop** and **BN**. The convolutional LSTM uses an input-to-hidden ($\mathbf{W}_{r*}$) filter size of 5×5 and a hidden-to-hidden ($\mathbf{W}_{h*}$) filter size of 7×7.

point in Figure 3.5. Following the description of Section 3.3.2, we construct a recurrent predictive module which propagates the feature representation $\mathbf{r}_t$ one time step into the future. The predictive module uses a convolutional LSTM with 128 features, an input-to-hidden ($\mathbf{W}_{r*}$) filter size of $5{\times}5$ and a hidden-to-hidden ($\mathbf{W}_{h*}$) filter size of $7{\times}7$. We additionally apply dropout to activations entering the LSTM and use zoneout (Krueger *et al.*, 2017) within the LSTM to improve generalization. The convolutional layer following the LSTM uses a filter size of $3{\times}3$ and a leaky ReLU with a negative slope coefficient of 0.01. Both layers are followed by a batch normalization layer and all convolutions operate with a stride of 1. We treat the initial hidden $\mathbf{h}_0$ and cell state $\mathbf{c}_0$ of the convolutional LSTM as model weights and learn them during training. The architecture and parametrization of the recurrent predictive module f_{rec} is depicted in Figure 3.6.

We train the predictive model four times using the different knowledge transfer variations of Section 3.3.3. Versions $\mathcal{PM}^{r,all}$ and $\mathcal{PM}^{y,all}$, respectively, use the representation loss $\mathcal{L}^r$ and target loss $\mathcal{L}^y$ and optimize all weights of the model. In the case of the representation loss, the weights of the decoder θ^S_{dec} are not updated. Versions $\mathcal{PM}^{r,rec}$ and $\mathcal{PM}^{y,rec}$ are trained by only optimizing the weights of the recurrent predictive module θ^S_{rec}. The weights of the encoder and decoder are initialized for all four settings with the corresponding weights of the feed-forward network (*i.e.* the teacher model). The training is conducted utilizing all 6,000 training sequences without the ground-truth labels. For each sequence of length 16, we provide the first 15 frames to the student and the teacher receives the last 15 frames. The weighting factor is set to $\lambda_t = ((t-1)/(N-1))^5$, with N equal to 15.

These models are compared with three baselines. Baseline $\mathcal{PM}^{copy}$ uses the weights of the feed-forward model and a copy function ($\hat{\mathbf{r}}_{t+1} = \mathbf{r}_t$) that serves as the predictive module. The second baseline $\mathcal{PM}^{warp}$ uses a post-processing procedure to temporally propagate the semantic segmentation $\mathbf{y}_t$ of the feed-forward model f:

$$\hat{\mathbf{y}}_{t+1} = f_{warp}(\mathbf{y}_t, \mathbf{x}_{t-1}, \mathbf{x}_t). \tag{3.15}$$

This baseline uses the segmentation $\mathbf{y}_t$ to separate pixels belonging to static objects (classes *squares*, *walls / borders*, and *background*) and dynamic objects (class *circles*). Pixels originating from dynamic objects (dynamic pixels) are forward warped (Szeliski, 2010) using the optical flow ($\mathbf{F}_{t\to t+1} \approx -\mathbf{F}_{t\to t-1}$) (Sánchez Pérez *et al.*, 2013) estimated from the last two observed input images ($\mathbf{x}_{t-1}$ and $\mathbf{x}_t$)[4]. The temporally predicted dynamic pixels are combined with the static pixels of $\mathbf{y}_t$ while using additional knowledge about the physics of the simulated environment: 1) pixels which are no longer occluded by a circle are set to the *background* class; 2) only background pixels are replaced by predicted dynamic pixels (*i.e.* temporally predicted pixels of class *circles* are not allowed to obscure pixels of class *squares* and class *walls / borders*). As no training is required, these two baselines do not need labeled data and only utilize the feed-forward model f analogous to our models. The third baseline $\mathcal{PM}^{sup}$ is a predictive model trained from scratch using the $1{,}000$ labeled training sequences. The architecture of this model is identical to the four models trained with our predictive transformation approach. Training of model $\mathcal{PM}^{sup}$ is performed in a supervised manner using the ground-truth labels as well as a pixel-wise cross-entropy loss. In this supervised setting, the predictive model receives the first 15 frames of each sequence and predicts a semantic segmentation of the 16th frame. The loss is computed using the predicted segmentation of frame 16 as well as the corresponding ground-truth label. Compared to the models trained with our knowledge transfer approach, this baseline requires labeled training data.

Additionally, we train a predictive model $\mathcal{PM}^{pre}$ which uses the weights of the best transformed model $\mathcal{PM}^{y,all}$ as an initialization and is then fine-tuned in accordance to $\mathcal{PM}^{sup}$. A similar two-step training approach, consisting of pre-training using soft targets and fine-tuning using hard targets, was used by Tang *et al.* (2016). To make the comparison of the different models as fair as possible, we perform an extensive random search for each model to determine the training, regularization, and loss parameters.

[4]The predicted pixel locations at time step $t+1$ most likely do not align with integer pixel coordinates. We assign in such cases the class of the predicted pixel (*i.e.* class *circles*) to all four nearest pixels.

Model	Mean	Background	Walls / Borders	Circles	Squares
$\mathcal{PM}^{y,all}$	$82.97\,\%$	$89.61\,\%$	$96.35\,\%$	**$59.37\,\%$**	$86.55\,\%$
$\mathcal{PM}^{r,all}$	$79.63\,\%$	$87.10\,\%$	$96.04\,\%$	$49.54\,\%$	$85.84\,\%$
$\mathcal{PM}^{y,rec}$	$80.05\,\%$	$87.58\,\%$	$96.01\,\%$	$51.52\,\%$	$85.09\,\%$
$\mathcal{PM}^{r,rec}$	$80.59\,\%$	$87.80\,\%$	$96.02\,\%$	$53.21\,\%$	$85.33\,\%$
$\mathcal{PM}^{copy}$	$75.72\,\%$	$83.73\,\%$	$95.56\,\%$	$41.14\,\%$	$82.45\,\%$
$\mathcal{PM}^{warp}$	$77.36\,\%$	$84.19\,\%$	$95.56\,\%$	$47.25\,\%$	$82.45\,\%$
$\mathcal{PM}^{sup}$	$80.63\,\%$	$88.23\,\%$	$97.57\,\%$	$43.36\,\%$	$93.35\,\%$
$\mathcal{PM}^{pre}$	**$84.96\,\%$**	**$91.06\,\%$**	**$97.61\,\%$**	$57.81\,\%$	**$93.37\,\%$**

Table 3.1: Mean IoU score ($\%$) and per-class IoU scores ($\%$) on the test dataset for all introduced models. The first six models are trained using only the feed-forward model and do not rely on additional training data. The last two models additionally rely on labeled training data.

3.4.3 Results

To evaluate our approach, Table 3.1 reports the per-class IoU scores and the mean IoU score of all introduced models on the 2000 sequences of the test dataset. The metrics are calculated using the labeled 16th frame of each sequence as well as the corresponding semantic prediction of the models. Equivalent to training, the prediction of a model is conditioned on the first 15 frames of a sequence or a subset of these frames. Model $\mathcal{PM}^{copy}$ only uses the information of frame 15 to create a semantic prediction of frame 16. Respectively, model $\mathcal{PM}^{warp}$ only utilizes the information of frame 14 and 15. All other models receive the first 15 frames of a sequence as input. In addition to the quantitative results in Table 3.1, we visualize in Figure 3.7 semantic predictions of the best predictive model $\mathcal{PM}^{y,all}$ trained with our knowledge transfer approach without using labeled data.

All models trained with our predictive knowledge transfer approach significantly outperform the two baselines $\mathcal{PM}^{copy}$ and $\mathcal{PM}^{warp}$ which also do not rely on labeled training data. Our best model $\mathcal{PM}^{y,all}$ improves the mean IoU score by more than $7\,\%$ compared to the copy baseline and by more than $5.5\,\%$ compared to baseline $\mathcal{PM}^{warp}$. The difference is especially visible when focusing on the class *circles* which represents moving objects. The copy baseline does not learn a recurrent predictive module and thus generates a rather low IoU score of $41.14\,\%$ for the class *circles*. Due to its ability to model motion using the optical flow, baseline $\mathcal{PM}^{warp}$ is able to increase the IoU score of class *circles* to $47.25\,\%$. Our best model $\mathcal{PM}^{y,all}$ achieves an IoU score of $59.37\,\%$ for class *circles*, indicating that the recurrent predictive module has learned meaningful dynamics and interactions of the different object types. The predictive capabilities of this model are additionally visible in Figure 3.7. Our model is able to recognize different object types and to predict the future pixel-wise semantic labels. It has learned the dynamics and interactions of circles and can predict occlusion by squares (Figure 3.7, prediction $\hat{\mathbf{y}}_{t-1}$) as well as elastic collisions with walls (Figure 3.7, prediction $\hat{\mathbf{y}}_{t+1}$). It can even resolve heavy occlusion of one of the circles in image $\mathbf{x}_t$ and predict its reappearance beneath the occluding square (Figure 3.7, prediction $\hat{\mathbf{y}}_t$). The predictions of $\mathcal{PM}^{y,all}$ are rather coarse which is due to the structure of the chosen feed-forward model f.

There are four options to train a predictive model in a self-supervised manner using our knowledge transfer approach — two loss variants and two optimization settings (see Section 3.3.3). When using the representation loss $\mathcal{L}^r$, it is beneficial to only train weights of the recurrent predictive module. However, the difference is relatively small — the mean IoU score of model $\mathcal{PM}^{r,rec}$ is $80.59\,\%$ and of model $\mathcal{PM}^{r,all}$ $79.63\,\%$. For the target loss $\mathcal{L}^y$, the best results are obtained when all weights are optimized. Compared to the representation loss, the performance differences of the two optimization settings are larger. Model $\mathcal{PM}^{y,all}$ achieves a mean IoU score of $82.97\,\%$ while the score of model $\mathcal{PM}^{y,rec}$ is $80.05\,\%$. In general, the best results are achieved by using the target loss and optimizing all weights. This is most likely due to the fact that in this setting the predictive model can be trained in an end-to-end manner using a loss tailored towards the target application (*i.e.* semantic segmentation). The weights of the decoder subnetwork can thus be adapted to changes in the distribution of the abstract feature representation and the encoder can be adapted to create an abstract representation which is best suited for temporal prediction.

Baseline $\mathcal{PM}^{sup}$, which is trained in a supervised manner using labeled data, achieves a mean IoU score of $80.63\,\%$. Our best model $\mathcal{PM}^{y,all}$ thus also outperforms this supervised baseline by more than $2\,\%$ according to the mean IoU score. This is most likely due to the benefits of training with softened targets compared to hard labels as well as due to the comparatively large number of available unlabeled sequences. Interestingly, model $\mathcal{PM}^{sup}$ outperforms model $\mathcal{PM}^{y,all}$ for class *wall / borders* and class *squares*, which represent static objects. The IoU score of class *circles*, on the other hand, is significantly worse than the corresponding score of model $\mathcal{PM}^{y,all}$. Model $\mathcal{PM}^{sup}$ achieves an IoU score of $43.36\,\%$ on class *circles* which is only $2.22\,\%$ larger compared to the score of the copy baseline $\mathcal{PM}^{copy}$. The IoU score on class *wall / borders* and *squares* are respectively $2.01\,\%$ and $10.76\,\%$ larger than the corresponding scores of the

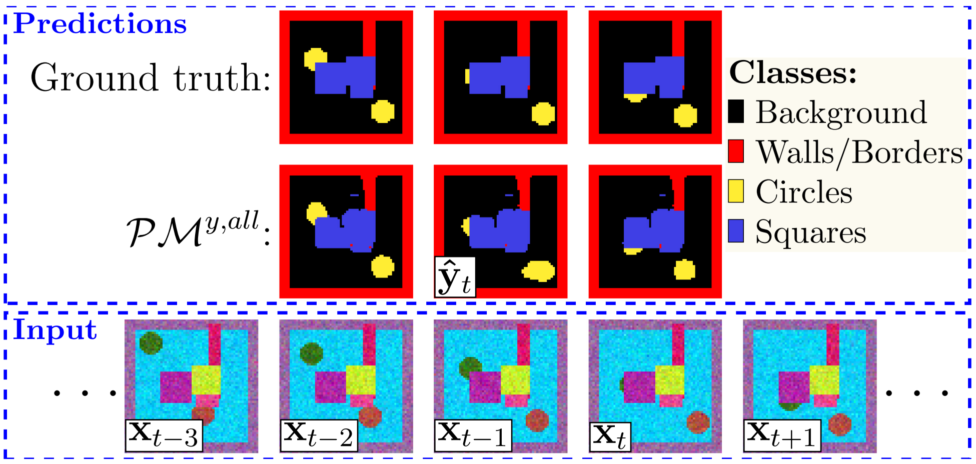


Figure 3.7: Example predictions of model $\mathcal{PM}^{y,all}$. For three time steps ($\mathbf{x}_{t-2}$, $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$) we visualize the semantic prediction of the model for the next time step ($\hat{\mathbf{y}}_{t-1}$, $\hat{\mathbf{y}}_t$ and $\hat{\mathbf{y}}_{t+1}$) as well as the corresponding ground truth semantic map of the next time step. The predictions are each conditioned on the previous 15 frames. The predictive model $\mathcal{PM}^{y,all}$ is able to infer the different object types as well as their dynamics and interactions.

feed-forward model f on the test data[5]. The per-class IoU scores of the supervised model indicate that it mainly uses information from previous time steps to provide more accurate predictions of static objects. Its ability to create precise semantic predictions of the interacting dynamic objects is however limited compared to model $\mathcal{PM}^{y,all}$.

The overall best results can be achieved when the model $\mathcal{PM}^{y,all}$ is further fine-tuned in a supervised manner using the labeled training data. The corresponding model $\mathcal{PM}^{pre}$ achieves a mean IoU score of $84.96\,\%$ on the test dataset. Fine-tuning using the ground-truth labels yields an increased performance for classes of static objects.

From a labeling point of view, the same amount of effort is required to create model $\mathcal{PM}^{sup}$ and $\mathcal{PM}^{pre}$. The same 1000 labeled images are used in our experiments to train each of the two models[6]. Model $\mathcal{PM}^{pre}$, which is significantly better than model $\mathcal{PM}^{sup}$, requires additional unlabeled sequences for training, which are usually easy and inexpensive to acquire. Besides the use case of transforming a feed-forward model into a predictive model using only unlabeled sequence data, our approach is thus also applicable in a semi-supervised setting (Ouali *et al.*, 2020) — *i.e.* a scenario in which we want to train a model which predicts the semantic segmentation of the next frame conditioned

[5]We use the IoU score of the feed-forward model f which creates a semantic prediction for time step 16 conditioned on the image of time step 16 as a reference.

[6]The 1000 labeled frames which are used to train model $\mathcal{PM}^{sup}$ are also used to train the feed-forward model. Using unlabeled sequence data, the feed-forward model is transformed to the predictive model $\mathcal{PM}^{y,all}$ and fine-tuned using the sequences containing the 1000 labeled frames to create $\mathcal{PM}^{pre}$.

on previous observations using a dataset which contains a comparatively small amount of labeled examples and a large amount of unlabeled examples. The semi-supervised learning version of our approach includes three training steps, consisting of: 1) training of a feed-forward model using the labeled frames; 2) predictive transformation in accordance to Section 3.3 using a large amount of unlabeled sequences; 3) fine-tuning of the predictive model using the sequences with labeled frames. With this approach, one can use a large amount of unlabeled data that is inexpensive to acquire to learn powerful predictive models.

3.5 Conclusion

To enable acting in an anticipatory way, autonomous agents have to reason about the future environment state. Current deep learning-based approaches which generate a semantic representation of the observed environment (*i.e.* a pixel-wise semantic segmentation) mainly operate in a non-predictive fashion. In this chapter, we proposed an approach to transform such non-predictive feed-forward networks into ones which predict the future. Our proposed predictive transformation extends a given feed-forward network with a recurrent predictive module, while reusing its original structure and encoded task-specific knowledge. To optimize the weights of the resulting predictive model, we proposed a teacher-student-like training strategy and evaluated four corresponding training losses. Training can thus be conducted without the need of labeled video data in a fully self-supervised fashion. Our qualitative and quantitative analysis on simulated motion sequences showed that the resulting predictive network can model the dynamics of the scene (*e.g.* the interaction and motion of objects), enabling meaningful predictions of the future pixel-wise semantic segmentation. We additionally demonstrated the advantages of using our approach when training a predictive model on a limited amount of labeled data without the availability of a corresponding feed-forward model (*i.e.* a semi-supervised setting). Although the main focus of this work has been on generating models that predict the pixel-wise semantic segmentation of the next time step conditioned on previous observed image frames, our general approach is applicable to a variety of feed-forward networks.

Concurrent to our work (Wagner *et al.*, 2017), Luc *et al.* (2017) and Jin *et al.* (2017b) investigated similar approaches to predict the future semantic segmentation. Unlike our model, Luc *et al.* (2017) do not use recurrent structures to model the temporal dependencies, but adopt the multi-scale architecture of Mathieu *et al.* (2016). Similar to our approach, they use a feed-forward model to create pseudo-labels for a large set of unlabeled videos. However, they do not reuse the structure and trained weights of the feed-forward model when constructing the predictive network. Their training is also inspired by network distillation, but differs in the choice of loss function. In accordance with Mathieu *et al.* (2016), they optimize a combination of an *L1* loss and a gradient difference loss. In addition, they also evaluate adversarial fine-tuning, which did not consistently lead to a significant performance improvement. Their best performing model predicts future segmentations based on previous segmentations, unlike our model which receives raw images of previous time steps as input.

Jin *et al.* (2017b) propose to temporally stabilize the pixel-wise semantic segmentation of the current image using information of previous time steps. Their proposed model (*PEARL*) is trained in two phases using additional auxiliary tasks. In phase one, the portion of the model which processes the images of the previous time steps is pre-trained. The pre-training is conducted in an unsupervised manner by performing future video frame prediction. The overall *PEARL* architecture is trained in phase two using a multi-task loss and labeled data. Their multi-task learning approach employs the concept of hard parameter sharing (Ruder, 2017) for the two tasks: future semantic segmentation and temporally stabilized semantic segmentation. Compared to our work as well as the work of Luc *et al.* (2017), future semantic forecasting is used by Jin *et al.* (2017b) only as an auxiliary training task and is not further evaluated in their experiments. The training of the *PEARL* model additionally requires labeled data.

Following the publication of the contents of this chapter, a variety of work developed new methods for predicting raw sensory data or future object trajectories. For detailed discussions of such methods, please refer to the review papers of Oprea *et al.* (2020) and Rudenko *et al.* (2020).

Several publications (Jin *et al.*, 2017a; Nabavi *et al.*, 2018; Luc *et al.*, 2018; Couprie *et al.*, 2019; Terwilliger *et al.*, 2019; Chiu *et al.*, 2020; Saric *et al.*, 2020; Hu *et al.*, 2020) also focused on developing new models for future semantic segmentation. Nabavi *et al.* (2018) forecast the future segmentation conditioned on the segmentation of four previous time steps. To capture spatio-temporal dependencies, they use convolutional LSTM cells on four abstraction levels. Pseudo training labels are generated using a standard semantic segmentation model. Terwilliger *et al.* (2019) use a warp layer to explicitly transform the segmentation of the last observed time step into the future. The transformation is parametrized by a model which predicts the future optical flow. By decomposing the task into future flow forecasting and current frame segmentation, they manage to design a model with a comparatively small number of parameters and a low inference time. Saric *et al.* (2020) propose a novel predictive module which forecasts the representation of a semantic segmentation model into the future. Their predictive module performs two independent forecasts of the feature representation and combines them using predicted pixel-wise weights. One of the forecasts is conducted by directly regressing future features from observed previous features, the other forecast uses an explicit transformation similar to Terwilliger *et al.* (2019). Our model as well as most of the referenced semantic forecasting approaches are deterministic. Thus, they do not model the stochasticity and multimodal nature of the world, which is especially relevant for complex scenes and long time horizons. Hu *et al.* (2020) adopt a conditional variational approach to design a model which predicts the future in a probabilistic manner. Conditioned on observed video data, their model jointly predicts future semantic segmentation, depth, and optical flow. They quantitatively highlight the ability of their model to perform plausible and diverse future predictions.

Building on the experience we have gained with our predictive transformation approach, we focus in Chapter 4 and Chapter 5 on using temporal information to reduce aleatoric failures of single-frame semantic segmentation networks.

4 Hierarchical Recurrent Filtering for Fully Convolutional DenseNets

Autonomous agents require the ability to generate a robust and reliable representation of the environment. Deep learning-based methods have greatly improved perception systems that operate on image data, but still fail in challenging situations. These failures are often not solvable on the basis of a single image. In this work, we present a parameter efficient temporal filtering concept which extends an existing single-frame semantic segmentation model to work with multiple video frames. The resulting recurrent architecture temporally filters feature representations on all abstraction levels in a hierarchical manner, while conceptually decoupling temporal dependencies from scene representation.

Using a synthetic video dataset, we quantitatively compare our model with other temporal architectures and qualitatively show its ability to suppress noise, derive additional object properties, and resolve missing information as well as ambiguities. Our best model significantly outperforms common non-recurrent temporal baseline models and achieves a $3.8\,\%$ better mean IoU score compared to a similar non-hierarchical, recurrent baseline model. A comparison to single-frame segmentation models further illustrates the clear advantage of temporal architectures with respect to robustness.

4.1 Introduction

A robust and reliable perception and interpretation of the environment is a crucial capability of autonomous vehicles, such as self-driving cars or mobile robots. Deep learning-based methods greatly advanced the generation of robust environment representations and dominate the majority of perception benchmarks. From a safety point of view, a major drawback of popular benchmark datasets, like *Cityscapes* (Cordts *et al.*, 2016) or *Caltech* (Dollár *et al.*, 2012), is their recording at daytime under good or normal environment conditions. In order to deploy autonomous vehicles in an unconstrained world without any human supervision, one has to make sure that they still work in challenging situations such as sensor outages or adverse weather. These situations induce failures of vision-based perception systems, which are not solvable by just using the information provided by a single image. In this chapter, we investigate the utilization of temporal information to improve the robustness of such systems.

In the scope of this work, we divide failures of the perception system according to the classification of uncertainties (Kiureghian and Ditlevsen, 2009; Kendall and Gal, 2017) into two categories: epistemic failures and aleatoric failures. We denote a failure as epistemic, if it can be solved by gathering more training data or by using a more powerful model (Kiureghian and Ditlevsen, 2009). Aleatoric failures, on the other hand, originate from perturbations inherent in the data (Kendall and Gal, 2017). More data or a better tuned model will not solve these failures. To tackle aleatoric failures, one has to enhance the information provided to the perception system. This can be achieved by using a better sensor, fusing information of multiple sensors, and / or by taking temporal information into account.

The aleatoric failures can be further subdivided into failures due to noise, ambiguities, missing information, or occlusions.

Noise Noise is an unwanted disturbance of a desired signal, often caused by harsh environment conditions, like rain or snow, or by the sensor itself (*e.g.* electronic noise). Harsh weather conditions, for example, drastically reduce the information content provided by an image and are challenging for computer vision algorithms and humans. In general, noise effects the whole image.

Ambiguities Sensors only observe a specific physical property of the world, *e.g.* the electromagnetic spectrum of a specific frequency band. The observed property may not be suitable to distinguish objects of different classes. For example, to reliably distinguish a parked car from a moving one, the information from a single image is generally insufficient. Such a classification would benefit from a sequence of video frames or information from a sensor which can directly measure the velocity of an object.

Missing Information A malfunction of the sensor hardware or limitations introduced by the measurement principle of a sensor can result in missing information. Sudden illumination changes, shadows, or bad lighting conditions, for example, drive cameras into their limitations. These perturbations can affect the whole image or only a subregion of the image and are typically highly correlated in space and / or time.

Occlusions Occlusions occur when one object of interest is fully or partially hidden behind another object of interest. This is a common failure case especially for autonomous vehicles which operate in crowded areas.

In this chapter, we focus on using temporal information to reduce aleatoric failures of a single-frame semantic segmentation model. Using information from previous time steps is quite cost-effective compared to other approaches (*e.g.* fusing the information of multiple sensors; see Section 2), since no additional sensor hardware is required. We build upon the *Fully Convolutional DenseNet* (FC-DenseNet) of Jégou *et al.* (2017) and propose a temporal filtering concept, which extends it to multiple video frames. The temporal integration is achieved by recurrently filtering the representations of the FC-DenseNet on all abstraction levels in a hierarchical manner. The resulting *Recurrent Fully Convolutional DenseNet* (RFC-DenseNet) conceptually decouples temporal dependencies from scene representation. This property makes it easy to transform any single-image FC-DenseNet into a corresponding multi-image RFC-DenseNet. The decoupling additionally increases the transparency of the model, enabling an easy exchange of modules responsible for temporal filtering (*Filter Module*s) or hierarchical feature generation. Due to the hierarchical nature of the filter concept, our model can utilize temporal correlations on all levels of abstraction. In comparison, many approaches in the literature only filter the representation of a high abstraction level (Fayyaz *et al.*, 2016; Valipour *et al.*, 2017). These approaches suffer in situations when the information required to resolve aleatoric failures is not propagated to high-level feature maps. Despite its recurrent nature, the proposed architecture is parameter efficient.

Using a simulated video dataset, we quantitatively and qualitatively evaluate our proposed RFC-DenseNet architecture and compare its performance with other non-temporal and temporal segmentation models. The use of simulated data enables a focus on aleatoric perturbations which are often underrepresented in current real-world benchmark datasets. In total, we propose and evaluate three different versions of the RFC-DenseNet architecture differing in the choice of the used *Filter Module*. Our experiments show a favorable performance for a recurrent *Filter Module* consisting of an encoder-decoder structure as well as an enclosing identity skip connection. By comparing our RFC-DenseNet model as well as other temporal architectures with single-frame models, we illustrate advantages in terms of robustness that can be achieved by using information from previous time steps. We additionally evaluate the benefits of recurrent and hierarchical filtering by comparing the performance of our model with two non-recurrent temporal architectures as well as a non-hierarchical, recurrent model. Our best RFC-DenseNet outperforms all other evaluated temporal semantic segmentation models, achieving a mean IoU score which is at least $3.8\,\%$ better compared to the other models. Using exemplary predictions, we show the ability of RFC-DenseNet to suppress noise, derive additional object properties, and resolve missing information as well as ambiguities.

The *Recurrent Fully Convolutional DenseNet* (RFC-DenseNet) presented in this chapter was first published at the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (Wagner *et al.*, 2018b).

4.2 Related Work

The majority of previous deep learning-based research used a single-image observation of the environment and focused on the reduction of epistemic failures. Starting from fairly simple architectures like *Regions with CNN features* (R-CNN) (Girshick *et al.*, 2014) for object detection or the *Fully Convolutional Network* (FCN) (Long *et al.*, 2015) for semantic segmentation, the performance of models consistently increased by introducing new layer structures (He *et al.*, 2016a; Huang *et al.*, 2017) or by task-specifically adjusting the general network architecture (Liu *et al.*, 2016b; Jégou *et al.*, 2017). Complementary research studied techniques to enhance the available training data using data augmentation (Krizhevsky *et al.*, 2012; Liu *et al.*, 2016b), applying transfer learning (Yosinski *et al.*, 2014; Ganin and Lempitsky, 2015; Wagner *et al.*, 2016), utilizing simulated data (Richter *et al.*, 2016), or by using additional unlabeled data (Rasmus *et al.*, 2015; Dosovitskiy *et al.*, 2016; Wagner *et al.*, 2017). In addition, new large benchmark datasets (Cordts *et al.*, 2016) have been introduced containing a high variability of scenes and high-quality labels.

Fewer deep learning-based research focuses on the reduction of aleatoric failures of vision systems. A popular approach to address such failures is to fuse data from various sensors. Mees *et al.* (2016) showed the advantage of fusing the modalities of a *RGB-D* camera by using a mixture of *Convolutional Neural Network* (CNN) experts. Hazirbas *et al.* (2016) proposed a CNN for semantic segmentation that fuses depth information with *RGB* data. Their encoder-decoder architecture consists of two encoder branches, one per modality. The features of the depth encoder branch are fused into the *RGB* encoder branch on multiple abstraction levels. In Section 2, we investigated the benefits of multispectral pedestrian detectors which use the information of a visible and a thermal camera. Especially at night, the complementary thermal features proved to be beneficial, when the visible camera suffers from low illumination conditions. A more detailed discussion of different multispectral pedestrian detectors is given in Section 2.2. Complementary approaches focus on exploiting temporal information to reduce aleatoric failures. In the remainder of this section, we provide an overview of such methods, focusing on semantic segmentation models.

A common technique for improving pixel-wise semantic segmentation models using temporal information is to use a non-hierarchical, global filtering approach. Fayyaz *et al.* (2016) and Valipour *et al.* (2017) generate an abstract feature representation for each image in a video sequence and use *Recurrent Neural Network*s (RNNs) to temporally filter them. The resulting filtered representation is post-processed and up-sampled to generate a pixel-wise semantic segmentation. Jin *et al.* (2017b) utilize a sequence of previous images to predict an abstract feature representation of the current image. The predicted representation is fused with the current one and propagated through a decoder network. Similar approaches exist, which apply post-processing steps on top of individual frame segmentations. Ghazvinian Zanjani and Gerven (2016) use a *Conditional Random Field* (CRF) to integrate short-term temporal information with structural scene properties. Lei and Todorovic (2016) propose the Recurrent-Temporal Deep Field architecture, which combines *Fully Convolutional Networks* (FCNs), a recurrent temporal restricted *Boltzmann* machine and a CRF into one framework. Kundu *et al.* (2016) use a densely connected CRF operating on a Euclidean feature space, optimized to minimize the distance between features associated with corresponding points in the scene. Our approach differs from these approaches due to its hierarchical nature.

Other approaches (Tran *et al.*, 2016; Zhang *et al.*, 2014) build semantic video segmentation networks using spatio-temporal features computed with 3D-Convolutions (Ji *et al.*, 2013; Tran *et al.*, 2015). We differ from these approaches due to the explicit utilization of filters, which are independently applied to feature representations of all abstraction levels. Gadde *et al.* (2017) propose to improve segmentation performance by warping one or multiple feature representations of the previous frame to the current frame using a dense optical flow field. The warped representations are fused with the corresponding representations of the current time step to increase stability and temporal consistency. The method of Gadde *et al.* (2017) relies on the availability of dense optical flow, which often cannot be calculated, especially in the case of heavy noise or missing information. The *Recurrent Convolutional Neural Network* of Pavel *et al.* (2015), which builds upon prior work of Behnke (2003), is similar to our architecture. This method uses layer-wise recurrent self-connections as well as top-down connections to stabilize representations. The approach focuses on a fully recurrent topology, while our approach conceptually decouples temporal filtering and scene representation. Additionally, our approach uses a dense connection pattern to get an improved signal flow.

Related fields of research are semantic forecasting and efficient semantic video segmentation. The former focuses on predicting the pixel-wise semantic segmentation of future image frames conditioned on previous frames or segmentations. In Chapter 3, we propose an approach to build semantic forecasting models based on the transformation of non-predictive feed-forward networks. The proposed predictive transformation can be conducted without the need of labeled video data in a fully self-supervised fashion. For a detailed discussion of other semantic forecasting models, we refer the reader to Section 3.2 and 3.5. The research field of efficient semantic video segmentation (Shelhamer *et al.*, 2016; Zhu *et al.*, 2017) deals with the development of temporal models which reuse features of previous time steps to reduce the computational cost and / or latency of frame segmentation. The goal is thus not to use information of previous time steps to improve the performance of models, but rather to avoid or delay computation by reusing features. Due to the different focus of these research fields, the developed methods and models are not directly comparable to our proposed *Recurrent Fully Convolutional DenseNet* (RFC-DenseNet) architecture.

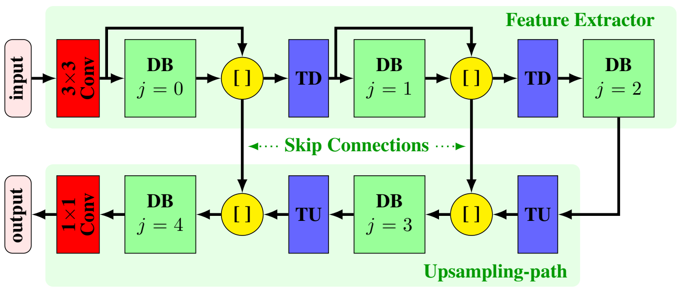
4.3 Recurrent Fully Convolutional DenseNet

In this section, we present an approach to extend the *Fully Convolutional DenseNet* (FC-DenseNet) of Jégou *et al.* (2017) to video data, by hierarchically filtering the representations of the model. First, we review the FC-DenseNet for semantic segmentation in Section 4.3.1. Second, we present our concept to temporally filter the representations of the model in a hierarchical fashion in Section 4.3.2. Finally, Section 4.3.3 elaborates three specific instances of the resulting recurrent segmentation architecture.

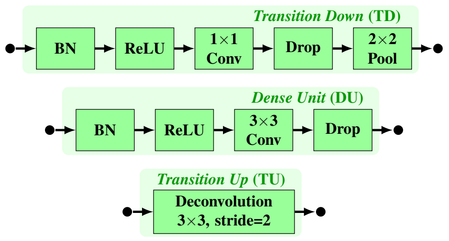
4.3.1 Revisiting the Fully Convolutional DenseNet (FC-DenseNet)

The FC-DenseNet builds on the *Fully Convolutional Network* (FCN) for semantic segmentation defined by Long *et al.* (2015) and extends it using concepts of the *Densely Connected Convolutional Network* (DenseNet) introduced by Huang *et al.* (2017).

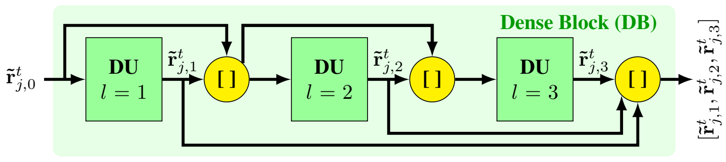
Fully Convolutional Networks (FCNs) (Long *et al.*, 2015) produce dense predictions of arbitrary-sized images by replacing the fully connected layers of classification networks, like *VGG* (Simonyan and Zisserman, 2015) or *GoogLeNet* (Szegedy *et al.*, 2015), with corresponding convolutional ones. The resulting coarse classification maps are upsampled to pixel-dense outputs using deconvolutional layers. The weights of the deconvolutional layers can either be learned or fixed to bilinear upsampling. To get a more refined pixel-wise segmentation the authors additionally propose to use skip layers, which enhance the coarse information of the final prediction layer with fine information from lower layers. In total, the *Fully Convolutional Network* (FCN) for semantic segmentation (Long *et al.*, 2015) consists of three components, a fully convolutional feature extractor, an upsampling-path, and interlinking skip layers.



(a) *Fully Convolutional DenseNet* (FC-DenseNet) of depth two ($n_{\text{pool}} = 2$), consisting of a DenseNet-based feature extractor, a DB enhanced upsampling-path, and interlinking skip connections. We apply a *softmax* function to the output of the last convolutional layer to compute pixel-wise class scores. The first convolutional layer computes n_{conv} features.



(b) Units of the FC-DenseNet.


Dense Block (DB)
$\tilde{\mathbf{r}}^t_{j,0}$
DU $l = 1$
$\tilde{\mathbf{r}}^t_{j,1}$
[ ]
DU $l = 2$
$\tilde{\mathbf{r}}^t_{j,2}$
[ ]
DU $l = 3$
$\tilde{\mathbf{r}}^t_{j,3}$
[ ]
$[\tilde{\mathbf{r}}^t_{j,1}, \tilde{\mathbf{r}}^t_{j,2}, \tilde{\mathbf{r}}^t_{j,3}]$

(c) *Dense Block* (DB) of length three ($n_{\text{layer}} = 3$). Each DU computes n_{feature} new features using all previous features of matching spatial feature-map size.

Figure 4.1: Structural layout of the *Fully Convolutional DenseNet* (FC-DenseNet) as well as its subcomponents: *Dense Block* (DB), *Transition Down* (TD), *Transition Up* (TU), and *Dense Unit* (DU). We use **[ ]** to indicate the concatenation of features. Each DU computes n_{feature} new features. The TU and TD units only change the spatial size of a feature representation while keeping the number of features constant.

Following the basic architectural structure of the FCN, FC-DenseNet is constructed by using a fully convolutional version of DenseNet as the feature extractor, utilizing a *Dense Block* (DB) enhanced upsampling-path, and interlinking both paths using skip connections. The architecture of FC-DenseNet is visualized in Figure 4.1a. The DenseNet, used in the feature extractor, is a convolutional neural network which iteratively adds features to a stack, which we refer to as the global feature state. Newly added features $\tilde{\mathbf{r}}^t_{j,l}$ are computed using all previous ones $(\tilde{\mathbf{r}}^t_{j,l-1}, \dots, \tilde{\mathbf{r}}^t_{j,0})$ of matching spatial feature-map size (see Figure 4.1c):

$$\tilde{\mathbf{r}}^t_{j,l} = f^{DU}_{j,l}([\tilde{\mathbf{r}}^t_{j,l-1}, \dots, \tilde{\mathbf{r}}^t_{j,0}]; \theta^{DU}_{j,l}); \tag{4.1}$$

where $f^{DU}_{j,l}$ is the function of the corresponding *Dense Unit* (DU), with parameters $\theta^{DU}_{j,l}$. We use $[\tilde{\mathbf{r}}^t_{j,l-1}, \dots, \tilde{\mathbf{r}}^t_{j,0}]$ to specify the concatenation of features retrieved from the global feature state. The global feature state is initialized with a set of features computed using one preceding convolutional layer. To ensure a slow growth of the global feature state, the number n_{feature} of newly added features is limited to a small value, *e.g.* $n_{\text{feature}} = 12$.

The dense connection pattern between units benefits the information and gradient flow and facilitates an extensive reuse of already computed features (Huang *et al.*, 2017). Due to the feature reuse introduced by the global feature state, DenseNet is very parameter efficient. To successively reduce the spatial size of feature maps, sequences of densely connected layers, also called *Dense Blocks* (DBs) (see Figure 4.1c), are interleaved by *Transition Down* (TD) units. These units downsample the global feature state while keeping the number of features constant. The persistence of the global state in the feature extractor is ensured by concatenating the output of each but the last *Dense Block* with the corresponding input. The high-level spatial feature maps computed in the feature extractor's last *Dense Block* are incrementally up-sampled using *Transition Up* (TU) units, while enriching the features with finer-grained information via skip connections. The features of *Transition Up* units are concatenated with the features provided by corresponding skip connections and are processed using *Dense Blocks*. The full persistence of the global feature state is discontinued in the upsampling-path to limit the growth of feature maps. All features computed in the last *Dense Block* of the model are processed by a 1×1 convolutional layer followed by pixel-wise *softmax* function to compute pixel-wise class scores. The implementation of the individual units is specified in Figure 4.1b. Experiments of Jégou *et al.* (2017) on the *CamVid* (Brostow *et al.*, 2008) and *Gatech* (Raza *et al.*, 2013) dataset show the ability of FC-DenseNet to outperform comparable segmentation models (Long *et al.*, 2015; Kendall *et al.*, 2017; Kundu *et al.*, 2016; Yu and Koltun, 2016) while being highly parameter efficient.

4.3.2 Temporal Representation Filtering

Due to perturbations inherent in the data (*e.g.* noise introduced by adverse weather conditions; see Section 4.1), the features $\tilde{\mathbf{r}}^t_{j,l}$ computed in each DU of the FC-DenseNet are only a crude approximation of the true feature representation without perturbations. To get an improved estimate of the features, we propose to temporally filter them using a recurrent *Filter Module* (FM):

$$\hat{\mathbf{r}}^t_{j,l} = f^{FM}_{j,l}(f^{DU}_{j,l}([\hat{\mathbf{r}}^t_{j,l-1}, \ldots, \hat{\mathbf{r}}^t_{j,0}]; \theta^{DU}_{j,l}), \mathbf{m}^{t-1}_{j,l}; \theta^{FM}_{j,l}); \tag{4.2}$$

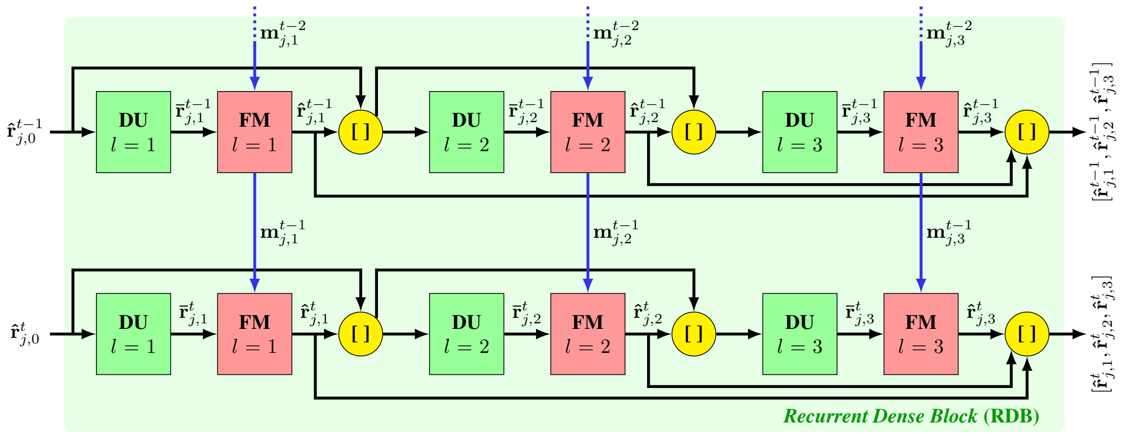

$\mathbf{m}^{t-2}_{j,1}$
$\mathbf{m}^{t-2}_{j,2}$
$\mathbf{m}^{t-2}_{j,3}$
$\hat{\mathbf{r}}^{t-1}_{j,0}$
DU $l = 1$
$\bar{\mathbf{r}}^{t-1}_{j,1}$
FM $l = 1$
$\hat{\mathbf{r}}^{t-1}_{j,1}$
[ ]
DU $l = 2$
$\bar{\mathbf{r}}^{t-1}_{j,2}$
FM $l = 2$
$\hat{\mathbf{r}}^{t-1}_{j,2}$
DU $l = 3$
$\bar{\mathbf{r}}^{t-1}_{j,3}$
FM $l = 3$
$\hat{\mathbf{r}}^{t-1}_{j,3}$
$[\hat{\mathbf{r}}^{t-1}_{j,1}, \hat{\mathbf{r}}^{t-1}_{j,2}, \hat{\mathbf{r}}^{t-1}_{j,3}]$
$\mathbf{m}^{t-1}_{j,1}$
$\mathbf{m}^{t-1}_{j,2}$
$\mathbf{m}^{t-1}_{j,3}$
$\hat{\mathbf{r}}^{t}_{j,0}$
$\bar{\mathbf{r}}^{t}_{j,1}$
$\hat{\mathbf{r}}^{t}_{j,1}$
$\bar{\mathbf{r}}^{t}_{j,2}$
$\hat{\mathbf{r}}^{t}_{j,2}$
$\bar{\mathbf{r}}^{t}_{j,3}$
$\hat{\mathbf{r}}^{t}_{j,3}$
$[\hat{\mathbf{r}}^{t}_{j,1}, \hat{\mathbf{r}}^{t}_{j,2}, \hat{\mathbf{r}}^{t}_{j,3}]$
Recurrent Dense Block (RDB)

Figure 4.2: *Recurrent Dense Block* (RDB) of length three ($n_{\text{layer}} = 3$) using a *Filter Module* (FM) after each *Dense Unit* (DU). Newly computed features $\bar{\mathbf{r}}^t_{j,l}$ are stabilized using temporal information before being added to the global feature state. We use **[ ]** to indicate the concatenation of features.

where $f^{FM}_{j,l}$ is the recurrent filter function, with parameters $\theta^{FM}_{j,l}$, and $\mathbf{m}^{t-1}_{j,l}$ the hidden state of the filter. The hidden state represents an internal memory, encoding all knowledge aggregated from previous time steps. Framing Equation 4.2 in the context of a *Bayes* Filter (Gu *et al.*, 2017), the recurrent filter function has to propagate the belief about the hidden state $\mathbf{m}^{t-1}_{j,l}$ one time step into the future, update the belief using the current input $\bar{\mathbf{r}}^t_{j,l}$ of the filter, and compute an improved estimate $\hat{\mathbf{r}}^t_{j,l}$ of the true feature representation.

A FC-DenseNet can be transformed into our proposed *Recurrent Fully Convolutional DenseNet* (RFC-DenseNet) by using a recurrent version of the *Dense Block*s (see Figure 4.2), which employs a *Filter Module* (FM) after each *Dense Unit* (DU). To compute a robust pixel-wise semantic segmentation $\hat{\mathbf{s}}^t$, RFC-DenseNet utilizes the information of multiple images. These images are propagated through the feature extractor and the subsequent upsampling-path, while taking skip-connections as well as temporal correlations via the *Filter Module*s into account. The weights of all computational units are shared over time. Additionally, we use the same dropout mask for each time step, as suggested by (Gal and Ghahramani, 2016a). The hidden state of each *Filter Modules* is initialized with zeros.

The proposed *Recurrent Dense Blocks* (RDBs) (see Figure 4.2) only add filtered features $\hat{\mathbf{r}}^t_{j,l}$ to the global feature state of the model. Features $\bar{\mathbf{r}}^t_{j,l}$ computed in each *Dense*

*Unit*s are thus derived from already filtered ones or the initial feature set[1], generating a hierarchy of filtered representations. Due to the hierarchical filter nature, RFC-DenseNet can utilize temporal correlations on all abstraction levels. In comparison, a global filter (*i.e.* a non-hierarchical filter) only has access to a sequence of high-level representations (Fayyaz *et al.*, 2016; Valipour *et al.*, 2017). The availability of all features, required to solve aleatoric failures within the filter, is not guaranteed in such a global setting.

The *Recurrent Fully Convolutional DenseNet* (RFC-DenseNet) conceptually decouples temporal dependencies from scene representation, by introducing dedicated *Filter Module*s. This property enables the transformation of any single-image FC-DenseNet into a corresponding multi-image RFC-DenseNet. One could also use the trained weights of the FC-DenseNet to initialize the non-recurrent part of the corresponding RFC-DenseNet. The decoupling additionally increases the transparency of the model, enabling the allocation of additional resources for temporal filtering (using a more complex filter) or hierarchical feature generation (using a deeper architecture)[2].

The proposed filter approach can also be employed in other models, but is especially suitable for the FC-DenseNet architecture. The explicit differentiation of newly computed features $\bar{\mathbf{r}}^t_{j,l}$ and features stored in the global feature state, makes the proposed temporal filtering concept very parameter efficient. Each filter only has to process a small number of newly computed feature maps, resulting in a moderate increase in the total number of model parameters. The distinct focus on a small feature set in each *Filter Module* also reduces the computational complexity of the filter task.

4.3.3 Instances of the Filter Module

We investigated three instances of the *Filter Module* (FM) with increasing complexity. Figure 4.3 visualizes the architecture of the three proposed modules. All *Filter Module*s are based on convolutional *Long Short-Term Memory* (LSTM) cells, which have proven to produce state-of-the-art results on a multitude of spatio-temporal sequence modeling

[1]The initial feature set refers to the features computed in the first convolutional layer of the model.

[2]Depending on the chosen *Filter Module* as well as the training procedure, a mutually exclusive allocation of resources is not necessarily guaranteed.

tasks (Graves, 2014). We adopt the implementation of Xingjian *et al.* (2015), but omit the peephole connections to limit the number of parameters:

$$\begin{aligned}
\mathbf{i}^t_{j,l} &= \sigma(\mathbf{W}^{ei}_{j,l} * \mathbf{e}^t_{j,l} + \mathbf{W}^{hi}_{j,l} * \mathbf{h}^{t-1}_{j,l} + \mathbf{b}^i_{j,l}), && (4.3)\\
\mathbf{f}^t_{j,l} &= \sigma(\mathbf{W}^{ef}_{j,l} * \mathbf{e}^t_{j,l} + \mathbf{W}^{hf}_{j,l} * \mathbf{h}^{t-1}_{j,l} + \mathbf{b}^f_{j,l}), && (4.4)\\
\mathbf{o}^t_{j,l} &= \sigma(\mathbf{W}^{eo}_{j,l} * \mathbf{e}^t_{j,l} + \mathbf{W}^{ho}_{j,l} * \mathbf{h}^{t-1}_{j,l} + \mathbf{b}^o_{j,l}), && (4.5)\\
\mathbf{c}^t_{j,l} &= \mathbf{f}^t_{j,l} \circ \mathbf{c}^{t-1}_{j,l} + \mathbf{i}^t_{j,l} \circ \tanh(\mathbf{W}^{ec}_{j,l} * \mathbf{e}^t_{j,l} + \mathbf{W}^{hc}_{j,l} * \mathbf{h}^{t-1}_{j,l} + \mathbf{b}^c_{j,l}), && (4.6)\\
\mathbf{h}^t_{j,l} &= \mathbf{o}^t_{j,l} \circ \tanh(\mathbf{c}^t_{j,l}), && (4.7)
\end{aligned}$$

where $*$ is the convolutional operator, $\circ$ the *Hadamard* product, and $\mathbf{e}^t_{j,l}$ the input of the convolutional LSTM. The input, forget, and output gate are denoted by $\mathbf{i}^t_{j,l}$, $\mathbf{f}^t_{j,l}$, and $\mathbf{o}^t_{j,l}$, respectively. For brevity, we summarize the hidden state $\mathbf{h}^t_{j,l}$ and cell state $\mathbf{c}^t_{j,l}$ of the convolutional LSTM by $\mathbf{m}^{t-1}_{j,l}$ (see Equation 4.2).

A property of all proposed *Filter Module*s are matching dimensions between the unfiltered $\bar{\mathbf{r}}^t_{j,l}$ and filtered $\hat{\mathbf{r}}^t_{j,l}$ representation. The growth rate of the global feature state of the RFC-DenseNet is thus identical to the one of the corresponding FC-DenseNet. Analogous to the computational units of FC-DenseNet, we use *pre-activations* (He *et al.*, 2016b) in all *Filter Module*s. To regularize the *Filter Module*s, we use the variational inference-based dropout variant proposed by Gal and Ghahramani (2016a) and apply it to the convolutional LSTM (Gal and Ghahramani, 2016b). The *input-to-hidden* convolutional kernels of the LSTM ($\mathbf{W}^{e*}_{j,l}$) have a filter size of 3×3 and the filter size of the *hidden-to-hidden* kernels ($\mathbf{W}^{h*}_{j,l}$) is chosen dependent on the position (*i.e.* the index j) of the corresponding *Recurrent Dense Block* (see Section 4.4.2 and Table 4.4 for further details).

The first instance of the *Filter Module* FM_{ff} uses a single convolutional *Long Short-Term Memory* (LSTM) cell following the two *pre-activation* layers (*Batch Normalization* (BN) (Ioffe and Szegedy, 2015) and *Rectified Linear Unit* (ReLU) (Nair and Hinton, 2010)). The number of feature maps stored in the cell state $\mathbf{c}^t_{j,l}$ of the convolutional LSTM matches the number of feature maps of the unfiltered representation $\bar{\mathbf{r}}^t_{j,l}$. This property restricts the filter capabilities but also limits the number of required parameters.

The second instance of the *Filter Module* FM_{res} uses the concept of deep residual learning (He *et al.*, 2016a) and applies it to the first *Filter Module*. Instead of directly fitting

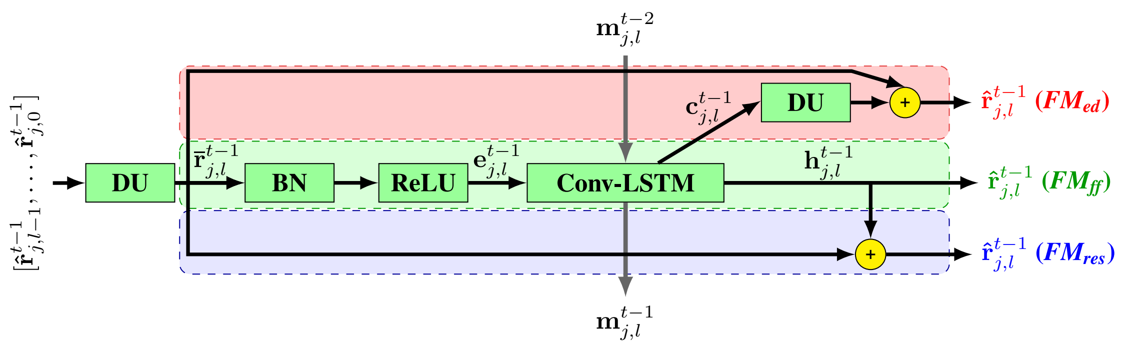

$\mathbf{m}_{j,l}^{t-2}$
$[\hat{\mathbf{r}}_{j,l-1}^{t-1}, \dots, \hat{\mathbf{r}}_{j,0}^{t-1}]$
DU
$\bar{\mathbf{r}}_{j,l}^{t-1}$
BN
ReLU
$\mathbf{e}_{j,l}^{t-1}$
Conv-LSTM
$\mathbf{c}_{j,l}^{t-1}$
DU
+
$\hat{\mathbf{r}}_{j,l}^{t-1}$ (FM_{ed})
$\mathbf{h}_{j,l}^{t-1}$
$\hat{\mathbf{r}}_{j,l}^{t-1}$ (FM_{ff})
+
$\hat{\mathbf{r}}_{j,l}^{t-1}$ (FM_{res})
$\mathbf{m}_{j,l}^{t-1}$

Figure 4.3: Proposed instances of the *Filter Module* (FM) which are used in the *Recurrent Dense Block*s to temporally filter the feature representation of the *Dense Unit*s (DUs). All *Filter Module*s are based on a convolutional *Long Short-Term Memory* (LSTM) cell following two pre-activation layers: *Batch Normalization* (BN) and *Rectified Linear Unit* (ReLU). The filters *FM*$_{res}$ and *FM*$_{ed}$ additionally use a identity skip connection and *FM*$_{ed}$ employs an encoder-decoder architecture. To better illustrate the differences, we visualize the *Filter Module*s in one diagram. The *Filter Module FM*$_{ed}$, for example, is defined by the sub-graph which connects the unfiltered representation with the first filtered output. In the experiments, each RFC-DenseNet architecture uses only one type of *Filter Module*. The weights of the *Filter Module*s are not shared.

the desired filter function $f^{FM}(\bar{\mathbf{r}})$, we let the convolutional LSTM fit the residual function $g^{FM}(\bar{\mathbf{r}}) = f^{FM}(\bar{\mathbf{r}}) - \bar{\mathbf{r}}$. The filtered representation can be recovered by solving for the original mapping: $\hat{\mathbf{r}} = g^{FM}(\bar{\mathbf{r}}) + \bar{\mathbf{r}}$. The identity skip connection in combination with pre-activations ensures a direct information and gradient flow (He *et al.*, 2016b). Using the residual network interpretation of Veit *et al.* (2016): The resulting RFC-DenseNet has many paths connecting the input with the output. One of these paths resembles the corresponding FC-DenseNet without *Filter Module*s. This FC-DenseNet can be recovered from the RFC-DenseNet by learning to return zero in all convolutional LSTMs (*e.g.* by closing the output gate of all LSTMs). As introduced in Section 4.3.2, the *Filter Module*s are intended to compute an improved estimate of the true feature representation. The introduction of identity skip connections might facilitate this property, as argued by Greff *et al.* (2017). The number of parameters of *FM*$_{res}$ is identical to the one of *FM*$_{ff}$ and the added computational cost is extremely small.

The third proposed instance of the *Filter Module FM*$_{ed}$ alleviates the limitation on the complexity of the filter, introduced by matching feature dimensions of the unfiltered representation and the cell state. This instance employs an encoder-decoder structure, consisting of a convolutional LSTM and a *Dense Unit*. The number of feature maps stored in the convolutional LSTM can be chosen to be a multitude α_{ed} of the number of unfiltered feature maps. This filter instance can be considered an extension of *FM*$_{res}$,

Class	*background*	*boarders / walls*	*dyn. squares / digit 0*	*dyn. squares / digit 1*	*dyn. squares / digit 2*	*dyn. squares / digit 3*	*dyn. squares / digit 4*	*dyn. squares / digit 5*	*dyn. squares / digit 6*	*dyn. squares / digit 7*	*dyn. squares / digit 8*	*dyn. squares / digit 9*	*static squares*	*circles*
Color	🟪	🟨	🟥	🟥	🟨	🟩	🟩	⬜	⬛	🟪	🟪	🟩	🟦	⬛

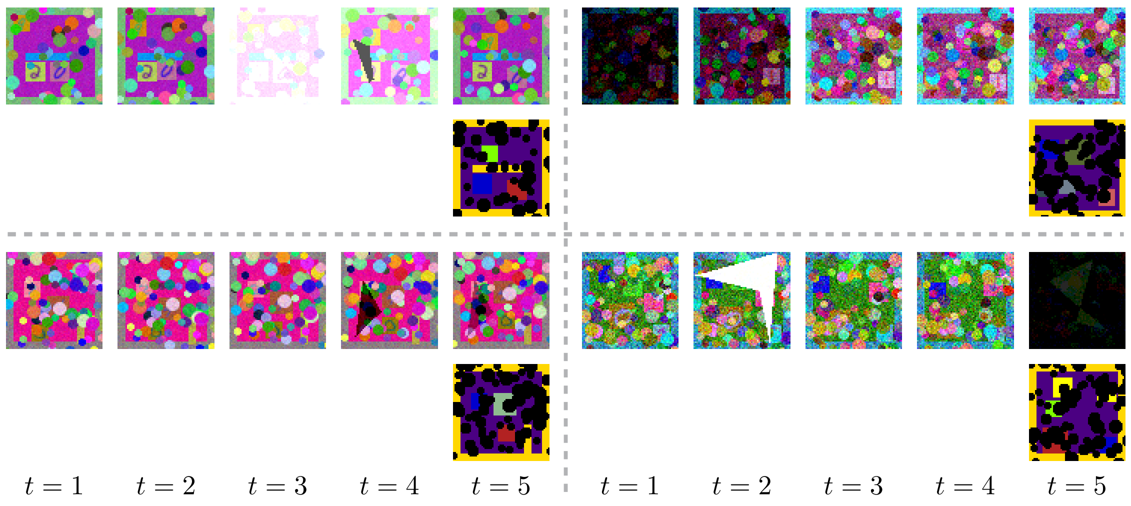
Table 4.1: Semantic classes of the simulated dataset with the corresponding label color coding used in the figures.

since it also uses a residual unit-like structure. The number of parameters of FM_{ed} is adjustable by changing the scaling factor α_{ed}.

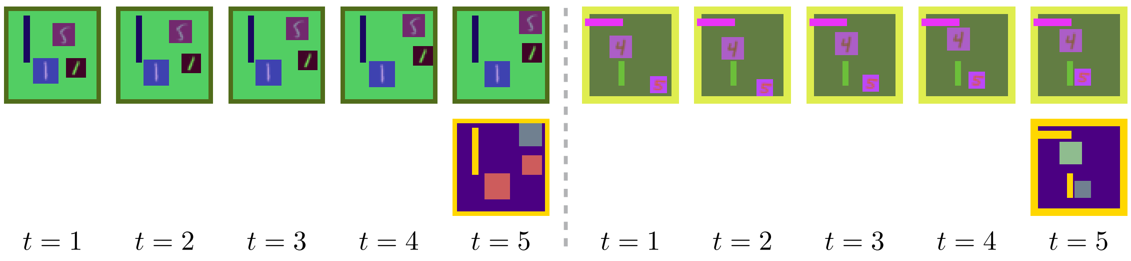
4.4 Experiments

4.4.1 Dataset

To systematically evaluate the proposed architectures, we use a simulated video dataset. This enables a focus on aleatoric failures which are often underrepresented in real-world benchmark datasets. The simulated video sequences emulate a 2D environment of 64×64 pixels, in which squares represent dynamic and static objects, rectangles represent boarders and walls, and circles represent moving foreground objects. Each square is marked with a digit sampled from the *MNIST* dataset (LeCun *et al.*, 1998). The dynamic squares elastically collide with all other squares, boarders, and walls. The moving foreground circles occlude each other, as well as all other objects. Circles can leave the frame and elastically collide with dedicated boarders located outside of the visible field of view. A circle that leaves the frame will consequently reappear at a later point in time. Due to the visible boarders surrounding the environment, dynamic squares are not able to leave the frame. The number of the objects, their size and color, the color of the *MNIST* digits, the color of the background, as well as the initial velocity of all dynamic objects is randomly sampled for each sequence. The number of dynamic squares per sequence is greater than the number of static squares. Each sequence contains two to four dynamic squares, whereas only one or no static square.



(a) Four example sequences of the test dataset with corresponding ground truth labels.



(b) Two example sequences of the clean test dataset with corresponding ground truth labels.

Figure 4.4: Example sequences of the test dataset as well as the clean test dataset with corresponding ground truth labels. The sequences emulate a 2D environment of 64×64 pixels, in which squares represent dynamic and static objects, rectangles represent boarders and walls, and circles represent moving foreground objects. The circles occlude all other objects as well as each other. Dynamic squares elastically collide with all other squares, boarders, and walls. A semantic label is only available for the last frame (*i.e.* $t = 5$) of each sequence. The 14 semantic classes as well as the corresponding label color coding are listed in Table 4.1.

To simulate aleatoric failures, we perturb the data with instances of the four failure classes proposed in Section 4.1. **Noise** is simulated by adding independent *Gaussian* noise with zero mean to each pixel. The variance of the noise is independently sampled for each sequence from the interval $[0, 0.05]$. The resulting pixel values are clipped to the interval $[0, 1]$ to obtain valid images. **Occlusions** are introduced by moving foreground circles, which occlude each other, as well as all other objects of the scene. These circles should especially impede the classification of *MNIST* digits which are plotted on all squares. To simulate **missing information**, we increase or decrease the intensity of pixels by a random value and let this offset decay over time. In $30\,\%$ of the sequences, we add the effect to full images, starting from a randomly selected frame. Additionally, we apply the perturbation to polygonal subregions of images. The shape of the subregions as well as their number is randomly sampled for each sequence. Each subregion-based perturbation starts at a randomly selected time step. To obtain valid images, we once again clip the pixel values to the interval $[0, 1]$ after applying the perturbations. **Ambiguities** are simulated by using different classes for static squares and for dynamic squares. Thus, the information of a single image is not sufficient to infer unambiguously the class of a square.

In total, our dataset consists of 25,000 sequences of length 5 which are split into 20,000 training, 4,000 validation, and 1,000 test sequences. Each split is simulated using all aleatoric failure classes. We additionally generate a test dataset containing 1,000 sequences with no added aleatoric perturbations. This dataset does not contain any foreground circles and only dynamic squares. The images are not altered by *Gaussian* noise or by the transformation used to simulated missing information. In the remainder of this chapter, we will refer to the second test dataset without added aleatoric perturbations as the *clean test dataset*. For the pixel-wise semantic segmentation task we define 14 classes: *background*, *boarders / walls*, *static squares*, *circles*, and *dynamic squares* with an individual class per digit. Table 4.1 contains a list of the semantic classes with the corresponding label color coding. Example sequences of the test dataset as well as the clean test dataset with corresponding labels are shown in Figure 4.4. A label is only available for the last frame of each sequence.

4.4.2 Implementation Details

We perform an extensive hyperparameter optimization to determine the best single-image FC-DenseNet architecture. The optimized architectural parameters are listed in Table 4.2, where n_{pool} is the depth of the model, n_{layer} the number of layers per *Dense Block*, n_{feature} the number of newly added features per *Dense Unit*, and n_{conv} the number of features computed in the first convolutional layer. A detailed description of the parameters is given in Section 4.3.1 as well as Figure 4.1. In addition to the architectural parameters, we also optimize all training and regularization parameters. The configuration of the best model FCD_b is listed in column two of Table 4.2. We additionally use a smaller version of the single-image FC-DenseNet, which has less layers per *Dense Block* as well as a smaller number of newly computed features per *Dense Unit*. The smaller

	FCD$_b$	*FCD*$_s$	*TM*$_{st}$
n_{pool}	2	2	2
n_{layer}	9	7	7
n_{feature}	12	8	10
n_{conv}	48	48	56

Table 4.2: Structural configuration of the best (*FCD*$_b$) and small FC-DenseNet (*FCD*$_s$) as well as the temporal model *TM*$_{st}$. The temporal model *TM*$_{st}$ is structurally identical to a normal FC-DenseNet but receives a stacked version of the image sequence as input. The sequence frames are concatenated along the feature dimension to construct a valid three-dimensional input tensor. n_{pool}: depth of the model; n_{layer}: number of layers per DB; n_{feature}: number of newly added features per DU; n_{conv}: number of features computed in the first convolutional layer.

RFC-DenseNet	*Filter Module* (FM)
RFCD$_{ff}$	*FM*$_{ff}$
RFCD$_{res}$	*FM*$_{res}$
RFCD$_{ed1}$	*FM*$_{ed}$, $\alpha_{ed} = 1$
RFCD$_{ed2}$	*FM*$_{ed}$, $\alpha_{ed} = 2$

Table 4.3: Type and parametrization of the *Filter Module* used in the four proposed versions of the RFC-DenseNet architecture.

j	0	1	2	3	4
f	9	5	3	5	9

Table 4.4: Spatial size $f \times f$ of the hidden-to-hidden kernels $\mathbf{W}_{j,l}^{h*}$ used in the jth *Recurrent Dense Block* of all RFC-DenseNet architectures.

model *FCD*$_s$ (see column three in Table 4.2) is used as the basis of our recurrent models to reduce training time.

In total, we train seven temporal models. Four RFC-DenseNets using our filter concept of Section 4.3.2, a recurrent model which uses a non-hierarchical filter approach *RM*$_{gf}$, and two non-recurrent models, *TM*$_{3D}$ and *TM*$_{st}$. The four RFC-DenseNet architectures are constructed by transforming *FCD*$_s$ according to our proposed filter concept. The models differ in terms of the choice of *Filter Module*. A summary of the type and parametrization of the *Filter Module* used by each model is given in Table 4.3. The spatial size $f \times f$ of the hiden-to-hidden kernels $\mathbf{W}_{j,l}^{h*}$ of all *Filter Module*s is set dependent on the index j of the corresponding *Recurrent Dense Block* (see Table 4.4). Model *RM*$_{gf}$ is constructed by extending *FCD*$_s$ with a filter which is placed between the last *Dense Block* and the output convolution. The added global filter temporally integrates high-level features computed by the last *Dense Block*. The filter is implemented using *FM*$_{ed}$ with $\alpha_{ed} = 0.625$ and a hidden-to-hidden filter size of 9. *RM*$_{gf}$ has approximately the same number of parameters as all RFC-DenseNets, besides *RFCD*$_{ed2}$. A model similar to *RM*$_{gf}$ was proposed by Valipour *et al.* (2017). We also tested other *Filter Module*s, however *FM*$_{ed}$ proved to be the best option for *RM*$_{gf}$. In addition to the five models using recurrent *Filter Module*s, we also evaluate two non-recurrent temporal FC-DenseNets. Model *TM*$_{3D}$ is structurally identical to *FCD*$_s$ except for the use of 3D convolutions (Tran *et al.*, 2015) instead of regular 2D convolutions in all *Dense Unit*s. *TM*$_{3D}$ preserves the temporal information of the input sequence throughout the model and uses 3D convolutions with a filter size of $3 \times 3 \times 3$ to compute spatio-temporal features. Model TM_{st} is a regular FC-DenseNet which operates on a stacked version of the input sequence (Simonyan and Zisserman, 2014). The frames of the input sequence are concatenated along the feature dimension, producing a valid three-dimensional input tensor. The parametrization of TM_{st} is given in column four of Table 4.2. We also experimented with larger versions of TM_{st}, but were not able to achieve a better segmentation performance.

Following the approach of Kendall and Gal (2017), we implement all models as *heteroscedastic Bayesian Neural Networks* (BNNs). The models use *Monte Carlo* dropout and place a *Gaussian* distribution over the *softmax* logits to model epistemic uncertainty as well as heteroscedastic aleatoric uncertainty. The parameters of the *Gaussian* distribution are predicted by the *Neural Network* (NN). In accordance to Kendall and Gal (2017), we use 50 *Monte Carlo* samples. Kendall and Gal (2017) show the ability of *heteroscedastic* BNNs to outperform comparable standard NNs on semantic segmentation and depth regression benchmarks and highlight the ability of their proposed loss to be more robust to noisy data.

4.4.3 Results

In Table 4.5a, we summarize the mean IoU score of all models on the test and the clean test dataset. In addition, we list in Table 4.5b per-class IoU scores computed on the test data. To increase readability, an aggregated score (*i.e.* the mean IoU score) is reported for the classes representing dynamic squares. All metrics are computed using the labeled 5th frame of each test sequence as well as the corresponding semantic prediction of the models. The predictions of all temporal models are conditioned on full input sequences. Models FCD_b and FCD_s, on the other hand, only use the information of a single image to compute a pixel-wise semantic segmentation. In addition to the quantitative results, we visualize in Figure 4.5 the semantic prediction of our best RFC-DenseNet architecture $RFCD_{ed1}$ as well as the best single-image FC-DenseNet FCD_b for four sequences of the test dataset. These examples illustrate the ability of our temporal RFC-DenseNet architecture to avoid aleatoric failures of a well-trained single-image model.

All models which utilize temporal information significantly outperform the single-image FC-DenseNets (FCD_b and FCD_s) on the test dataset, showing the benefit of temporal

	FCD_b	FCD_s	$RFCD_{ff}$	$RFCD_{res}$	$RFCD_{ed1}$	$RFCD_{ed2}$	RM_{gf}	TM_{3D}	TM_{st}
Test	45.4 %	43.4 %	67.1 %	67.9 %	**69.2** %	68.4 %	65.4 %	60.8 %	50.4 %
Clean test	93.1 %	91.0 %	92.0 %	92.2 %	**94.4** %	93.0 %	92.1 %	89.8 %	89.4 %

(a) Mean IoU score (%) of all models on the test dataset as well as the clean test dataset.

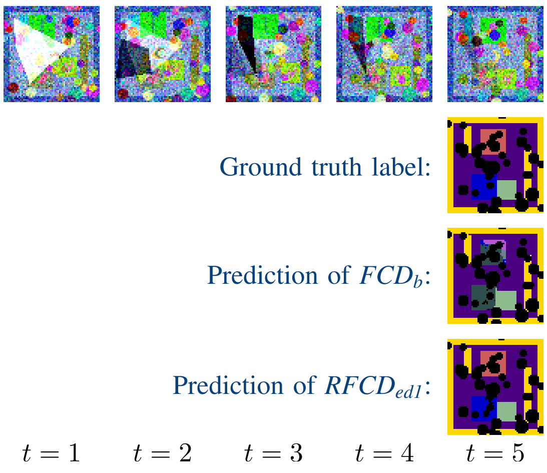
	FCD_b	FCD_s	$RFCD_{ff}$	$RFCD_{res}$	$RFCD_{ed1}$	$RFCD_{ed2}$	RM_{gf}	TM_{3D}	TM_{st}
background	89.9 %	88.8 %	94.0 %	94.3 %	94.8 %	**95.0** %	93.7 %	93.7 %	91.9 %
boarders / walls	92.0 %	91.2 %	94.7 %	94.8 %	95.3 %	**95.4** %	94.6 %	94.6 %	93.5 %
dyn. squares (mean)	36.5 %	33.8 %	57.2 %	58.2 %	**59.7** %	58.6 %	55.4 %	48.7 %	35.5 %
static squares	2.8 %	3.3 %	88.8 %	88.5 %	**89.6** %	89.0 %	83.2 %	86.0 %	77.6 %
circles	86.8 %	85.6 %	90.4 %	91.0 %	91.8 %	**91.9** %	90.6 %	90.2 %	87.7 %

(b) Per-class IoU scores (%) computed on the test dataset. To increase readability, we report an aggregated score for the classes representing dynamic squares.

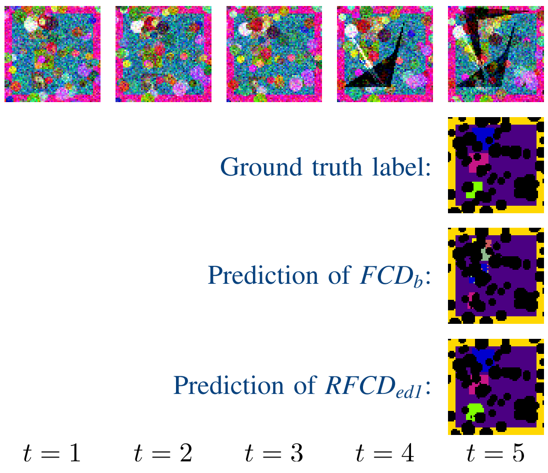
Table 4.5: Quantitative performance metrics of all introduced models. The first two models (*FCD_b* and *FCD_s*) utilize the information of a single input frame, all other models are temporal models that perform a prediction conditioned on full input sequences. Models *$RFCD_*$* are different instances of the proposed *Recurrent Fully Convolutional DenseNet* (RFC-DenseNet) architecture. *TM_{3D}* and *TM_{st}* are non-recurrent temporal models and *RM_{gf}* is a recurrent model which uses a non-hierarchical (*i.e.* global) filter approach.

filtering. The best temporal model *$RFCD_{ed1}$* improves the mean IoU score by 23.8 % compared to the best single-image model *FCD_b*. In comparison to *FCD_s*, which is the basis of the *$RFCD_{ed1}$* architecture, the mean IoU score is improved by 25.8 %. Looking at the temporal model *TM_{st}* with the worst performance on the test dataset, we still observe a 5 % better mean IoU score compared to *FCD_b*. The difference between single-image and multi-image models is especially visible when focusing on the class *static squares*. The information provided by one image is not sufficient to distinguish a static square from a dynamic square. By using the information of previous frames, a temporal model can resolve such ambiguities. On the clean test dataset, the performance difference between *FCD_b* and the temporal models is comparatively small, suggesting that the additional information provided by previous frames especially benefits the reduction of aleatoric failures. The superior performance of *FCD_b* on the clean test data compared to most of the temporal models can most likely be attributed to its increased non-temporal depth and width.

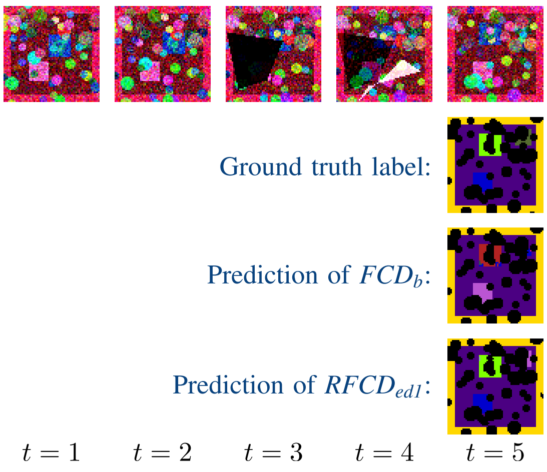
The mean IoU scores of the different RFC-DenseNet instances suggest a correlation between filter complexity and segmentation performance. RFC-DenseNet architecture *$RFCD_{ff}$*, which uses the most simple *Filter Module FM_{ff}*, achieves a mean IoU score of 67.1 % on the test dataset. By adding a residual connection to the *Filter Module*s



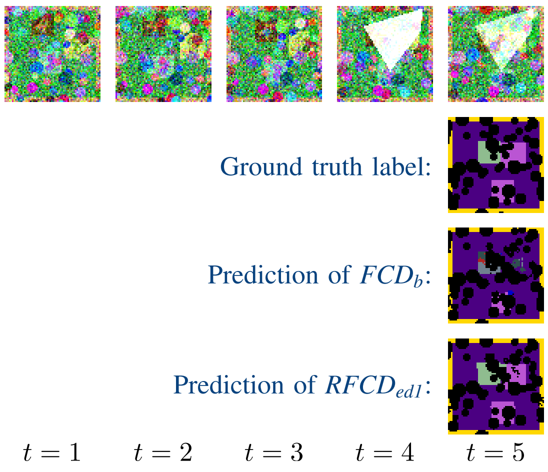
(a) $RFCD_{ed1}$ resolves heavy occlusion of the upper dynamic square marked with digit 1 (🟥) and is able to recognize the static square (🟦).



(b) Using information of previous time steps, $RFCD_{ed1}$ is able to suppress noise as well as to reconstruct missing information (🟥 and 🟩). Additionally, it can infer the static motion state of the upper square (🟦).



(c) $RFCD_{ed1}$ can infer the static motion state of the lower square (🟦) and is able to resolve heavy occlusion of one of the two upper squares (🟩).



(d) $RFCD_{ed1}$ can resolves the miss-classification of the two upper dynamic squares (🟪 and 🟩) and is able to improve the segmentation of the lower square (🟪) compared to the single-image baseline FCD_b.

Figure 4.5: Example predictions of the best RFC-DenseNet model $RFCD_{res1}$ as well as the best single-image model FCD_b. For four sequences of the test dataset, we visualize the ground truth pixel-wise semantic label as well as the corresponding prediction of $RFCD_{ed1}$ and FCD_b. The examples highlight scenes in which our proposed architecture is able to use temporal information to significantly reduce the impact of noise, missing information, and occlusions compared to the single-image FCD_b. The information provided by previous time steps additionally enable $RFCD_{ed1}$ to distinguish dynamic from static squares. A list of the semantic classes with the corresponding label color coding is given in Table 4.1.

(model $RFCD_{res}$) the score can be improved to 67.9 %. The best RFC-DenseNet $RFCD_{ed1}$ with a mean IoU score of 69.2 % uses the encoder-decoder *Filter Module*. The difference introduced by the type of *Filter Module* is, however, relatively small. Model $RFCD_{ed2}$

performs unexpectedly poor in comparison to $RFCD_{ed1}$. The use of other recurrent regularization techniques (*e.g. zoneout* (Krueger *et al.*, 2017)) might possibly benefit the performance of $RFCD_{ed2}$.

The hierarchical RFC-DenseNet models outperform the non-hierarchical, recurrent baseline RM_{gf} by $1.7\,\%$ to $3.8\,\%$ on the test dataset, indicating a better ability of the trained RFC-DenseNet models in resolving aleatoric perturbations. We suspect that our hierarchical filter concept more efficiently utilizes temporal information, compared to the non-hierarchical model. Taking only the dynamic squares with marked *MNIST* digits into account, the performance difference increases further to $4.3\,\%$. For these classes, it is important to model low-level temporal dependencies. The non-hierarchical approach possibly suffers, because of the loss in scene details from lower layers to upper layers.

The performance of the non-recurrent models, TM_{3D} and TM_{st}, is inferior to the recurrent ones. TM_{3D} achieves a mean IoU score of $60.8\,\%$ on the test dataset, which is $8.4\,\%$ less than the score of the best RFC-DenseNet. Model TM_{st} has the worst performance of all temporal models with a mean IoU score of $50.4\,\%$. We suspect that the superior performance of the RFC-DenseNet models and RM_{gf} is due to the better fit of the inductive bias of these architectures. In particular, the recurrent nature of the filters as well as the use of explicit *Filter Module*s could be favorable features.

In Figure 4.5, we show pixel-wise semantic segmentations of $RFCD_{ed1}$ and FCD_b for four sequences of the test dataset that are challenging for a single-image model. The predictions highlight the ability of our RFC-DenseNet model to suppress noise, derive additional temporal object properties, and to resolve missing information as well as occlusions.

4.5 Conclusion

In this chapter, we proposed a parameter efficient approach to temporally filter the representations of the *Fully Convolutional DenseNet* (FC-DenseNet) (Jégou *et al.*, 2017) in a hierarchical fashion, while conceptually decoupling temporal dependencies from scene representation. The resulting *Recurrent Fully Convolutional DenseNet* (RFC-DenseNet) can utilize information of multiple time steps to create a more robust and reliable semantic environment representation.

Using a synthetic video dataset, we showed the benefits of using temporal information in regard to aleatoric failures and evaluated the advantages introduced by our recurrent and hierarchical filtering concept. Our best RFC-DenseNet was able to significantly outperform common non-recurrent, temporal architectures and achieved a $3.8\,\%$ better mean IoU score compared to a similar non-hierarchical, recurrent baseline model. Based on exemplary predictions, we additionally showcased the ability of our model to suppress noise, derive additional temporal object properties, and to resolve missing information as well as occlusions.

In addition to a good prediction performance, an increased level of transparency and interpretability is advantageous for models used in safety-critical applications. Higher transparency and interpretability can be achieved, for example, by introducing human interpretable intermediate representations. Such representations facilitate debugging of a model, simplify the introduction of prior knowledge as well as constraints, and ease the utilization of additional auxiliary losses. The conceptual decoupling of temporal dependencies from scene representation in our proposed RFC-DenseNet is already a first small step towards a more transparent model. In the next chapter, we focus on increasing the transparency, explicitness, and interpretability of temporal models by proposing a functionally modularized temporal representation filter.

Following the publication of the contents of this chapter (Wagner *et al.*, 2018b), several methods and models have been proposed for solving aleatoric failures of perception systems. An overview of more recent approaches can be found in Section 5.6.

5 Functionally Modular and Interpretable Temporal Filtering

The performance of autonomous vehicles heavily relies on their ability to generate a robust representation of the environment. Deep neural networks have greatly improved vision-based perception systems but still fail in challenging situations, *e.g.* sensor outages or adverse weather. These failures are often introduced by data-inherent perturbations, which significantly reduce the information provided to the perception system.
In this chapter, we propose a functionally modularized temporal filter, which stabilizes an abstract feature representation of a single-frame segmentation model using information of previous time steps. Our filter module splits the filtering task into multiple less complex and more interpretable subtasks. The basic structure of the filter is inspired by a *Bayes* estimator consisting of a prediction and an update step. To make the prediction more transparent, we implement it using a geometric projection and estimate its parameters. In addition, this enables the decomposition of the filter task into static representation filtering and low-dimensional motion filtering. The proposed filter structure enables our model to cope with heavy noise and missing information, *e.g.* a missing image frame due to a sensor outage. Using photorealistic, synthetic video data, we show the ability of the proposed architecture to reduce the performance degradation caused by data-inherent perturbations. The experiments especially highlight the advantages introduced by an interpretable and explicit filter module. We believe that a high degree of interpretability is an important property of models, as it facilitates debugging and verification tasks, simplifies the integration of domain knowledge, and eases the utilization of additional auxiliary losses.

5.1 Introduction

The performance of autonomous vehicles, such as self-driving cars or mobile robots, is heavily influenced by their ability to generate a robust representation of their surroundings. Errors in the environment representation are propagated to subsequent processing steps and are hard to recover. In order to increase the reliability and safety of autonomous vehicles, robust methods / models for observing and interpreting the environment are required. Common measures for ensuring safety include the ability to debug models, validate system behavior, assess risk factors, and verify predictions. Besides prediction performance, it is therefore advantageous to also consider transparency and interpretability during method / model development. In addition, a high degree of transparency and interpretability provides further advantages, such as a simplified integration of prior knowledge as well as domain constraints and an easier introduction of additional auxiliary losses.

Deep learning-based methods have greatly advanced the state-of-the-art of perception systems. Especially vision-based perception benchmarks (*e.g.* *Cityscapes* (Cordts *et al.*, 2016) or *Caltech* (Dollár *et al.*, 2009b)) are dominated by approaches utilizing deep neural networks. From a safety perspective, a major disadvantage of such benchmark datasets is their recording during daytime under idealized environment conditions. To deploy autonomous vehicles in an open world scenario without any human supervision, one not only has to guarantee their reliability in good conditions, but also has to make sure that they still work in challenging situations (*e.g.* during sensor outages or in adverse weather conditions). One source of such challenges are perturbations inherent in the data, which significantly reduce the information provided to the perception system. We denote failures originating from data-inherent perturbations as aleatoric failures in accordance to the classification of uncertainties (Kiureghian and Ditlevsen, 2009; Kendall and Gal, 2017). These failures cannot be resolved using a more powerful model or additional training data. To solve aleatoric failures, one has to enhance the information provided to the perception system. This can be achieved by fusing the information of multiple sensors (see Section 2), by utilizing context information, by using a better sensor, and / or by considering temporal information. A second class of failures are epistemic failures, which are model or dataset dependent. They can be mitigated by using more training data and / or a more powerful perception model (Kiureghian and Ditlevsen, 2009). Please refer to Section 4.1 for a more detailed discussion of the two failure modes.

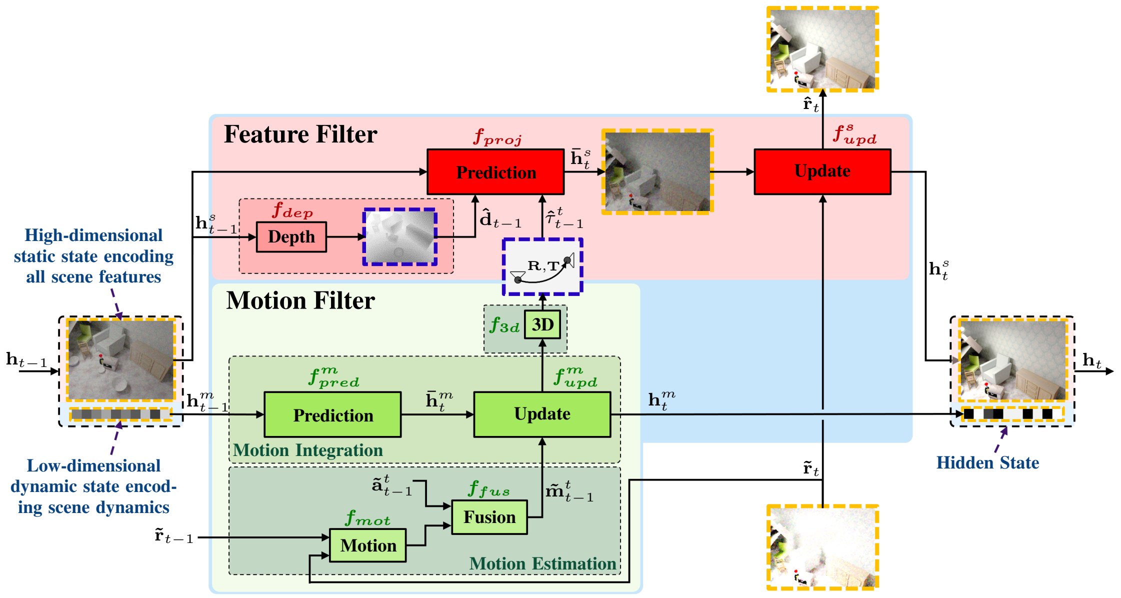


Figure 5.1: Functionally modularized temporal filter used to compute an improved estimate $\hat{\mathbf{r}}_t$ of the encoder representation. The filter task is decomposed into less complex and more transparent subtasks by splitting the hidden state $\mathbf{h}_t$ into a high-dimensional static state $\mathbf{h}^s_t$ and a low-dimensional dynamic state $\mathbf{h}^m_t$. Separate filter modules are employed to filter the two sub-states. The subfilters adopt the basic structure of a recursive *Bayesian* estimator and utilize auxiliary modules to compute human interpretable intermediate representations and to preprocess features. For illustration purposes, abstract feature representations (*e.g.* $\tilde{\mathbf{r}}_t$ and $\hat{\mathbf{r}}_t$) are displayed as images framed by an orange dashed line. Human interpretable intermediate representations ($\hat{\mathbf{d}}_{t-1}$ and $\hat{\tau}^t_{t-1}$) are highlighted by a blue dashed frame. The pictures used to illustrate the various representations are altered images of the *SceneNet RGB-D* (McCormac *et al.*, 2017) dataset.

In this work, we focus on tackling aleatoric failures of a single-frame semantic segmentation model using temporal consistency. Temporal integration is achieved by recurrently filtering a representation of the model by means of a functionally modularized filter. A structural overview of our proposed representation filter is depicted in Figure 5.1. In contrast to other available approaches, our filter consists of multiple submodules, decomposing the filter task into less complex and more transparent subtasks. The basic structure of the filter is inspired by a recursive *Bayesian* estimator, consisting of a prediction and an update step. Within the prediction step, the feature representation of the previous time step is propagated one step into the future and in the update step the predicted representation is fused with more recent information from the image of the current time step (see feature filter in Figure 5.1). We model the prediction of the representation as an explicit geometric projection given estimates of the scene geometry and the scene dynamics. The scene geometry and dynamics are represented as a per-pixel depth and a 6-DoF camera motion. Both parameters are estimated within the filter using two task-specific subnetworks.

The decomposition of the prediction task into a transformation based on a physical model as well as a depth and a motion estimation introduces several advantages. Instead of having to learn dynamics of a high-dimensional representation, we enable modeling motion separately in a low-dimensional feature space. The overall filter can therefore be subdivided into two subfilters: a motion filter, which predicts and integrates low-dimensional camera motion, and a feature filter, which handles the integration and prediction of abstract scene features.

An advantage of our approach is its improved transparency, interpretability, and explicitness. Within the filter, we estimate three human interpretable representations: a depth map, the camera motion, and the weight matrix (update gate) of our update module. These representations can be used to inspect the functionality of the model, to split the filter into pre-trainable subnetworks, or to debug and validate network behavior. In contrast to other methods, the proposed filter also works in cases when the current image is not available. For example, methods that use the optical flow fail in such situations due to their inability to compute a meaningful warping.

Using a photorealistic, synthetic video dataset, we evaluate our proposed functionally modularized temporal filter as well as its subcomponents. First, we highlight the possibility to pre-train subcomponents, evaluate them, and inspect their behavior. The analysis of the feature update module verifies the capability of our filter to integrate information over time. In addition, we evaluate the ability of our motion filter to propagate and aggregate dynamics over time. In a second step, we compare the performance of a segmentation model using our modular filter (FMTNet) with a single-image baseline as well as a feature-level filter approach without task-specific modeling. The results indicate a better ability of our filter in coping with missing data. Finally, we show predictions of FMTNet and use the human interpretable intermediate representations of our filter to inspect model behavior.

The functionally modularized temporal representation filter presented in this chapter was first published at the British Machine Vision Conference (Wagner *et al.*, 2018a).

5.2 Related Work

In this section, we introduce approaches that use temporal information to tackle aleatoric failures of segmentation models and assess their interpretability. We classify the related work into three categories: feature-level filtering, temporal post-processing, and spatio-temporal fusion.

Feature-level temporal filtering. A common approach to temporally stabilize network predictions are feature-level filters. These filters are applied to one or several feature representations, which are integrated using information of previous time steps. Several works implement such a filter using fully-learned, model-free architectures[1]. Fayyaz *et al.* (2016) and Valipour *et al.* (2017) generate a feature representation for each image in a sequence and use RNNs to temporally filter them. The resulting filtered representation is post-processed and up-sampled to generate the semantic segmentation. Jin *et al.* (2017b) utilize a sequence of previous images to predict a feature representation of the current image. The predicted representation is fused with the one of the current image and propagated through a decoder network. Yurdakul and Yemez (2017) evaluate different recurrent network structures to temporally integrate the joint feature representation of RGB and depth data. The *Recurrent Fully Convolutional DenseNet* of Wagner *et al.* (2018b) utilizes a hierarchical filter concept to increase the robustness of a segmentation model (see Chapter 4). Being model-free, these filters usually require many parameters and are harder to train. Due to their low interpretability, it is difficult to include constraints, to inspect or debug their behavior, and thus to perform a safety analysis.

A second class of feature-level filters utilizes a partially model-based approach to temporally integrate features. These approaches use an explicit model to implement the temporal propagation of features and learn a subnetwork which fuses the propagated features with features of the current time step. A common model for implementing the propagation is the optical flow. The replacement field parametrizing the flow can be predicted in the model (Vu *et al.*, 2019) or computed using classical methods (Gadde *et al.*, 2017; Nilsson and Sminchisescu, 2018). These models often fail to resolve aleatoric failures. This is due to their dependence on the availability of the current frame. More sophis-

[1]We define model-free architectures as architectures which are designed without taking task-specific domain knowledge or knowledge about the physical world into account. These architectures rely on more general structures (*e.g.* common CNN or RNN architectures) and fully learn the expected prediction behavior from data.

ticated feature propagation models exist (Zhou *et al.*, 2017; Mahjourian *et al.*, 2017; Mahjourian *et al.*, 2018; Yin and Shi, 2018), which additionally constrain the transformation. Such a model was used by Radwan *et al.* (2018) to temporally aggregate learned features within a multi-task model. Our model is also partially model-based using a more sophisticated propagation model that is similar to the one by Radwan *et al.* (2018). In contrast to all presented model-based approaches, our filter does not depend on the availability of the current frame. Therefore, it is especially well-suited to tackle aleatoric failures, while still being transparent due to its modularity. Model-based and partially model-based architectures contain human interpretable representations (*e.g.* optical flow fields), which aid an engineer to inspect, debug, and validate the model. In addition, it is usually much easier to integrate auxiliary losses during training and to enforce physical constraints.

Post-processing-based temporal integration. Some approaches use post-processing steps to integrate predictions of a single-frame segmentation model. Lei and Todorovic (2016) propose the *Recurrent-Temporal Deep Field* model for video segmentation, which combines a convolutional network, a recurrent temporal restricted Boltzmann machine, as well as a conditional random field. Kundu *et al.* (2016) propose a long-range spatio-temporal regularization using a conditional random field operating on a Euclidean feature space, optimized to minimize the distance between features associated with corresponding points in the scene. Our temporal integration approach differs from post-processing methods, due to the integration of rich feature representations instead of segmentations. The modular structure of our filter, with its human interpretable representations, makes it also more transparent. Additionally, feature-level filters are more general and can easily be integrated into any other network architecture (*e.g.* an object detection model). Several post-processing-based approaches use the optical flow, which is not always reliably computable, especially in the presence of high image noise or missing information.

Spatio-temporal fusion. Other approaches build semantic video segmentation networks using spatio-temporal features. Tran *et al.* (2016) and Zhang *et al.* (2014) use 3D convolutions to compute such features. The *Recurrent Convolutional Neural Network* of Pavel *et al.* (2017) is another spatio-temporal architecture. It uses layer-wise recurrent self-connections as well as top-down connections to stabilize representations. However, it is difficult to integrate physical constraints into such approaches and they require large numbers of parameters.

5.3 Functionally Modularized Temporal Filtering

5.3.1 Model Design

The goal of this work is to improve the robustness of a CNN f using information of previous time-steps. The temporal integration of information is achieved by means of a filter module f_{FM}, which stabilizes an abstract feature representation of the CNN. In comparison to other filters, our filter has a modular structure and contains human interpretable intermediate representations, resulting in an improved interpretability and transparency. Although we focus on segmentation models in the remainder of this chapter, our filter approach can be applied in a variety of models (*e.g.* CNNs for object detection, free space detection, ...).

We assume a semantic segmentation model f is given, which consists of two parts: a convolutional feature encoder f_{enc} and a semantic decoder f_{sem}. The feature encoder operates on an image $\tilde{\mathbf{x}}_t$ and generates an abstract feature representation $\tilde{\mathbf{r}}_t$. This representation is up-sampled and refined by the semantic decoder f_{sem} to produce a dense pixel-wise semantic segmentation $\tilde{\mathbf{s}}_t$:

$$\tilde{\mathbf{s}}_t = f(\tilde{\mathbf{x}}_t; \theta) = f_{sem}(\tilde{\mathbf{r}}_t; \theta_{sem}) = f_{sem}(f_{enc}(\tilde{\mathbf{x}}_t; \theta_{enc}); \theta_{sem}), \tag{5.1}$$

where θ are the learnable parameters of the overall model and θ_{enc} as well as θ_{sem} are the parameters of the encoder and decoder, respectively.

Due to data-inherent perturbations (*e.g.* noise introduced by adverse weather conditions), the representation $\tilde{\mathbf{r}}_t$ is a crude approximation of the true feature representation without perturbations. Using a temporal filter f_{FM}, we improve the estimate of the features $\hat{\mathbf{r}}_t$ and consequently also the segmentation of the decoder:

$$\hat{\mathbf{s}}_t = f_{sem}(\hat{\mathbf{r}}_t; \theta_{sem}) = f_{sem}(f_{FM}(\tilde{\mathbf{r}}_t, \mathbf{h}_{t-1}; \theta_{FM}); \theta_{sem}). \tag{5.2}$$

The hidden state $\mathbf{h}_{t-1}$ represents an internal memory, encoding all the prior knowledge about scene features and dynamics, aggregated from previous time steps. Framing Equation 5.2 in the context of a *Bayes* Filter (Gu *et al.*, 2017), the filter module f_{FM} has to propagate the belief about the hidden state $\mathbf{h}_{t-1}$ one time step into the future, update the belief using the current input $\tilde{\mathbf{r}}_t$ of the filter, and compute an improved estimate $\hat{\mathbf{r}}_t$ of

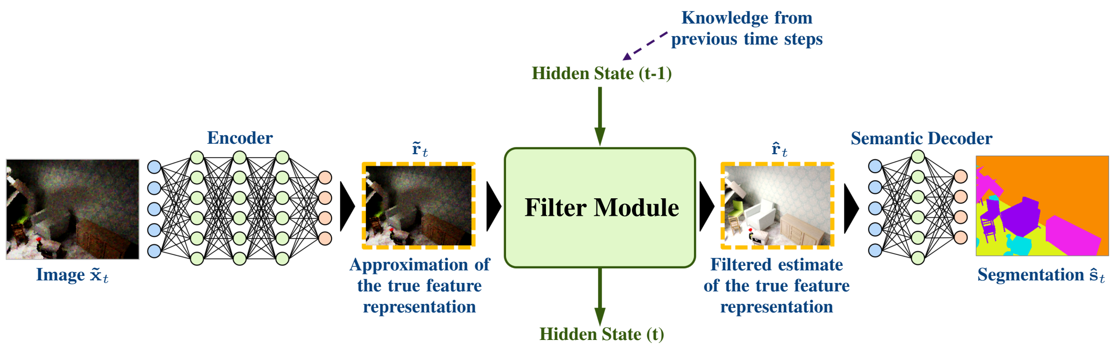

Knowledge from previous time steps
Hidden State (t-1)
Encoder
$\tilde{\mathbf{r}}_t$
Filter Module
$\hat{\mathbf{r}}_t$
Semantic Decoder
Image $\tilde{\mathbf{x}}_t$
Approximation of the true feature representation
Filtered estimate of the true feature representation
Segmentation $\hat{\mathbf{s}}_t$
Hidden State (t)

Figure 5.2: Semantic segmentation model f consisting of an encoder f_{enc} and decoder f_{sem}, extended by a filter module f_{FM}. The filter module uses information of previous time steps to derive an improved estimate $\hat{\mathbf{r}}_t$ of the true feature representation. For illustration purposes, abstract feature representations ($\tilde{\mathbf{r}}_t$ and $\hat{\mathbf{r}}_t$) are displayed as images framed by an orange dashed line. The pictures used to illustrate $\tilde{\mathbf{x}}_t$, $\tilde{\mathbf{r}}_t$, $\hat{\mathbf{r}}_t$, and $\hat{\mathbf{s}}_t$ are altered images of the *SceneNet RGB-D* (McCormac *et al.*, 2017) dataset.

the true feature representation. A visualization of the resulting segmentation model with filter is depicted in Figure 5.2.

A common architecture to implement filter modules for spatial representations are *Recurrent Neural Networks* (RNNs), such as convolutional LSTM cells (Xingjian *et al.*, 2015). Such models are well suited to model sequential data due to their recurrent structure and are usually fully learned from data. However, due to being black-box architectures, they are hard to understand, debug, and validate. Domain knowledge or additional constraints (*e.g*. the consistency of camera motion across spatial dimensions) are also not easy to integrate.

To overcome these drawbacks, we propose a more interpretable and transparent filter module. We adopt the basic structure of a recursive *Bayesian* estimator and split the filter f_{FM} into a prediction module f_{pred} and an update f_{upd} module:

$$\hat{\mathbf{r}}_t = f_{FM}(\tilde{\mathbf{r}}_t, \mathbf{h}_{t-1}; \theta_{FM}) = f_{upd}(\tilde{\mathbf{r}}_t, f_{pred}(\mathbf{h}_{t-1}; \theta_{pred}); \theta_{upd}), \tag{5.3}$$

where θ_{pred} and θ_{upd} are the respective parameters of the two components. The prediction module f_{pred} propagates the hidden state $\mathbf{h}_{t-1}$ one time step into the future, while the update module f_{upd} refines it using information of the current input representation $\tilde{\mathbf{r}}_t$ to obtain an improved estimate of the encoder representation $\hat{\mathbf{r}}_t$. Thus, the prediction module has to learn the complex dynamics of a high-dimensional hidden-state. To increase interpretability and divide the prediction task into easier subtasks, we decouple static scene modeling and dynamic motion modeling. Decoupling is achieved by splitting the hidden state $\mathbf{h}_t$ into a high-dimensional static state $\mathbf{h}_t^s$ encoding all static scene features (*i.e.* an abstract representation of the content and geometry of the scene) and a low-dimensional dynamic state $\mathbf{h}_t^m$ encoding scene dynamics (*e.g.* the motion of the camera). In the proposed implementation, the static state corresponds to the target output of the filter: $\mathbf{h}_t^s = \hat{\mathbf{r}}_t$. Both sub-states are temporally integrated using separate filter modules — a feature filter and a motion filter. The two subfilters as well as their modules and interconnection are depicted in Figure 5.1.

The prediction module of the feature filter only has to account for the spatial displacement of static scene features due to motion. This displacement can be explicitly modeled via a geometric projection f_{proj}. To compute a valid projection of the static state $\mathbf{h}_{t-1}^s$, estimates of the scene geometry and the scene dynamics are required. We encode the scene geometry as a dense depth map $\hat{\mathbf{d}}_{t-1}$ derived from the static state $\mathbf{h}_{t-1}^s$ by means of a depth decoder f_{dep}. A 3D rigid transformation $\hat{\tau}_{t-1}^t$ is used to characterize the scene dynamics, assuming the dynamics are dominated by camera motion. Scene dynamics are separately estimated and filtered in the motion filter. The predicted static state $\bar{\mathbf{h}}_t^s$ is updated in a second module f_{upd}^s using new information of the input representation $\tilde{\mathbf{r}}_t$. All components of the static feature filter are described in more detail in Section 5.3.2.

Scene dynamics are estimated and filtered in a second subfilter, the motion filter. A motion embedding module f_{mot} is used to project the high-dimensional scene feature space into a low-dimensional motion feature space. The transformation is fully learned, enabling the model to generate a representation well-suited for motion integration. The prediction f_{pred}^m and update step f_{upd}^m of the motion filter is implemented using a *Gated Recurrent Unit* (GRU) (Cho *et al.*, 2014). The 3D rigid transformation $\hat{\tau}_{t-1}^t$ is derived from the filtered hidden state $\mathbf{h}_t^m$ by means of a small fully connected network f_{3d}. Section 5.3.3 further elaborates on the different components of the motion filter.

One advantage of decoupling motion and scene features is the easy integration of auxiliary information such as data recorded by additional sensors (*e.g.* acceleration data $\tilde{\mathbf{a}}_{t-1}^t$ of the recording camera). This kind of motion information can be fused much more targeted with the appropriate motion features derived from image pairs. The feature fusion is implemented using a fully-learned fusion module f_{fus}. An additional advantage of the decoupling is a global modeling of camera motion. The motion is guaranteed to be consistent across spatial scene features and can be estimated using correlations across

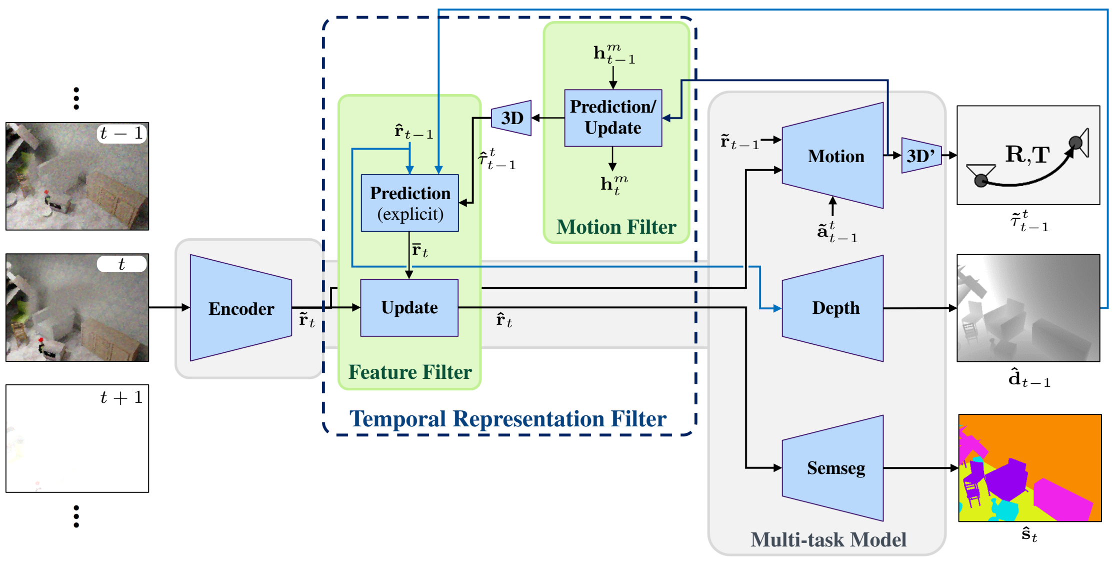


Figure 5.3: Multi-task-based interpretation of the resulting overall model. The model consists of an underlying multi-task architecture which predicts a semantic segmentation, a depth map, and a 3D rigid transformation. The predictions of the multi-task model are temporally stabilized by means of a representation filter. The depth map and the 3D transformation predicted by the multi-task model are used in the temporal representation filter to propagate previous knowledge. Due to the filter design, the model can also make a reasonable prediction if the input image at the current time step t does not contain any meaningful information (*e.g.* the image is black due to a short failure of the camera). The pictures used to illustrate the input images, the depth map, as well as the pixel-wise semantic segmentation are altered images of the *SceneNet RGB-D* (McCormac *et al.*, 2017) dataset.

full image pairs. Filters based on convolutional RNNs (Wagner *et al.*, 2018b) usually do not posses these properties, due to their convolutional processing nature.

The overall architecture (*i.e.* segmentation model plus functionally modularized temporal filter) can also be interpreted as a multi-task model:

$$\begin{aligned} f_{MT}(\cdot;\theta_{MT}) =& f_{enc}(\cdot;\theta_{enc}) \cup f_{sem}(\cdot;\theta_{sem}) \cup f_{dep}(\cdot;\theta_{dep}) \cup \\ & f_{mot}(\cdot;\theta_{mot}) \cup f_{3d}(\cdot;\theta_{3d}), \end{aligned} \tag{5.4}$$

which predicts a segmentation, a depth map, and a 3D rigid transformation (see Figure 5.3). The encoder representation $\tilde{\mathbf{r}}_t$ of the multi-task model is integrated over time using an additional filter module, which utilizes decoder outputs to propagate previous knowledge. As decoders (*i.e.* the depth and semantic segmentation decoder) operate on the filtered encoder representation $\hat{\mathbf{r}}_t$ or utilize a separate filter (see motion decoder), the functionality of our model does not dependent on new input images $\tilde{\mathbf{x}}_t$. Thus, the model can even make a reasonable motion, depth, and semantic prediction if the input image $\tilde{\mathbf{x}}_t$

at the current time step t does not contain any meaningful information (*e.g.* the image is black due to a short failure of the camera). This property sets our filter apart from other approaches in the literature (Gadde *et al.*, 2017; Nilsson and Sminchisescu, 2018). We thus believe that our approach is much better suited for reducing aleatoric failures of single-frame semantic segmentation models.

The overall filter is developed to increase transparency and interpretability, by modularizing functionalities, using explicit physical models as subcomponents, and introducing human interpretable representations. Compared to other existing architectures, it is much easier to debug and validate the model, inspect intermediate results, pre-train subnetworks, and introduce physical constraints. These properties are also particularly relevant with regard to safety analysis. From a multi-task perspective, the two auxiliary tasks (depth and motion estimation) may also benefit segmentation, due to an implicit regularization (Ruder, 2017).

5.3.2 Feature Filter

In this section, we discuss the three components of the feature filter in more detail. Two of the components, the update module and the depth decoder, are fully learned from data. The prediction module, on the other hand, is implemented via a geometric projection and does not contain any learnable parameters. Thus, the feature prediction is performed in a more transparent manner, making it easier to determine the contribution of information from previous time steps.

Prediction / Geometric Projection

The prediction module temporally propagates static scene features encoded in the static state $\mathbf{h}^s_{t-1}$ one time step into the future. By focusing on a static representation, the prediction is simplified to a spatial displacement of features. This displacement is modeled using a geometric projection (Zhou *et al.*, 2017; Mahjourian *et al.*, 2017). Consequently, the resulting prediction module does not contain any learnable parameters and is fully parametrized by the scene geometry and the scene dynamics. The former parameter is provided by the depth decoder and the later by the motion filter (see Section 5.3.3). The idea of the proposed prediction approach is to use knowledge about the physical world as well as the imaging process to increase interpretability and transparency. This enables an easier integration of geometric as well as physical constraints (*e.g.* motion constraints) and usually results in a reduced number of filter parameters.

Let $\mathbf{p}^*_{t-1} = (i,\, j,\, 1)^T$ be the homogeneous coordinate of a pixel in the static state $\mathbf{h}^s_{t-1}$ at time step $t-1$ and $\hat{d}^{ij}_{t-1}$ the associated depth value. Using the camera intrinsic matrix $\mathbf{K}$ one can compute the corresponding 3D point in camera coordinates:

$$\mathbf{P}_{t-1} = \hat{d}^{(i,j)}_{t-1} \cdot \mathbf{K}^{-1} \cdot \mathbf{p}^*_{t-1}. \tag{5.5}$$

This 3D point is transformed into the camera coordinate system at time step t using the camera motion encoded in the transformation matrix $\hat{\tau}^t_{t-1}$. The resulting 3D point is projected back to the pixel coordinate system to get the homogeneous coordinate $\mathbf{p}^*_t$ at time step t:

$$\mathbf{p}^*_t \sim \mathbf{K} \cdot \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right) \cdot \hat{\tau}^t_{t-1} \cdot \mathbf{P}^*_{t-1}. \tag{5.6}$$

We use the asterisk superscript to mark homogeneous coordinates, *e.g.* $\mathbf{P}^*_{t-1}$ is a homogeneous version of $\mathbf{P}_{t-1}$. The coordinate $\mathbf{p}_t$ is continuous and has to be discretized in a subsequent processing step. Additionally, it is necessary to account for ambiguities, in cases where multiple pixels of time step $t-1$ are assigned to the same pixel of time step t. We resolve these ambiguities by choosing the transformed pixel with the smallest depth (*i.e.* we use objects closer to the sensor). Using Equation 5.5 and 5.6 as well as the proposed post-processing steps, one can temporally propagate the static state $\mathbf{h}^s_{t-1}$ one time step into the future. The full projection is differentiable with respect to scene features. In contrast to other explicit prediction methods, our implementation does not depend on information of time step t. This is an important property for resolving failures due to missing or incomplete information available in the current input frame.

Depth Estimation and Supervision

The scene geometry is parametrized by a dense depth map $\hat{\mathbf{d}}_{t-1}$ (*i.e.* one depth value $\hat{d}^{ij}_{t-1}$ per pixel) and estimated using a decoder network f_{dep} operating on the filtered static state representation $\mathbf{h}^s_{t-1}$. By operating on the representation $\mathbf{h}^s_{t-1}$ of the previous time

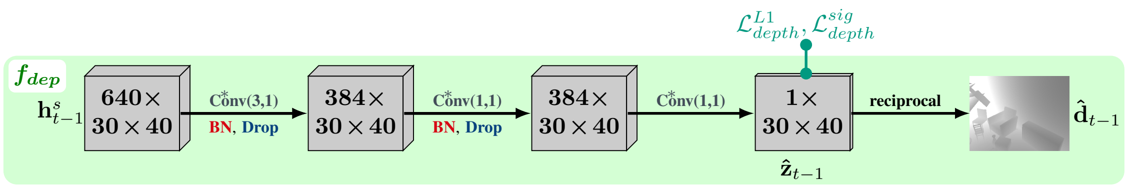


Figure 5.4: Depth decoder f_{dep} consisting of three convolutional layers. The depth decoder estimates a dense depth map $\hat{\mathbf{d}}_{t-1}$ using the information provided by the filtered representation $\mathbf{h}^s_{t-1}$. **$\overset{*}{\text{C}}$onv(f,s)**: convolutional layer with **f** filters, a stride of **s**, and a ReLU nonlinearity. The first two layers additionally apply batch normalization (**BN**) and dropout (**Drop**). The inverse depth $\hat{\mathbf{z}}_{t-1}$ is used to compute the depth decoder specific loss terms. The depiction of the depth map $\hat{\mathbf{d}}_{t-1}$ is taken from the *SceneNet RGB-D* (McCormac *et al.*, 2017) dataset.

step it is possible to get meaningful depth estimates even when the current image $\tilde{\mathbf{x}}_t$ at time step t does not contain any meaningful information. The depth map is used in the prediction module to compute the geometric projection of the representation $\mathbf{h}^s_{t-1}$.

The depth decoder consists of three convolutional layers with kernel size 3×3, 1×1, and 1×1, respectively. We apply batch normalization (Ioffe and Szegedy, 2015) and dropout (Srivastava *et al.*, 2014) in the first two layers and use ReLU nonlinearities (Nair and Hinton, 2010) in each layer. Therefore, the predicted depth is always positive and valid. In the first two layers the number of features is set to 384 and the last layer predicts one value per pixel. Instead of directly predicting depth values, we estimate the inverse depth $\hat{\mathbf{z}}_{t-1}$, which puts less focus on wrong predictions in larger distance. In a last step, the inverse depth is explicitly converted to the depth map $\hat{\mathbf{d}}_{t-1} = 1/\hat{\mathbf{z}}_{t-1}$. The overall architecture of the depth decoder is depicted in Figure 5.4.

We use two depth decoder specific loss terms during training. The first loss term $\mathcal{L}^{L1}_{depth}$ is a $L1$ loss on the inverse depth:

$$\mathcal{L}^{L1}_{depth} = \sum_{i,j} |z^{ij}_{t-1} - \hat{z}^{ij}_{t-1}|, \tag{5.7}$$

where $\hat{z}^{ij}_{t-1}$ is the predicted inverse depth at pixel location (i, j) and z^{ij}_{t-1} the ground truth inverse depth. The second loss term $\mathcal{L}^{sig}_{depth}$ is a squared $L2$ loss on the scale invariant gradient $\mathbf{g}^{ij}_h[\cdot]$ computed for three spatial scales $h \in \{1, 2, 4\}$ (Ummenhofer *et al.*, 2017):

$$\mathcal{L}^{sig}_{depth} = \sum_{h\in\{1,2,4\}} \sum_{i,j} \left\| \mathbf{g}^{ij}_h[\mathbf{z}_{t-1}] - \mathbf{g}^{ij}_h[\hat{\mathbf{z}}_{t-1}] \right\|^2_2 . \tag{5.8}$$

This loss penalizes incorrect depth changes in small local neighborhoods. The scale invariant gradient of the ground truth depth and the predicted depth is defined by (Ummenhofer *et al.*, 2017):

$$\mathbf{g}_h^{ij}[\mathbf{n}] = \left(\frac{n^{i+h,j} - n^{ij}}{|n^{i+h,j}| + |n^{ij}|}, \frac{n^{i,j+h} - n^{ij}}{|n^{i,j+h}| + |n^{ij}|} \right)^T . \tag{5.9}$$

Update / Feature Fusion

The update module linearly combines previous knowledge encoded in the predicted static state $\bar{\mathbf{h}}_t^s$ with new information of the current input representation $\tilde{\mathbf{r}}_t$. The weights of the linear combination are calculated in a data-dependent manner to obtain an optimal fusion of the two representations. Weighting is conducted on a per-pixel basis using information of a local neighborhood. For each pixel position a weighting value i_t^{ij} is estimated that indicates whether one should rely on prior knowledge encoded in $\bar{\mathbf{h}}_t^s$ or on information of the new input $\tilde{\mathbf{r}}_t$. This weight matrix $\mathbf{i}_t$ (also referred to as update gate) is calculated similarly to convolution LSTM gates (Xingjian *et al.*, 2015), but contains only one value per pixel instead of one value per pixel and feature:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{hi} * \bar{\mathbf{h}}_t^s + \mathbf{W}_{ri} * \tilde{\mathbf{r}}_t + \mathbf{b}_i). \tag{5.10}$$

The convolutional operator is indicated by $*$, $\mathbf{W}_{hi}$ and $\mathbf{W}_{ri}$ are 3×3 kernels, and $\mathbf{b}_i$ is a bias term. Using $\mathbf{i}_t$ and element-wise multiplications represented by $\circ$, the update module computes:

$$\mathbf{h}_t^s = \hat{\mathbf{r}}_t = (1 - \mathbf{i}_t) \circ \bar{\mathbf{h}}_t^s + \mathbf{i}_t \circ \tilde{\mathbf{r}}_t. \tag{5.11}$$

Note that weighting values are broadcasted along the feature dimension during multiplication. The initial predicted state $\bar{\mathbf{h}}_0^s$ used in the first update step is a parameter of the update module and learned during training.

5.3.3 Motion Filter

The scene dynamics are estimated and temporally integrated using the motion filter. Within this work, we assume that scene dynamics are dominated by camera motion.

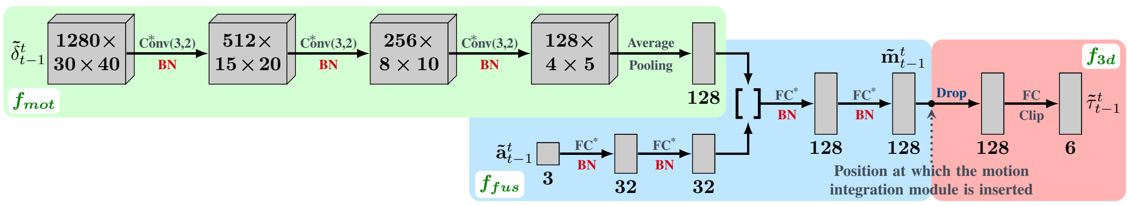


Figure 5.5: Motion estimation module (motion decoder) consisting of a motion embedding subnetwork f_{mot}, a fusion subnetwork f_{fus}, and a subnetwork f_{3d} to infer the camera motion $\tilde{\tau}^t_{t-1}$. The module uses a pair of encoder representations $\tilde{\delta}^t_{t-1} = [\tilde{\mathbf{r}}_{t-1}, \tilde{\mathbf{r}}_t]$ and the acceleration data $\tilde{\mathbf{a}}^t_{t-1}$ of the camera as input. **Conv*(f,s)**: convolutional layer with **f** filters, a stride of **s**, and a ReLU nonlinearity. **FC*** / **FC**: fully connected layers, an asterisk superscript indicates the presence of a ReLU nonlinearity. The symbol [ ] indicates the concatenation of feature representations. All layers with learnable parameters, except the last fully connected layer, use batch normalization (**BN**). We additionally apply dropout (**Drop**) to one representation and clip three entries of the final activation (the angle estimates) to $[-1, 1]$, indicated by **Clip**.

Thus, the motion filter has to estimate the rotation and translation of the camera, which we parametrize by a 3D rigid transformation $\hat{\tau}^t_{t-1}$. In comparison to the feature filter, all components of the motion filter are fully learned from data. In principle, it would also be possible to exchange subcomponents of the motion filter by physical models.

Motion Estimation

Using the motion embedding network f_{mot}, the motion estimation module learns a projection from the high-dimensional scene feature space to the low-dimensional motion feature space. To stabilize motion estimates, the projected features are combined in the fusion module f_{fus} with features derived from the acceleration data $\tilde{\mathbf{a}}^t_{t-1}$ of the camera. The fused representation is propagated through a small fully connected network f_{3d} to infer the 3D rigid camera transformation $\tilde{\tau}^t_{t-1}$. Figure 5.5 visualizes the overall structure of the motion estimation module.

Pairs of encoder representations $\tilde{\delta}^t_{t-1} = [\tilde{\mathbf{r}}_{t-1}, \tilde{\mathbf{r}}_t]$ concatenated along the feature dimension are used as the input of the motion embedding network f_{mot}. The motion embedding network is a CNN consisting of three convolutional layers with kernel size 3×3 and a stride of 2. The embedding of the CNN is processed by an average pooling layer to aggregate motion features at all spatial locations. The motion network can be relatively small as it does not use raw inputs, but abstract representations generated by the encoder network f_{enc}. The aggregated feature vector is concatenated with the motion features derived from the acceleration data $\tilde{\mathbf{a}}^t_{t-1}$. To compute the acceleration features we use a small network consisting of two fully connected layers. The concatenated features are further transformed by two fully connected layers to create the motion embedding $\tilde{\mathbf{m}}_{t-1}^{t}$. We apply batch normalization and ReLU nonlinearities in each layer with learnable parameters. The shape of all activations is specified in Figure 5.5.

To infer the 3D rigid camera transformation $\tilde{\mathcal{T}}_{t-1}^{t}$ from the motion embedding $\tilde{\mathbf{m}}_{t-1}^{t}$, we use a linear layer with 6 units. This layer predicts the translation vector $\tilde{\mathbf{T}}_{t-1}^{t}$ and the sinus of the rotation angles $(\sin \alpha_{t-1}^{t}, \sin \beta_{t-1}^{t}, \sin \gamma_{t-1}^{t})^{\top}$. The angle sinus estimates are clipped to $[-1, 1]$ and converted to a rotation matrix $\tilde{\mathbf{R}}_{t-1}^{t}$ (Vijayanarasimhan *et al.*, 2017; Zhou *et al.*, 2017). The rigid camera transformation is obtained by:

$$\tilde{\mathcal{T}}_{t-1}^{t} = \begin{bmatrix} \tilde{\mathbf{R}}_{t-1}^{t} & \tilde{\mathbf{T}}_{t-1}^{t} \\ \mathbf{0}^{T} & 1 \end{bmatrix}. \tag{5.12}$$

Temporal Motion Integration and Supervision

The prediction of the motion estimation module only depends on the information of two consecutive time steps. This makes the motion estimation quite susceptible to incomplete or missing input data. In order to always obtain a meaningful motion estimate, we integrate motion features over time in a fully-learned, model-free filter. Filtering may also lead to a general improvement in motion estimation.

The filter consists of a GRU (Cho *et al.*, 2014) followed by a fully connected layer and is defined by:

$$\mathbf{o}_{t-1}^{t} = \sigma(\mathbf{W}_{\tilde{m}o}\tilde{\mathbf{m}}_{t-1}^{t} + \mathbf{W}_{ho}\mathbf{h}_{t-1}^{m} + \mathbf{b}_{o}), \tag{5.13}$$

$$\mathbf{u}_{t-1}^{t} = \sigma(\mathbf{W}_{\tilde{m}u}\tilde{\mathbf{m}}_{t-1}^{t} + \mathbf{W}_{hu}\mathbf{h}_{t-1}^{m} + \mathbf{b}_{u}), \tag{5.14}$$

$$\mathbf{c}_{t-1}^{t} = \tanh(\mathbf{W}_{\tilde{m}c}\tilde{\mathbf{m}}_{t-1}^{t} + \mathbf{o}_{t-1}^{t} \circ (\mathbf{W}_{hc}\mathbf{h}_{t-1}^{m}) + \mathbf{b}_{c}), \tag{5.15}$$

$$\mathbf{h}_{t}^{m} = (1 - \mathbf{u}_{t-1}^{t}) \circ \mathbf{h}_{t-1}^{m} + \mathbf{u}_{t-1}^{t} \circ \mathbf{c}_{t-1}^{t} \tag{5.16}$$

$$\hat{\mathbf{m}}_{t-1}^{t} = \mathrm{relu}(\mathrm{bn}(\mathbf{W}_{out}\mathbf{h}_{t}^{m} + \mathbf{b}_{out})). \tag{5.17}$$

It is inserted after the fusion module (see marked location in Figure 5.5) and stabilizes the motion embedding $\tilde{\mathbf{m}}_{t-1}^{t}$. The initial state vector $\mathbf{h}_{0}^{m}$ of the GRU is a parameter of the model and leaned during training. The output vector $\mathbf{h}_{t}^{m}$ consisting of 256 features is transformed by the fully connected layer to a feature vector with the same size as the input embedding $\tilde{\mathbf{m}}_{t-1}^{t}$. The function bn in Equation 5.17 indicates the usage of batch normalization. Using the filtered motion embedding $\hat{\mathbf{m}}_{t-1}^{t}$ as well as the network f_{3d}, one can compute the stabilized rigid camera transformation $\hat{\tau}_{t-1}^{t}$ (see Equation 5.12).

Two motion specific loss terms are used during training. The losses are based on the relative transformation between the predicted and ground-truth motion as defined by Vijayanarasimhan *et al.* (2017):

$$\mathcal{L}_{motion}^{trans} = (\Delta T)^2 = \left\| \mathrm{inv}\left(\hat{\mathbf{R}}_{t-1}^{t}\right)\left(\mathbf{T}_{t-1}^{t} - \hat{\mathbf{T}}_{t-1}^{t}\right)\right\|_2^2, \tag{5.18}$$

$$\begin{aligned}\mathcal{L}_{motion}^{rot} &= \Delta R \\ &= \arccos\left(\min\left(1, \max\left(-1, \frac{\mathrm{trace}(\mathrm{inv}\left(\hat{\mathbf{R}}_{t-1}^{t}\right)\mathbf{R}_{t-1}^{t}) - 1}{2}\right)\right)\right),\end{aligned} \tag{5.19}$$

where $\mathbf{T}_{t-1}^{t}$ is the ground-truth camera translation vector and $\mathbf{R}_{t-1}^{t}$ the ground-truth rotation matrix. $\hat{\mathbf{T}}_{t-1}^{t}$ and $\hat{\mathbf{R}}_{t-1}^{t}$ are the corresponding predicted translation vector and rotation matrix.

5.4 Implementation Details

5.4.1 Dataset

We evaluate our proposed modular temporal filter using the *SceneNet RGB-D* (McCormac *et al.*, 2017) dataset which consists of 5M photorealistically rendered RGB-D images recorded from 15k indoor trajectories. The scene layout, lighting conditions, camera trajectories, and textures are randomly sampled for each sequence, while considering physical constraints. Besides the camera motion, all scenes are static. Due to its simulated nature, the dataset provides labels for semantic segmentation, depth estimation, and camera motion estimation. We split the training data into a training and validation set and use the provided validation data as the test set. For training, we use all non-overlapping sequences of length 7 generated from the training trajectories. The test set is constructed by sampling 5 non-overlapping sequences of length 7 from each test trajectory, resulting in 5,000 test sequences. Unless otherwise specified, we use this test set in our evaluation.

To add aleatoric uncertainty to the input data, all sequences are perturbed with noise, clutter, and changes in lighting conditions. In the following, we give a detailed description of the process used to generate these perturbations. After applying the perturbations to a clean sequence $\mathbf{x}_s$ generated from the *SceneNet RGB-D* dataset, we clip pixel values to the interval $[0, 1]$ to get a valid perturbed sequence $\tilde{\mathbf{x}}_s$. Example sequences are shown in Figure 5.6.

Clutter is introduced by setting subregions of each image to the pixel mean μ_s, computed on a per-sequence basis. The clutter is generated once per-sequence and applied to each frame. Thus, the resulting clutter pattern is the same in each frame, comparable to dirt on the camera lens. The perturbed image $\mathbf{x}'_{sj}$ is calculated by:

$$\mathbf{x}'_{sj} = \mathbf{x}_{sj} \cdot (1 - \mathbf{m}_s) + \mu_s \cdot \mathbf{m}_s, \tag{5.20}$$

where $\mathbf{m}_s$ is a per-sequence clutter mask, μ_s the per-sequence pixel mean, and $\mathbf{x}_{sj}$ the clean image (the subscript j indicates the frame number and the index s the sequence index). The clutter mask is generated by summing N_s *Gaussian* kernels whose centers are randomly placed (uniformly sampled) within the image dimensions. Each kernel is normalized to the maximum value one. The number of kernels N_s is uniformly sampled from the interval $[0, 8]$ for each sequence. In addition, we uniformly sample the standard deviations of each 2D-kernel from the interval $[10, 36]$. The kernels are truncated at three times the standard deviation. In a final step, the resulting clutter mask is clipped to the range $[-1, 1]$.

Rapid changes in lighting conditions are simulated by increasing or decreasing the intensity of frames by a random value and letting this offset decay over time. For each sequence, we uniformly sample one frame index i_s and a multiplier p_s which with a probability of 0.5 is either 1 or -1. In addition, we draw a scaling factor c_s from the interval $[0.5, 1.0]$. The perturbed images $\mathbf{x}''_{sj}$ are calculated by:

$$\forall j < i_s : \mathbf{x}''_{sj} = \mathbf{x}'_{sj}, \tag{5.21}$$

$$\forall j \geq i_s : \mathbf{x}''_{sj} = \mathbf{x}'_{sj} + p_s \cdot 0.3^{j-i_s} \cdot c_s. \tag{5.22}$$

Such a perturbation pattern occurs in real life, for example, when the light is suddenly switched off in a room.

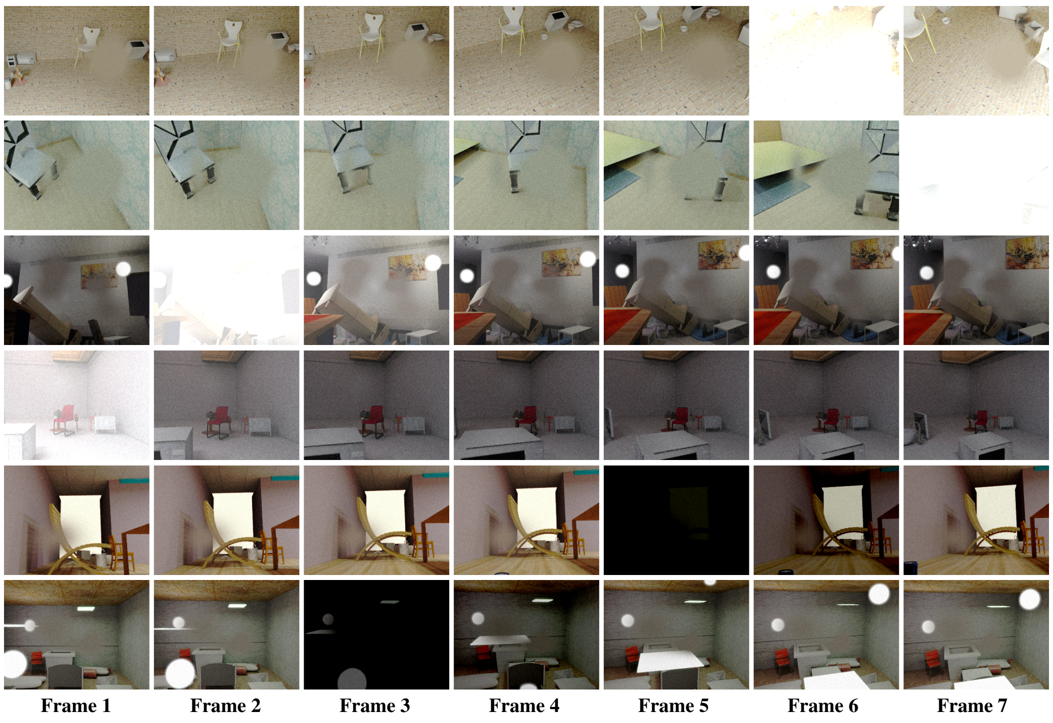


Figure 5.6: Example sequences of the training dataset (McCormac *et al.*, 2017). One sequence of length 7 is shown per row. Each sequence is perturbed with randomly sampled noise, clutter, and changes in lighting conditions.

Noise is simulated by adding independent Gaussian noise with zero mean to each pixel of a sequence. The variance of the noise is independently sampled for each sequence from the interval $[0, 0.001]$.

By utilizing simulated perturbations, we can focus on aleatoric failures, which are often underrepresented in real-world benchmarks. The usage of a large-scale, fully-labeled simulated dataset additionally reduces the amount of dataset-dependent epistemic failures. The generated dataset is thus well suited to investigate the benefits of temporal integration.

5.4.2 Segmentation Models

All architectures used in this work are based on the *Pyramid Scene Parsing Network* (PSPNet) of Zhao *et al.* (2017). This model uses a dilated *Fully Convolutional Network* (Chen *et al.*, 2015; Yu and Koltun, 2016) supplemented by a *Pyramid Pooling Module* to produce a per-pixel semantic prediction.

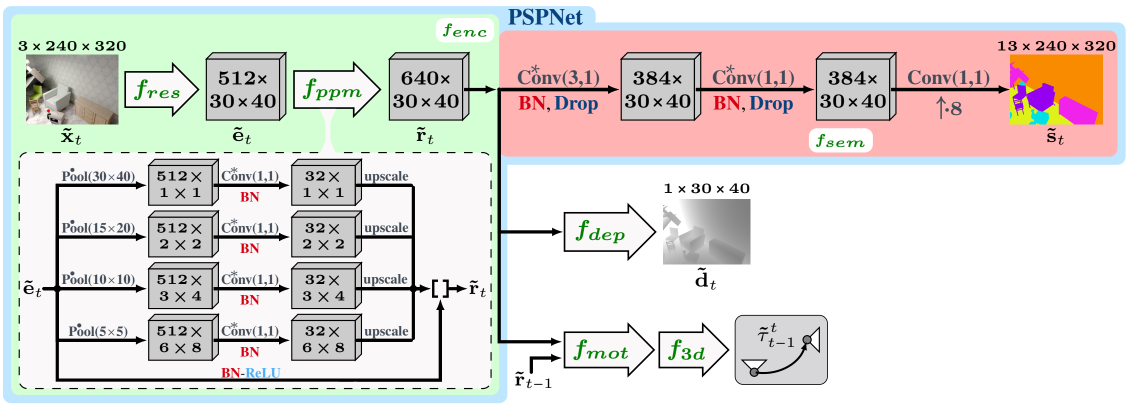


Figure 5.7: *Multi-Task Pyramid Scene Parsing Network* (MPSPNet) consisting of a feature encoder f_{enc} and three decoders (f_{sem}: semantic decoder; f_{dep}: depth decoder; $f_{mot} + f_{3d}$: motion decoder). The encoder uses a dilated ResNet f_{res} and a *Pyramid Pooling Module* f_{ppm}. The MPSPNet corresponds to a single-task *Pyramid Scene Parsing Network* (PSPNet) (highlighted in blue), extended by a motion and depth decoder. $\overset{*}{\text{Conv}}$**(f,s)**: convolutional layer with **f** filters and a stride of **s**; $\overset{\bullet}{\text{Pool}}$**(k)**: Average pooling layer without zero padding using a kernel size and stride of **k**. The use of batch normalization and dropout is indicated by **BN** and **Drop**, respectively. To obtain a semantic segmentation $\tilde{s}_t$ with the same resolution as the input image $\tilde{x}_t$, we use an eightfold bilinear upsampling (**↑·8**) as well as a pixel-wise softmax function in the last layer of the semantic decoder. The input image, the depth map, as well as the pixel-wise semantic segmentation are taken from the *SceneNet RGB-D* (McCormac *et al.*, 2017) dataset.

Our implementation of the PSPNet, consisting of a feature encoder f_{enc} and a semantic decoder f_{sem}, is shown in Figure 5.7. Within the feature encoder we employ a dilated *Residual Network* (ResNet) (He *et al.*, 2016a) to extract a representation $\tilde{e}_t$ with a fairly large spatial resolution while avoiding a reduction of the receptive field of the neurons. To improve information flow, we utilize *pre-activations* in all residual blocks (*i.e.* the order of operations is normalization-nonlinearity-convolution) as proposed by He *et al.* (2016b). The features of the ResNet are further supplemented by global context information aggregated in a *Pyramid Pooling Module* (Zhao *et al.*, 2017). This module average pools the representation $\tilde{e}_t$ on four spatial scales, embeds the resulting representations into a lower dimensional feature space, and upscales them to the original spatial resolution using bilinear interpolation. The resulting representations are concatenated with the ResNet representation $\tilde{e}_t$ to produce the output $\tilde{r}_t$ of the encoder. The structure of the dilated ResNet is depicted in Figure 5.8 and the *Pyramid Pooling Module* is shown in the grey box of Figure 5.7. The weights of the feature encoder f_{enc} are pre-trained using the *ImageNet* dataset. The PSPNet is comparatively small to keep the computational effort and the required memory of the filtered models manageable.

The semantic decoder f_{sem} (Figure 5.7, highlighted in red) has a structure similar to the depth decoder, consisting of three convolutional layers with kernel sizes 3×3, 1×1,

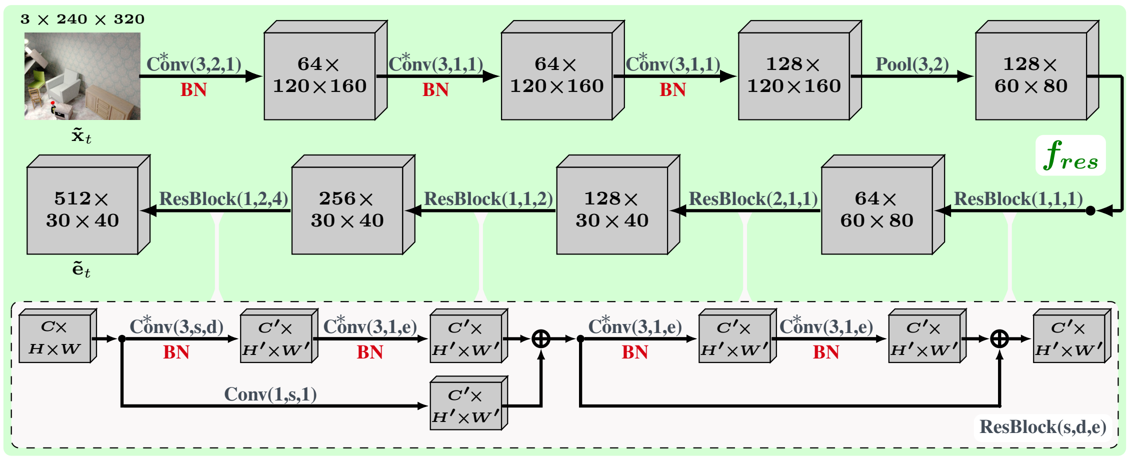


Figure 5.8: Dilated Residual Network. **Conv(f,s,d)**: dilated convolutional layer with **f** filters, a stride of **s**, and a dilation rate of **d**. An asterisk superscript indicates the presence of a ReLU nonlinearity. **Pool(f,s)**: Max-pooling operator using a stride of **s** and pooling regions of length **f**. The residual blocks (**ResBlock**, highlighted in light grey) use *pre-activations*, meaning the order of operations is normalization-nonlinearity-convolution. To normalize representations, we use batch normalization, indicated by **BN**. The input image $\tilde{\mathbf{x}}_t$ is taken from the *SceneNet RGB-D* (McCormac *et al.*, 2017) dataset.

and 1×1, respectively. We apply batch normalization and dropout in the first two layers and use an eightfold bilinear upsampling as well as a pixel-wise softmax function in the last convolution layer. Thus, the prediction $\tilde{\mathbf{s}}_t$ of the semantic decoder has the same spatial resolution as the input image. We use one segmentation specific loss term during training — the pixel-wise cross-entropy loss:

$$\mathcal{L}_{sem} = -\sum_{i,j}\sum_{c}\log(\tilde{s}_t^{cij})\cdot s_t^{cij}, \tag{5.23}$$

where $\tilde{s}_t^{cij}$ is the predicted softmax score of pixel (i,j) for class c and s_t^{cij} the ground-truth score.

Unfiltered Baseline Model

In our evaluation, an unfiltered segmentation model serves as a reference architecture. To make the comparison of the reference with our filtered model as fair as possible, we use a multi-task version of the PSPNet as the unfiltered baseline. This model, which will be called *Multi-Task Pyramid Scene Parsing Network* (MPSPNet) in the remainder of this chapter, extends the PSPNet by a motion and depth decoder. Thus, during training,

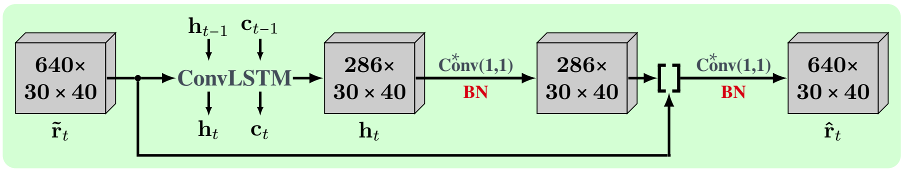


Figure 5.9: Fully-learned, feature-level filter using a convolutional LSTM (**ConvLSTM**) to create spatio-temporal features. $\overset{*}{\textbf{Conv}}$**(f,s)**: convolutional layer with **f** filters, a stride of **s**, and a ReLU nonlinearity. We use batch normalization **BN** in both convolutional layers.

the baseline can also benefit from the additional depth and motion specific loss terms (see Section 5.3.2 and 5.3.3) used to train our filtered architecture. Such a multi-task learning setting (Ruder, 2017) can increase the segmentation performance as shown by Kendall *et al.* (2018). The full architecture of the MPSPNet is depicted in Figure 5.7. It uses the depth decoder introduced in Section 5.3.2 and the motion estimation module of Section 5.3.3. Although the MPSPNet operates on image pairs, its semantic prediction $\tilde{\mathbf{s}}_t$ depends only on the current image $\tilde{\mathbf{x}}_t$.

Model with Modular Filter

Building upon the MPSPNet, we set-up a segmentation model which uses our functionally modularized filter concept introduced in Section 5.3. The filter is applied to the output $\tilde{\mathbf{r}}_t$ of the encoder and stabilizes the representation using information of previous time steps. Please refer to Figure 5.3 for a detailed depiction of the filter integration. Since the representation $\tilde{\mathbf{r}}_t$ has a lower resolution than the input image, the camera intrinsic matrix $\mathbf{K}$ used in the feature filter has to be adjusted to be consistent with the representation downsampling. The segmentation model, which uses our proposed filter, will be called FMTNet hereinafter.

Filtered Baseline Model

Using a model-free, feature-level filter (Fayyaz *et al.*, 2016; Valipour *et al.*, 2017), we create an additional temporally filtered baseline. Such a filter is well suited to solve aleatoric failures, as it does not necessarily require information of the current frame. We use a filter module similar to the one introduced in Wagner *et al.* (2018b). This filter module (see Figure 5.9) operates on the encoder representation $\tilde{\mathbf{r}}_t$ of the MPSPNet and generates an improved estimate of the representation $\hat{\mathbf{r}}_t$. The base component of the filter is a convolutional LSTM (Xingjian *et al.*, 2015), which receives the representation $\tilde{\mathbf{r}}_t$ of the current time step as well as the hidden $\mathbf{h}_{t-1}$ and cell state $\mathbf{c}_{t-1}$ of the previous time step to produce a spatio-temporal embedding $\mathbf{h}_t$:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{ri} * \tilde{\mathbf{r}}_t + \mathbf{W}_{hi} * \mathbf{h}_{t-1} + \mathbf{b}_i\right), \tag{5.24}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{rf} * \tilde{\mathbf{r}}_t + \mathbf{W}_{hf} * \mathbf{h}_{t-1} + \mathbf{b}_f\right), \tag{5.25}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{ro} * \tilde{\mathbf{r}}_t + \mathbf{W}_{ho} * \mathbf{h}_{t-1} + \mathbf{b}_o\right), \tag{5.26}$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbf{W}_{rc} * \tilde{\mathbf{r}}_t + \mathbf{W}_{hc} * \mathbf{h}_{t-1} + \mathbf{b}_c), \tag{5.27}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t). \tag{5.28}$$

All convolutions indicated by the symbol $*$ use weight tensors $(\mathbf{W}_{r*}, \mathbf{W}_{h*})$ with kernel size 3×3. The initial hidden $\mathbf{h}_0$ and cell state $\mathbf{c}_0$ of the convolutional LSTM are parameters of the model and learned during training. The spatio-temporal embedding $\mathbf{h}_t$ is transformed by a convolutional layer and concatenated with the unfiltered representation $\tilde{\mathbf{r}}_t$. Using a second convolutional layer the filter computes the output representation $\hat{\mathbf{r}}_t$. We also evaluated larger kernel sizes and different feature counts in the convolutional LSTM path, but could not achieve further performance improvements.

By inserting this filter in the MPSPNet after the feature encoder f_{enc}, we create a baseline segmentation model utilizing information of previous time-steps and taking advantage of all the benefits of multi-task learning (Ruder, 2017). In the following, we will refer to this model as GFNet. To be comparable with respect to filter complexity, we set the number of parameters in the fully-learned filter to match the number of parameters in our modularized filter. In the case of our filter, we count the parameters of the depth and motion decoder to the filter, since these decoders are required for filtering. The use of all three decoders in the GFNet model guarantees comparable training signals, but is not necessary to create a semantic prediction. Hence, we do not assign the depth and motion decoder weights to the filter, resulting in GFNet having 1.4 times the parameters of FMTNet.

5.4.3 Training Procedure

All segmentation models MPSPNet, FMTNet, and GFNet are trained using the multi-task loss introduced by Kendall *et al.* (2018), which learns the optimal weighting between the cross-entropy segmentation loss, the two depth losses, as well as the two motion losses:

$$\begin{aligned}\mathcal{L} = &\frac{1}{2\sigma_{d_1}^2} \cdot \mathcal{L}_{depth}^{L1} + \log \sigma_{d_1}^2 + \frac{1}{2\sigma_{d_2}^2} \cdot \mathcal{L}_{depth}^{sig} + \log \sigma_{d_2}^2 + \\ &\frac{1}{2\sigma_{m_1}^2} \cdot \mathcal{L}_{motion}^{trans} + \log \sigma_{m_1}^2 + \frac{1}{2\sigma_{m_2}^2} \cdot \mathcal{L}_{motion}^{rot} + \log \sigma_{m_2}^2 + \\ &\frac{1}{\sigma_s^2} \cdot \mathcal{L}_{sem} + \log \sigma_s^2. \end{aligned} \tag{5.29}$$

During training, we minimize this objective with respect to the model parameters as well as the noise parameters $\{\sigma_s, \sigma_{d_1}, \sigma_{d_2}, \sigma_{m_1}, \sigma_{m_2}\}$. We use the Adam optimizer (Kingma and Ba, 2015) with a weight decay of 0.0001 and apply dropout with a probability of 0.1 in the three decoders. All components of FMTNet and GFNet that do not belong to the filter are initialized with the corresponding weights of the trained MPSPNet. Due to its modularity, we can additionally pre-train two components of our proposed filter. First, we pre-train all weights of the motion filter and fine-tune the motion decoder, while keeping the encoder weights fixed. Second, we pre-train the weights of the feature update module as well as the encoder, while keeping all decoders fixed. The second training is performed with sequences containing the same image in each frame, perturbed with random noise patterns. Finally, we fine-tune the overall architecture.

5.5 Evaluation

In this section, we highlight the advantages of using an interpretable filter module and assess the performance gains in semantic segmentation introduced by a temporal representation filter. To evaluate the segmentation performance, we use the mean IoU score on test sequences (see Section 5.4.1), computed over the 13 semantic classes of the *SceneNet RGB-D* (McCormac *et al.*, 2017) dataset. Table 5.1 contains a list of the semantic classes with the corresponding label color.

bed	books	ceiling	chair	floor	furniture	objects
🟦	🟥	🟩	🟪	🟨	⬜	🟦
painting	sofa	table	tv	wall	window	
🟦	🟫	🟪	🟩	🟧	⬜	

Table 5.1: Semantic classes of the *SceneNet-RGBD* dataset with the corresponding color coding that is used in the figures of the evaluation.

Due to the modular design of our filter and the introduction of human interpretable intermediate representations, it is possible to debug and validate individual components of the model. In Section 5.5.1 and Section 5.5.2, we utilize these properties and evaluate the update module of the feature filter as well as the motion filter on toy-like data. To show the advantage of using temporal information, Section 5.5.3 compares the performance of a single-frame semantic segmentation model (MPSPNet) with a version of the model that additionally uses our proposed filter. In addition, we also compare against a segmentation model using a model-free, feature-level filter.

5.5.1 Static Feature Integration

To validate the functionality of the feature update module, we use a separate static toy-dataset with sequences of length four. Each frame of these sequences contains the same clean image, 50% of which is replaced by *Gaussian* noise. The clean images are taken from the *SceneNet RGB-D* dataset and are not altered with any of the aleatoric perturbations introduced in Section 5.4.1. We create a training dataset containing 2 million sequences and a test dataset of 300,000 sequences.

We use the MPSPNet pre-trained on clean *SceneNet RGB-D* data and extend it by our proposed feature filter. Due to the static nature of the sequences, we use the identity transformation (static camera) in the filter. The encoder network f_{enc} and feature update module f^s_{upd} of the resulting model are fine-tuned using the introduced static toy-data.

In Figure 5.10, we depict the semantic prediction of the resulting model and the update gate $\mathbf{i}_t$ (*i.e.* the data-dependent weighting matrix) of its feature filter for one input sequence. A white gating pixel means that the filter uses new information from the current time step. A black pixel means that the filter relies on information of previous time steps.

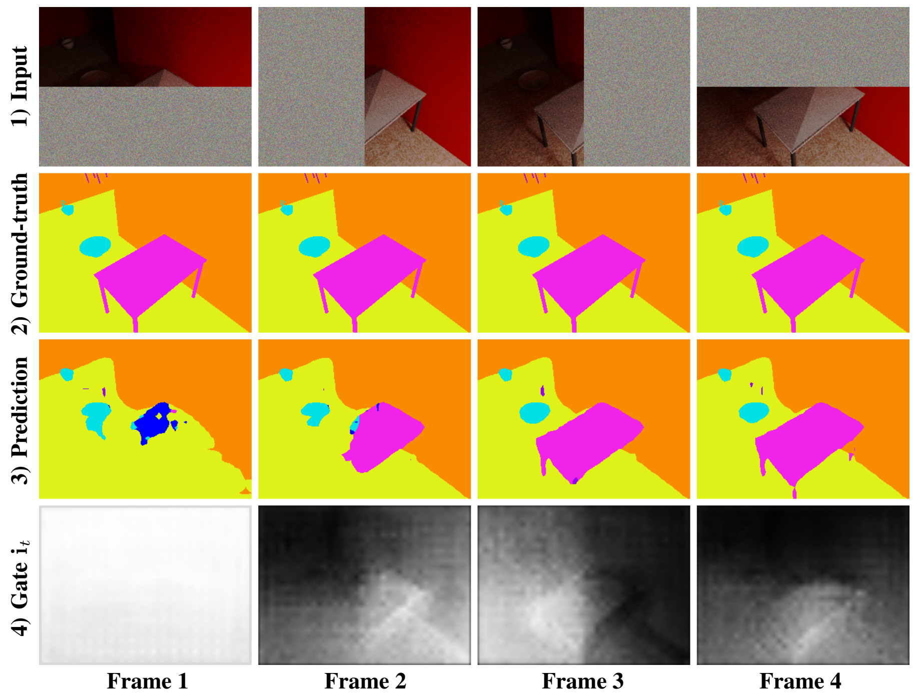


Figure 5.10: Predictions of the MPSPNet with feature filter applied to a static sequence. The images of the sequence are partially occluded by *Gaussian* noise. 1) Input sequence (McCormac *et al.*, 2017) with only 50% of the image visible in each frame. 2) Ground-truth semantic segmentation. 3) Semantic prediction of the MPSPNet with feature filter. 4) Visual representation of the weight matrix $\mathbf{i}_t$ (update gate) of the feature filter. A white pixel corresponds to a value of one, meaning information of the current time step is used. A black pixel corresponds to a value of zero, meaning information of previous time steps is used.

The feature filter successfully integrates information over time. It has learned a meaningful data-dependent weighting between previous information stored in the hidden filter state and information provided by new frames. At each time step, the semantic prediction of the model is improved by using new information from image parts that were previously not visible due to noise. In the first time step, there is no prior knowledge, so the model fully relies on the information of the current frame. This behavior is indicated by the fully white image of the update gate for frame 1. In the following time steps only parts of the update gate image are white. These parts correlate primarily with unseen regions or areas / objects for which only inaccurate information is available (*e.g.* the lower left quarter of frame 2). The same behavior can be seen in Table 5.2, which reports the mean IoU score on a per-frame basis, computed using 300,000 test sequences. The mean IoU score of our model increases over time due to new information.

Frame 1	Frame 2	Frame 3	Frame 4
$32.8\,\%$	$42.0\,\%$	$46.2\,\%$	$46.8\,\%$

Table 5.2: Mean IoU score of the MPSPNet with feature filter on sequences of the static test dataset computed for every time step. The model aggregates information over time resulting in an increasing performance with every new frame.

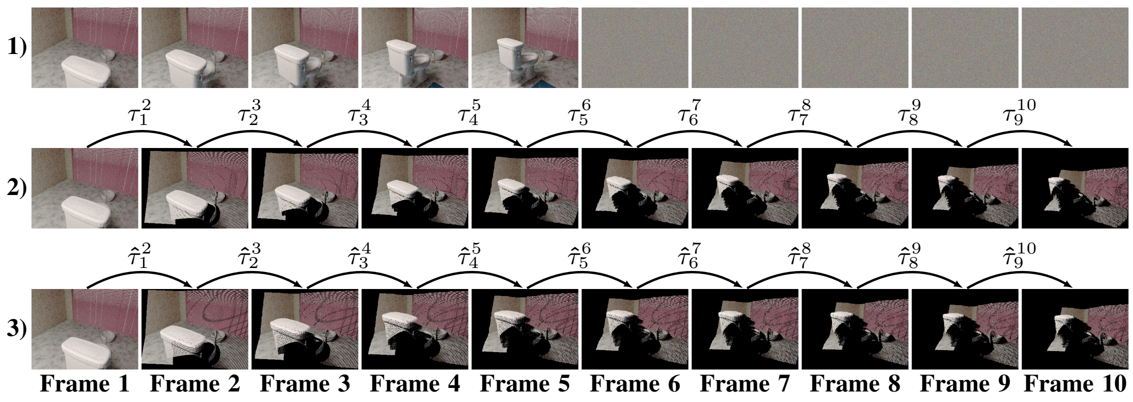

1)
2)
3)
τ_1^2 τ_2^3 τ_3^4 τ_4^5 τ_5^6 τ_6^7 τ_7^8 τ_8^9 τ_9^{10}
$\hat{\tau}_1^2$ $\hat{\tau}_2^3$ $\hat{\tau}_3^4$ $\hat{\tau}_4^5$ $\hat{\tau}_5^6$ $\hat{\tau}_6^7$ $\hat{\tau}_7^8$ $\hat{\tau}_8^9$ $\hat{\tau}_9^{10}$
Frame 1 Frame 2 Frame 3 Frame 4 Frame 5 Frame 6 Frame 7 Frame 8 Frame 9 Frame 10

Figure 5.11: Successive projection of the first frame of a dynamic test sequence, once computed with ground-truth motions τ and once using predicted motions $\hat{\tau}$. We use the ground-truth depth maps for both successive projections. 1) Input test sequence (McCormac *et al.*, 2017) used to estimate the camera motion $\hat{\tau} = \left(\hat{\tau}_1^2, \ldots, \hat{\tau}_9^{10}\right)$. 2) Projection of frame 1 using ground-truth motions $\tau = \left(\tau_1^2, \ldots, \tau_9^{10}\right)$. 3) Projection of frame one using motion estimates $\hat{\tau}$ of our model.

5.5.2 Temporal Motion Integration

In order to obtain a meaningful motion estimate for images that do not contain any information, it is essential to propagate and aggregate dynamics over time. Using a dynamic toy-dataset, we evaluate the ability of our motion filter to perform these two tasks. The generated toy-dataset contains clean sequences of length 10 sampled from the *SceneNet RGB-D* data for which we replace the last N frames with *Gaussian* noise. To create the training dataset consisting of 2 million sequences, we sample the parameter N for each sequence from a discrete uniform distribution with support $k \in \{0, 1, \ldots, 6\}$. To obtain easier interpretable quantitative results in the evaluation, we use a fixed N of five for the test-dataset, which contains 30,000 sequences. An example sequence of the test data is shown in the first row of Figure 5.11. Additionally, we create a clean version of the test set in which all frames are available (*i.e.* N is set to 0).

For the evaluation, we use the MPSPNet pre-trained on clean *SceneNet RGB-D* data and enhance it by the motion filter. The motion filter and motion decoder of the resulting model is fine-tuned on the dynamic toy-data using the two motion losses introduced in Section 5.3.3. In Table 5.3, we report the performance of the resulting model on the

	All frames available		Last 5 frames are *Gaussian* noise	
	ΔT	ΔR	ΔT	ΔR
Frame 1-2	0.0854	0.0339	0.0854	0.0339
Frame 2-3	0.0686	0.0209	0.0686	0.0209
Frame 3-4	0.0648	0.0193	0.0648	0.0193
Frame 4-5	0.0640	0.0189	0.0640	0.0189
Frame 5-6	0.0624	0.0188	0.0831	0.0279
Frame 6-7	0.0624	0.0183	0.0970	0.0303
Frame 7-8	0.0624	0.0185	0.1058	0.0317
Frame 8-9	0.0624	0.0181	0.1127	0.0322
Frame 9-10	0.0632	0.0190	0.1179	0.0336

Table 5.3: Translation norm ΔT in meters and rotation angle ΔR in radian of the relative transformation between predicted and ground-truth motion. Metrics are computed for each frame pair using MPSPNet with motion filter on the two test datasets. Note that due to the filter, the predicted motions depend not only on two time steps, but also on all previous time steps in the sequence.

clean and noisy test data. We use the translation norm ΔT (see Equation 5.18) and the rotation angle ΔR (see Equation 5.19) of the relative transformation between predicted and ground-truth motion as evaluation metrics. The translation norm and rotation angle decrease in the first three to four prediction steps, highlighting the ability of our motion filter to integrate information over time. The results on the clean data suggest, that a longer integration horizon does not yield a further performance gain. As expected, performance decreases when the input images contain only noise (*i.e.* starting from frame 6). However, due to the filter, the predictions are still reasonable for the last 5 frame pairs and the performance only slowly decrease due to accumulating errors. This shows the ability of our model to compensate for short sensor failures.

In Figure 5.11, we visualize for one test sequence the capabilities of our motion filter. Using the proposed model, we estimate the camera transformation between all successive time steps $\hat{\tau} = (\hat{\tau}_1^2, \hat{\tau}_2^3, \dots, \hat{\tau}_9^{10})$. Note that due to the motion filter, the estimation $\hat{\tau}_{t-1}^t$ depends not only on time step t and $t-1$, but also on all previous time steps. To visualize the motion estimates we plot the successive projection of the first frame of the input sequence. The projection is calculated according to Equation 5.5 and Equation 5.6, using ground-truth depth maps and the motion estimates $\hat{\tau}$. As a reference, we additionally plot the successive projection of the first frame, computed based on the ground-truth camera motion τ. Our model is able to predict meaningful motion estimates for all time steps.

	Frame 1	Frame 2	Frame 3	Frame 4	Frame 5	Frame 6	Frame 7
MPSPNet	**39.0 %**	38.5 %	37.9 %	38.8 %	38.4 %	38.1 %	38.1 %
GFNet	37.1 %	39.3 %	39.9 %	40.6 %	40.2 %	40.3 %	**40.4 %**
FMTNet (ours)	38.9 %	**40.8 %**	**41.1 %**	**41.3 %**	**41.0 %**	**40.9 %**	**40.4 %**
MPSPNet	Mean IoU score for frame 7, if the last image (*i.e.* frame 7) in all test sequences is replaced by a white image						5.7 %
GFNet							31.6 %
FMTNet (ours)							**34.1 %**

Table 5.4: Mean IoU score on test sequences which are perturbed by aleatoric noise. MPSPNet: Baseline segmentation model. This model does not use a temporal filter, the predictions thus only depend on the information of one time step. GFNet: MPSPNet enhanced by a model-free, feature-level filter module. FMTNet (ours): MPSPNet enhanced by our modular filter module. In addition to the performance on the test data, we report the mean IoU score for frame 7 on a slightly altered version of the dataset. In this dataset, we replace the last image in all sequences by a white image (see first row of Figure 5.12).

Even for the time steps where the input image is only *Gaussian* noise, the predicted motion is consistent with the ground-truth motion.

5.5.3 Comparison with Baselines

To compare our model FMTNet with the introduced baselines (MPSPNet and GFNet; see Section 5.4.2), we use the test set described in Section 5.4.1. In Table 5.4, we report the mean IoU score of all models on a per-frame basis. Results show an improved performance of the models with filter (GFNet, FMTNet), compared to the unfiltered baseline (MPSPNet). Only for the first frame, MPSPNet outperforms the filtered architectures. This is probably due to the hidden filter state that is not yet properly initialized. Our model surpasses the other filtered baseline, suggesting that a poorly initialized hidden state has less impact. The second observation can most likely be attributed to the modular and explicit structure of our filter and is consistent with the results of Section 5.5.1. Unexpectedly, the performance of our model decreases again from Frame 5 forward. We suspect that this is due to the fairly simple design of our feature update module. A more sophisticated fusion approach could counter this behavior. As an additional measure, one could alter the optimization objective. In the current setting, the predictions off all time-steps are equally weighted in the loss. By putting a higher weight on loss components of later time steps, one could enforce the expected filter behavior.

To evaluate the robustness of the models against short sensor failures, we additionally report the mean IoU score for frame 7 on a slightly altered version of the dataset. In

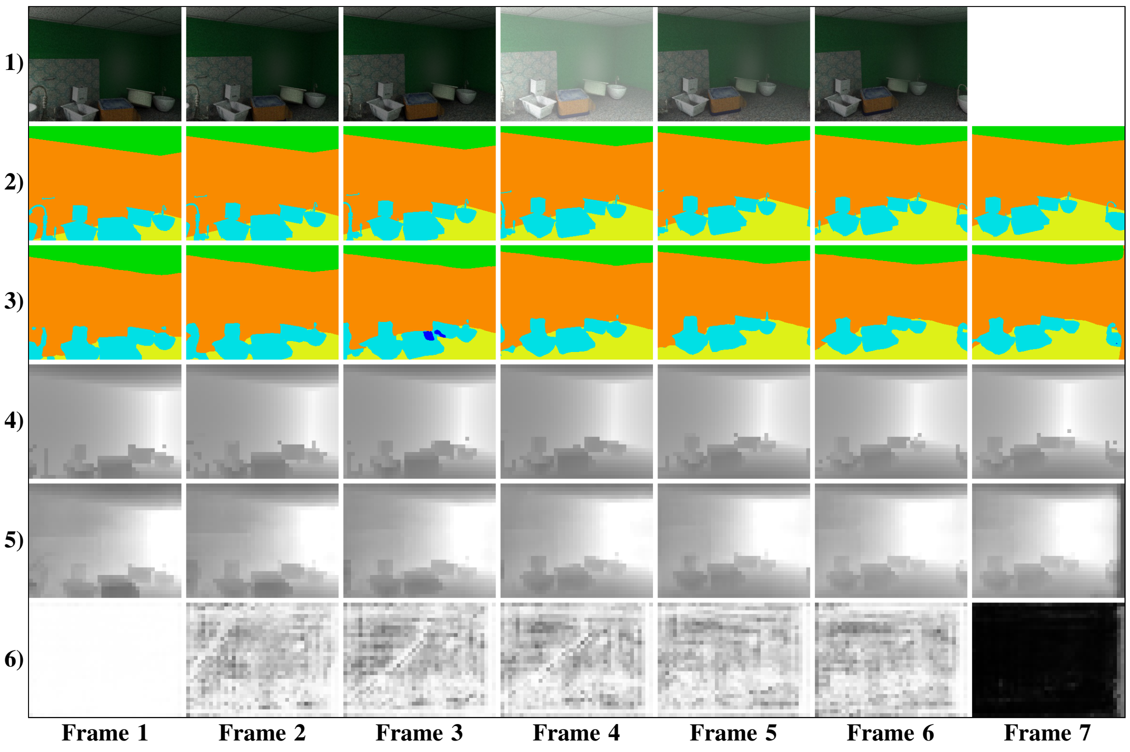


Figure 5.12: Example prediction of our model FMTNet. 1) Input sequence (McCormac *et al.*, 2017) with aleatoric perturbations. The last frame of the sequence is replaced by a white image to simulate a missing frame. 2) Ground-truth semantic segmentation (McCormac *et al.*, 2017). 3) Predicted semantic segmentation of FMTNet. 4) Ground-truth depth map (McCormac *et al.*, 2017). 5) Predicted depth map of FMTNet. 6) Visual representation of the update gate $\mathbf{i}_t$ used in the feature filter. A white pixel corresponds to a value of one, meaning information of the current time step is used. In contrast, a black pixel means that new information is disregarded and that the prediction is solely based on previous time steps.

this data, we replace the last image (*i.e.* frame 7) in all test sequences by a white image, simulating a missing frame. A corresponding sequence is shown in the first row of Figure 5.12. The mean IoU score of all three models on the altered data is reported in the lower half of Table 5.4. As expected, the score of the single-image baseline MPSPNet drops significantly to just $5.7\,\%$, since predictions are only conditioned on empty frames. The two filtered models (GFNet and FMTNet), on the other hand, are still able to achieve a mean IoU score of $31.6\,\%$ and $34.1\,\%$, respectively. Our model FMTNet outperforms the other filtered model by $2.5\,\%$. The performance difference may indicate a better ability of our model to learn a meaningful motion model.

In Figure 5.12, we show an example prediction of our FMTNet on a test sequence whose last frame is missing (*i.e.* the frame is fully white). In addition to visualizing the predicted semantic segmentation (Figure 5.12, row 3), we also show the predicted depth map (Figure 5.12, row 5) and the update gate (Figure 5.12, row 6), which are two of the human interpretable representations computed within our functionally modularized temporal filter. The model is able to predict a meaningful depth map as well as camera motion, which are required to propagate information over time. This is especially visible in the last frame of the sequence – although the last frame is missing, the model is still able to produce a meaningful semantic segmentation. In the last row of Figure 5.12, we show the gate $\mathbf{i}_t$ of our update module. A white pixel corresponds to a gate value of one, which means that the model uses information provided by the current input frame. A black pixel, on the other hand, corresponds to a gate value of zero – the model relies on prior knowledge of previous frames. As expected, the gate of the first frame is fully white, since the filter has to rely on new information. In the last frame, the gate is mainly black, since no meaningful information is provided in that frame. The gate values at the right border of all frames are more white, as the model has never seen these areas before due to camera motion. By inspecting the human interpretable representations of our filter, one is able to debug and verify functionalities.

5.6 Conclusion

In this chapter, we introduced a functionally modularized temporal representation filter to tackle aleatoric failures of a single-frame semantic segmentation model. The main idea behind the filter is to decompose the filter task into less complex and more transparent subtasks. The resulting filter consists of multiple submodules, which can be pre-trained, debugged, and evaluated independently. Our proposed architecture increases the interpretability and transparency of the filter. In our opinion, these are important properties of models used in safety-critical applications. In contrast to many other approaches in the literature, our filter also works in challenging situation, *e.g.* brief sensor outages or adverse weather conditions. Using a photorealistic, synthetic video dataset, we evaluated the different subcomponents of the filter, highlighted possibilities to inspect and debug model behavior, and showed the ability of the proposed architecture to cope with missing information.

Our filter assumes that dynamics are dominated by camera motion. This assumption is only valid for a restricted set of applications. A natural extension of our approach is to explicitly model the motion of dynamic objects. For example, one could predict a segmentation mask and 3D rigid transformation for each moving object (*e.g.* car, bus). These masks and transformations can be used in the prediction module in addition to the estimated camera motion to improve the temporal prediction of the static scene features (Vijayanarasimhan *et al.*, 2017; Yang and Ramanan, 2021). Additionally, it would be interesting to evaluate other options for the different submodules. For example, one could investigate more sophisticated feature fusion approaches (*e.g.* incorporating the uncertainty of the depth and semantic prediction) or evaluate the impact of more interpretable motion filters (*e.g.* using a *Kalman* filter to integrate motion).

Following the publication of the contents of this chapter (Wagner *et al.*, 2018a), different models for the generation of more robust and reliable semantic environment representations were presented. To best utilize the information of previous time steps, Pfeuffer *et al.* (2019) evaluate different options for the placement of convolutional LSTM cells in a modern multi-branch semantic segmentation architecture (Zhao *et al.*, 2018). Their results provide valuable insights on the placement of recurrent cells depending on the maximum available inference time. In a subsequent paper, Pfeuffer and Dietmayer (2019) propose to use spatially separable convolutional LSTM cells to reduce inference time as well as the model parameter count. Based on the work of Pfeuffer *et al.* (2019) and Pfeuffer and Dietmayer (2019), Pfeuffer and Dietmayer (2020) propose a time-efficient video segmentation model and highlight improvements in terms of robustness compared to a single-image baseline in simulated and real adverse weather conditions. Although this chapter focuses on an encoder-decoder architecture for semantic segmentation (Zhao *et al.*, 2017), our presented filter can be seamlessly integrated into other architectures, like multi-branch segmentation networks (Zhao *et al.*, 2018). When filtering multiple representations of a model (see Chapter 4), one can additionally save complexity by predicting the parameters of the geometric projection (*i.e.* the depth map and camera motion) once and using the estimates in all filters.

Multiple approaches rely on optical flow estimates (Dosovitskiy *et al.*, 2015; Ilg *et al.*, 2017) to temporally propagate features (Xu *et al.*, 2018; Jain *et al.*, 2019; Saemann *et al.*, 2019; Zhuang *et al.*, 2021). Given the feature map of the current time step as well as the warped features of previous time steps, Saemann *et al.* (2019) propose to perform a confidence-based fusion of the feature maps. The confidence maps are derived from the softmax scores of the pixel-wise semantic segmentation. To obtain better calibrated uncertainties, they perform temperature scaling (Guo *et al.*, 2017). Due to the reliance on the optical flow, this approaches is in our opinion not well suited for solving aleatoric failures of single-frame segmentation models (see Section 5.2). Zhuang *et al.* (2021) propose to perform a distortion-aware feature fusion. The distortion maps used in the weighted sum of the feature fusion step encode the quality of the optical flow on a per-pixel basis — *i.e.* regions for which the optical flow is inaccurate have a high distortion score. Our proposed filter module could also benefit from conditioning feature fusion on the predictive uncertainty of the motion filter. Compared to our work, Zhuang *et al.* (2021) aim to reduce the overall computational complexity by reusing features computed in previous time steps. Therefore, they only apply the full semantic segmentation model to a reduced set of key-frames and use a lightweight model for the other frames to predict and correct distorted regions of the propagated representation.

In this chapter, we proposed a filter approach which provides additional insights into the inner workings of the model by means of a modular structure as well as human interpretable intermediate representations. Alternatively, one could also use post-hoc explanation methods to derive further insights into the decision-making process of a model. We propose and discuss such a post-hoc explanation approach in the next chapter.

6

Explaining Model Predictions

The increasing use of neural networks in safety-critical, high-risk domains, like autonomous driving, creates the need for methods to inspect and explain their behavior. Especially for verification and validation of neural networks, it is essential to gain insights into their decisions, limitations, as well as possible shortcomings of training data. In addition to safety aspects, explainability can also help to strengthen user confidence and offers new possibilities to derive model improvements.

In this chapter, we propose a new post-hoc, optimization-based explanation method (FGVis), which generates detailed, class-discriminative visual explanations. Our method is based on a novel technique to defend against adversarial evidence (*i.e.* faulty evidence due to artifacts) by filtering gradients during optimization. The defense does not depend on human-tuned parameters. Unlike previous work, the visual explanations of our approach are both fine-grained and preserve the characteristics of images, such as edges and colors. Thus, they are intuitively interpretable, well suited for visualizing detailed evidence, and can be tested as they are valid model inputs. Using a variety of neural network architectures, we evaluate our proposed visual explanation method qualitatively and quantitatively.

6.1 Introduction

Convolutional Neural Networks (CNNs) have proven to produce state-of-the-art results on a multitude of vision benchmarks, such as *ImageNet* (Russakovsky *et al.*, 2015), *Caltech* (Dollár *et al.*, 2009b), or *Cityscapes* (Cordts *et al.*, 2016). Their superior performance has led to CNNs being used in numerous real-world systems (*e.g.* autonomous vehicles) and services (*e.g.* language translation). The transition from classical, rule-based / hand-engineered approaches to fully learned models is straight forward in low risk domains which are not subject to strict safety standards. Web translation services, for example, do not have to comply with any safety regulations and are not subject to corresponding audits. The use of CNNs in safety-critical domains, on the other hand, presents researchers and engineers with new challenges due to their black-box character. The high complexity of CNNs usually goes hand in hand with a low level of transparency and interpretability (Arrieta *et al.*, 2020). However, these properties are critical to verify and validate models, understand failure cases as well as limitations, and uncover short-comings of training data. To overcome the interpretability and transparency disadvantage of black-box models, post-hoc explanation methods have been introduced (Springenberg *et al.*, 2015; Zhou *et al.*, 2016; Zhang *et al.*, 2016; Selvaraju *et al.*, 2017; Fong and Vedaldi, 2017; Dabkowski and Gal, 2017; Petsiuk *et al.*, 2018; Frosst and Hinton, 2017; Ghorbani *et al.*, 2019). Such methods provide insights into the decision-making process of a model. Post-hoc explanation methods can be divided into two categories: global explanation methods and local explanation methods. Global explanation methods seek to characterize the overall model behavior. They specify the model output for all possible inputs in an intuitive manner (Zilke *et al.*, 2016; Frosst and Hinton, 2017), provide insights into the internal data representation of a network (Zhou *et al.*, 2015; Nguyen *et al.*, 2016), or uncover generic decision patterns (Ghorbani *et al.*, 2019). Local explanation methods, on the other hand, focus on individual model inputs. Such methods explain the prediction of a CNN for an individual input image. A common concept is to highlight the evidence on which a model bases its decisions (Selvaraju *et al.*, 2017; Fong and Vedaldi, 2017; Sundararajan *et al.*, 2017; Petsiuk *et al.*, 2018). The most common form of explanations are visual, image-like representations, which depict the important pixels or image regions in a human interpretable manner (see Figure 6.1). In this work, we focus on such visual, local explanation methods for CNNs. For the sake of brevity,

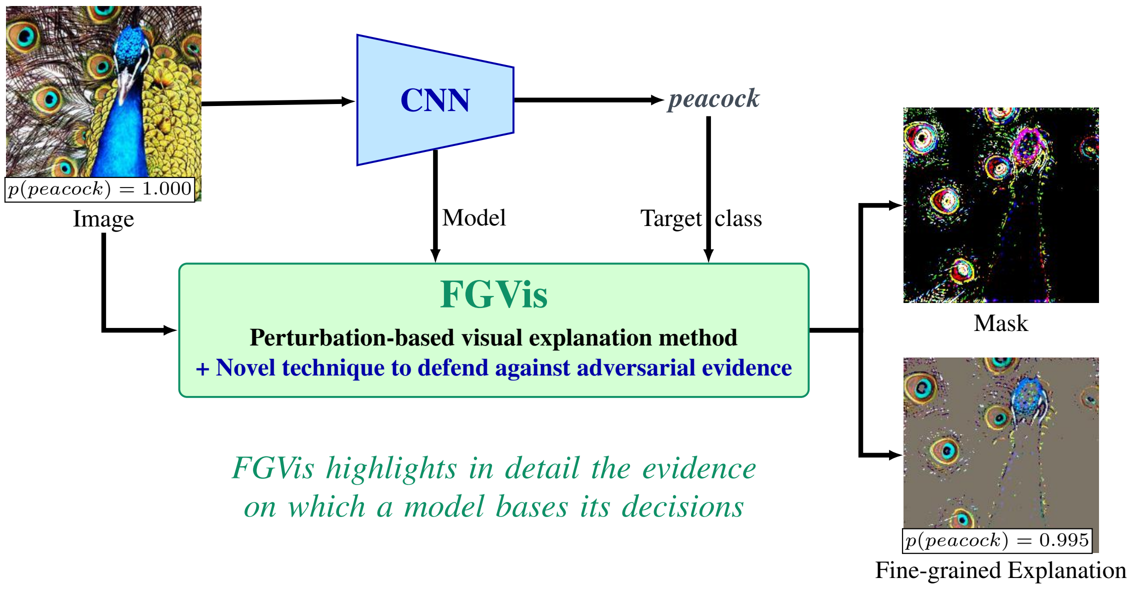

$p(peacock) = 1.000$
Image
CNN
peacock
Model
Target class
FGVis
Perturbation-based visual explanation method
+ Novel technique to defend against adversarial evidence
FGVis highlights in detail the evidence on which a model bases its decisions
Mask
$p(peacock) = 0.995$
Fine-grained Explanation

Figure 6.1: Fine-grained visual explanation computed by removing irrelevant pixels. The provided input image is classified by the CNN with a score of 1.000 as class *peacock*. Our method (FGVis) tries to find a sparse explanation mask with all irrelevant pixels set to zero, *i.e.* they visually appear black. The resulting visual explanation, *i.e.* 'image × mask', is optimized in the image space and can thus be directly used as a model input. The optimization is parameterized to produce an explanation with a softmax score comparable to the image. The input image containing the peacock is from the *ImageNet* (Russakovsky *et al.*, 2015) dataset.

we will use the term explanation as a synonym for local explanation in the remainder of this chapter.

A visual explanation should be easy to interpret, intuitive, and comprehensible not only to experts but also to users of a black-box model. Explanations accessible to end-users may help to increase the acceptance of a model, as users tend to trust a model more when they understand its decision-making process and are able to anticipate or verify results (McAllister *et al.*, 2017). Additionally, a visual explanation should be class-discriminative (*i.e.* focus on one object) and fine-grained (Selvaraju *et al.*, 2017). The latter property is particularly important for domains, where fine structures have a major influence on the prediction of a model (*e.g.* classification tasks in the medical domain). Besides, a visual explanation method should also capture the importance of different color channels, *e.g.* to detect a color bias in the training dataset.

Moreover, visual explanations should be faithful, meaning they accurately explain the function of the black-box model (Selvaraju *et al.*, 2017). A common approach to evaluate and verify the faithfulness of explanations is to use human-based metrics, such as weaklocalization or pointing game performance (Selvaraju *et al.*, 2017; Petsiuk *et al.*, 2018; Simonyan *et al.*, 2014; Fong and Vedaldi, 2017). These metrics depend on human labels and are thus biased towards human perception (Petsiuk *et al.*, 2018). They are a good indicator to check whether an explanation is consistent with a human's reasoning, but are not suitable for verifying whether an explanation correctly represents the evidence on which a model bases its prediction. To overcome this drawback, recent works (Selvaraju *et al.*, 2017; Petsiuk *et al.*, 2018; Chattopadhay *et al.*, 2018) have introduced faithfulness metrics which are directly based on model predictions for explanations. To be able to compute such metrics without having to rely on proxy measures (Selvaraju *et al.*, 2017), it is beneficial to employ explanation methods which generate valid model inputs, *e.g.* a perturbed version of the input image.

A major concern of visual explanation methods is adversarial evidence, *i.e.* faulty evidence generated by artifacts introduced in the computation of the explanation. Especially optimization-based explanation methods are susceptible to adversarial evidence and consequently employ additional constraints or regularizations to prevent artifacts (Fong and Vedaldi, 2017; Dabkowski and Gal, 2017; Du *et al.*, 2018). A drawback of these defenses are added hyperparameters and the necessity of either a reduced resolution of the explanation or a smoothed explanation, making them not well-suited for displaying fine-grained evidence.

In this chapter, we propose a new adversarial defense technique which selectively filters gradients in the optimization which would lead to adversarial evidence otherwise (see Section 6.4). Using this defense, we extend the work of Fong and Vedaldi (2017) and derive a new fine-grained visual explanation method (FGVis). Unlike previous defenses, the proposed one does not depend on hyperparameters and is the key to producing fine-grained visual explanations as no smoothing or regularizations are necessary. Like other optimization-based approaches, FGVis computes a perturbed version of the input image, in which either all irrelevant or the most relevant pixels are removed. The resulting explanations are valid model inputs and their faithfulness can thus be directly verified. Moreover, they are additionally fine-grained and preserve the characteristics of images, such as edges and colors. To the best of our knowledge, FGVis is the first method to be able to produce fine-grained explanations directly in the image space. The image-like nature of the resulting visual explanations ensures an intuitive interpretability, even for non-experts (see Figure 6.1). We evaluate our defense technique as well as FGVis qualitatively and quantitatively in a multitude of experiments.

Our proposed post-hoc, visual explanation method (FGVis) provides insights into the decision-making process of an existing black-box model. Alternatively, one can design a model to have a higher degree of interpretability and transparency by, for example, introducing human interpretable intermediate representations. We present such a model, which is inherently more interpretable and transparent, in Chapter 4.

The content presented in this chapter was first published at the Conference on Computer Vision and Pattern Recognition (Wagner *et al.*, 2019)[1].

6.2 Related Work

A variety of publications have introduced methods for explaining the behavior of models and designing more interpretable deep learning architectures. Detailed surveys of the different approaches are presented by Zhang and Zhu (2018) and Du *et al.* (2019).

In this section, we provide an overview of post-hoc explanation methods which generate visual, image-like explanations for individual model inputs (*i.e.* image-like, local explanations). In particular, we focus on explanation methods for *Convolutional Neural Networks* (CNNs). To structure the overview, we subdivided these explanation methods into three categories: *Backpropagation-Based Methods* (BBMs), *Activation-Based Methods* (ABMs), and *Perturbation-Based Methods* (PBMs).

6.2.1 Backpropagation-Based Methods (BMMs)

BBMs generate an importance measure for each pixel by backpropagating an error signal to the input image. Simonyan *et al.* (2014), building on prior work by Baehrens *et al.* (2010), use the derivative of a class score with respect to the input image as an importance measure. Similar approaches have been introduced by Zeiler and Fergus (2014) and Springenberg *et al.* (2015), who additionally manipulate the gradient when

[1]Wagner, J., J. M. Köhler, T. Gindele, L. Hetzel, J. T. Wiedemer, and S. Behnke (2019). "Interpretable and Fine-Grained Visual Explanations for Convolutional Neural Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9089–9099. DOI: 10.1109/CVPR.2019.00931; © 2019 IEEE

backpropagating through ReLU nonlinearities (Maas *et al.*, 2013). The approach *Integrated Gradients* of Sundararajan *et al.* (2017) additionally accumulates gradients along a path from a base image to the input image. *SmoothGrad* (Smilkov *et al.*, 2017) and *VarGrad* (Adebayo *et al.*, 2018a) visually sharpen explanations by combining multiple explanations of noisy copies of the image. Other BBMs such as *Layer-wise Relevance Propagation* (Bach *et al.*, 2015), *Excitation Backprop* (Zhang *et al.*, 2016), or *DeepLift* (Shrikumar *et al.*, 2017) utilize top-down relevancy propagation rules.

BBMs usually have a low computational cost and produce fine-grained importance / relevancy maps. However, these maps are generally of low quality (Dabkowski and Gal, 2017; Du *et al.*, 2018) and are less interpretable. To verify their faithfulness, it is additionally necessary to apply proxy measures or use pre-processing steps, which may falsify the result.

6.2.2 Activation-Based Methods (ABMs)

Activation-based approaches use a linear combination of activation maps from convolutional layers to form a visual explanation. Prominent methods of this category are *Class Activation Mapping* (CAM), introduced by Zhou *et al.* (2016), and its generalizations *Gradient-Weighted Class Activation Mapping* (Grad-CAM) (Selvaraju *et al.*, 2017) and *Grad-CAM++* (Chattopadhay *et al.*, 2018). These methods differ mainly in the calculation of the linear combination weights and in the restrictions they impose on the CNN. Extensions of such approaches have been proposed by Selvaraju *et al.* (2017) and Du *et al.* (2018), which combine ABMs with backpropagation or perturbation-based approaches.

ABMs generate easy to interpret heat-maps which can be overlaid on the input image. However, they are generally not well-suited to visualize fine-grained evidence or color dependencies. Additionally, it is not guaranteed that the resulting explanations are faithful and reflect the decision-making process of the model (Du *et al.*, 2018; Selvaraju *et al.*, 2017).

6.2.3 Perturbation-Based Methods (PBMs)

Such methods perturb the input image and monitor the prediction of the model. Zeiler and Fergus (2014) slide a grey occlusion square over the image and use the change in class score as a measure of importance. Several approaches are based on this idea, but use other occlusion strategies or importance measures. Petsiuk *et al.* (2018) use randomly sampled occlusion masks and define importance based on the expected model score over sampled masks. The approach *Local Interpretable Model-Agnostic Explanations* (LIME) of Ribeiro *et al.* (2016) uses a super-pixel-based occlusion strategy and a surrogate model to compute importance scores. Further super-pixel or segment-based methods are introduced by Seo *et al.* (2019) and Zhou *et al.* (2015).

The so far mentioned PBM approaches do not need access to the internal state or structure of the CNN. However, their calculation is often quite time-consuming and they only generate coarse visual explanations.

Other PBMs generate an explanation by optimizing for a perturbed version of the input image (Dabkowski and Gal, 2017; Fong and Vedaldi, 2017; Du *et al.*, 2018; Chang *et al.*, 2019). The perturbed image $\mathbf{e}$ is defined by:

$$\mathbf{e} = \mathbf{m} \cdot \mathbf{x} + (1 - \mathbf{m}) \cdot \mathbf{r}, \tag{6.1}$$

where $\mathbf{m}$ is a pixel-wise mask, $\mathbf{x}$ the input image, and $\mathbf{r}$ a reference image containing little information. Common references are a constant value (*e.g.* zero image), *Gaussian* noise image, or a blurred version of the input image (Fong and Vedaldi, 2017; Dabkowski and Gal, 2017). A more recent PBM approach by Chang *et al.* (2019) uses a generative model to sample reference images.

To avoid adversarial evidence, PBMs utilize additional regularizations (Fong and Vedaldi, 2017), constrain the explanation (*e.g.* optimize for a coarse mask (Chang *et al.*, 2019; Fong and Vedaldi, 2017; Du *et al.*, 2018)), introduce stochasticity (Fong and Vedaldi, 2017), or utilize regularizing surrogate models (Dabkowski and Gal, 2017). These approaches generate easy to interpret explanations in the image space, which are valid model inputs and faithful (*i.e.* a faithfulness measure is incorporated in the optimization).

Our method also optimizes for a perturbed version of the input. Compared to existing approaches, we propose a new adversarial defense technique which filters gradients during optimization. This defense does not rely on hyperparameters which have to be fine-tuned. In addition, we optimize each pixel individually, thus the resulting explanations are not limited in resolution and are fine-grained.

6.3 Optimization-based Visual Explanations

In this section, we briefly review the general paradigm of optimization-based visual explanation methods, which compute an explanation by optimizing for a perturbed version of the input image (Fong and Vedaldi, 2017; Du *et al.*, 2018; Chang *et al.*, 2019; Dabkowski and Gal, 2017). Following this paradigm, a visual explanation is defined by:

Explanation by Preservation: The smallest subset of the input image which must be retained to preserve the model output (*i.e.* minimal sufficient evidence).

Explanation by Deletion: The smallest subset of the input image which must be deleted to change the prediction of the model.

To formally derive a visual explanation method based on this paradigm, we assume that a CNN f_{cnn} is given which maps an input image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$ to an output $\mathbf{y}_x = f_{cnn}(\mathbf{x}; \theta_{cnn})$. The output $\mathbf{y}_x \in \mathbb{R}^C$ is a vector representing the softmax scores y_x^c of the different classes c. Given an input image $\mathbf{x}$, a visual explanation $\mathbf{e}^*_{c_T}$ for a target class c_T (*e.g.* the most-likely class $c_T = c_{ml}$) is computed by removing either relevant (*explanation by deletion*) or irrelevant information (*explanation by preservation*) from the image. Since it is not possible to remove information without replacing it, and we do not have access to the process that generated the image, we have to use an approximate removal operator (Fong and Vedaldi, 2017). A common approach is to use a mask-based operator Φ, which computes a weighted average between the input image $\mathbf{x}$ and a reference image $\mathbf{r}$, using a mask $\mathbf{m}_{c_T} \in [0,1]^{3\times H\times W}$:

$$\mathbf{e}_{c_T} = \Phi(\mathbf{x}, \mathbf{m}_{c_T}) = \mathbf{x} \cdot \mathbf{m}_{c_T} + (1 - \mathbf{m}_{c_T}) \cdot \mathbf{r}. \tag{6.2}$$

Common choices for the reference image $\mathbf{r}$ are constant values (*e.g.* a zero image), a blurred version of the original image, *Gaussian* noise, or sampled references of a gen-

Reference image r	***AlexNet***	***GoogleNet***	***VGG16***	***ResNet50***
Blurred *ImageNet* image ($\sigma_b = 5$)	3.67 ± 1.12	3.15 ± 1.31	4.08 ± 1.43	2.38 ± 1.58
Blurred *ImageNet* image ($\sigma_b = 10$)	4.56 ± 0.88	4.09 ± 1.08	4.83 ± 0.86	3.22 ± 1.25
Gaussian noise image ($\sigma_n = 8$)	5.11 ± 0.16	4.62 ± 0.16	5.59 ± 0.09	4.56 ± 0.14
Gaussian noise image ($\sigma_n = 32$)	2.61 ± 0.29	4.67 ± 0.22	4.38 ± 0.23	4.07 ± 0.30
Zero image	6.90	4.08	6.31	5.09
ImageNet image	1.73 ± 1.43	1.09 ± 1.14	1.06 ± 1.22	0.67 ± 0.91
Maximum (1000 classes)	6.91	6.91	6.91	6.91

Table 6.1: Entropy of reference images $\mathbf{r}$ for different convolutional architectures. The entropy is averaged over 1000 random instances of each reference image. For all random references we report the mean $\pm$ standard deviation of the entropy. *Gaussian* noise images are generated by independently sampling for each pixel from a *Gaussian* distribution with zero-mean and a standard deviation of σ_n. The blurred *ImageNet* (Russakovsky *et al.*, 2015) images are computed using a *Gaussian* blur filter with a standard deviation of σ_b. The maximum possible entropy is listed in the last row of the table. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

erative model (Fong and Vedaldi, 2017; Du *et al.*, 2018; Chang *et al.*, 2019; Dabkowski and Gal, 2017). A good reference image $\mathbf{r}$ should carry little information and lead to a model prediction with a high entropy, meaning, ideally all classes are assigned the same softmax score. To compare references, we report in Table 6.1 the predictive entropy for multiple models given different references. For all models except *GoogleNet* the zero image reference has the highest entropy. Interestingly, for the zero image reference, the more recent architectures (*i.e.* *GoogleNet* and *ResNet50*) have a lower entropy. For comparison, we report the entropy of 1000 random *ImageNet* validation images for the different models.

Due to the high entropy and the low computational effort, we use the zero reference image in this work. In our opinion, this reference additionally produces the most pleasing visual explanations, as irrelevant image areas are set to zero[2] (see Figure 6.2) and not replaced by other structures. We did not consider model-based references as proposed by Chang *et al.* (2019), since these introduce significant additional computational efforts.

In addition to an approximate removal operator, a similarity metric $\varphi(y_x^{c_T}, y_e^{c_T})$ is needed, which measures the consistency of the model output generated by the explanation $y_e^{c_T}$ and the output of the image $y_x^{c_T}$ with respect to a target class c_T. This similarity metric should be small if the explanation preserves the output of the target class and large if the explanation manages to significantly drop the probability of the target class (Fong and Vedaldi, 2017). Typical choices for the metric are the cross-entropy with the class c_T

[2]The models used in this work operate on zero-centered data. A value of zero for a tensor $\mathbf{x}, \mathbf{e}, \mathbf{r}$ thus corresponds to a grey color, *i.e.* the color of the data mean.

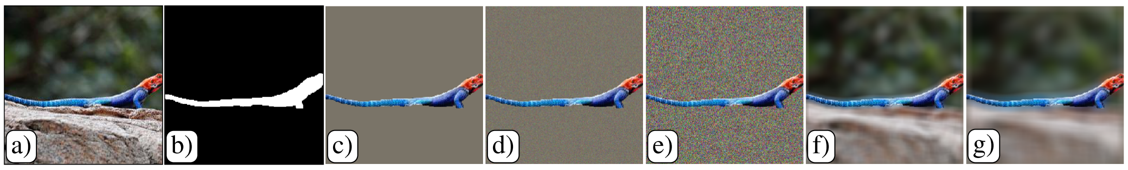


Figure 6.2: Visual explanations (c - g) computed using a mask-based removal operator (see Equation 6.2) for different reference images $\mathbf{r}$. The mask (b) is hand-labeled to only contain pixels of the *agama*. The zero image reference (c) produces the most pleasing visual explanation in our opinion, as irrelevant image areas visually appear gray (*i.e.* the color of the data mean) and are not replaced by other structures. Used references: c) Zero image; d) *Gaussian* noise image ($\sigma_n = 8$); e) *Gaussian* noise image ($\sigma_n = 32$); f) Blurred version of the image ($\sigma_b = 5$); g) Blurred version of the image ($\sigma_b = 10$). The input image (a) containing the agama is from the *ImageNet* (Russakovsky *et al.*, 2015) dataset.

as a hard target (Hinton *et al.*, 2014) or the negative softmax score of the target class. The similarity metric ensures that the explanation remains faithful to the model and thus accurately explains its function. This property is a major advantage of optimization-based explanation methods.

Using a mask-based removal operator with a zero reference image ($\mathbf{r} = \mathbf{0}$) as well as a similarity metric φ, a *preserving explanation* can be computed by:

$$\begin{aligned} \mathbf{e}^*_{c_T} &= \mathbf{m}^*_{c_T} \cdot \mathbf{x}, \\ \mathbf{m}^*_{c_T} &= \underset{\mathbf{m}_{c_T}}{\arg\min} \{\varphi(y_x^{c_T}, y_e^{c_T}) + \lambda \cdot \|\mathbf{m}_{c_T}\|_1\}. \end{aligned} \tag{6.3}$$

In accordance with Fong and Vedaldi (2017) we refer to the optimization in Equation 6.3 as the *preservation game*. Masks (see Figure 6.3 / b2)[3] generated by the *preservation game* are sparse (*i.e.* many pixels are zero / appear black; enforced by minimizing $\|\mathbf{m}_{c_T}\|_1$) and only contain large values at most important pixels. The corresponding explanation $\mathbf{e}^*_{c_T}$ is computed by multiplying the mask $\mathbf{m}^*_{c_T}$ with the image $\mathbf{x}$ (see Figure 6.3 / c2)[3].

Alternatively, we can compute a *deleting explanation* using:

$$\begin{aligned} \mathbf{e}^*_{c_T} &= \mathbf{m}^*_{c_T} \cdot \mathbf{x}, \\ \mathbf{m}^*_{c_T} &= \underset{\mathbf{m}_{c_T}}{\arg\max} \{\varphi(y_x^{c_T}, y_e^{c_T}) + \lambda \cdot \|\mathbf{m}_{c_T}\|_1\}. \end{aligned} \tag{6.4}$$

This optimization will be called *deletion game* (Fong and Vedaldi, 2017) henceforward. Masks (see Figure 6.3 / b1) generated by this game contain mainly ones (*i.e.* appear

[3]The shorthand notation 'Figure 6.3 / b2' references Figure 6.3, column b – 2nd row.

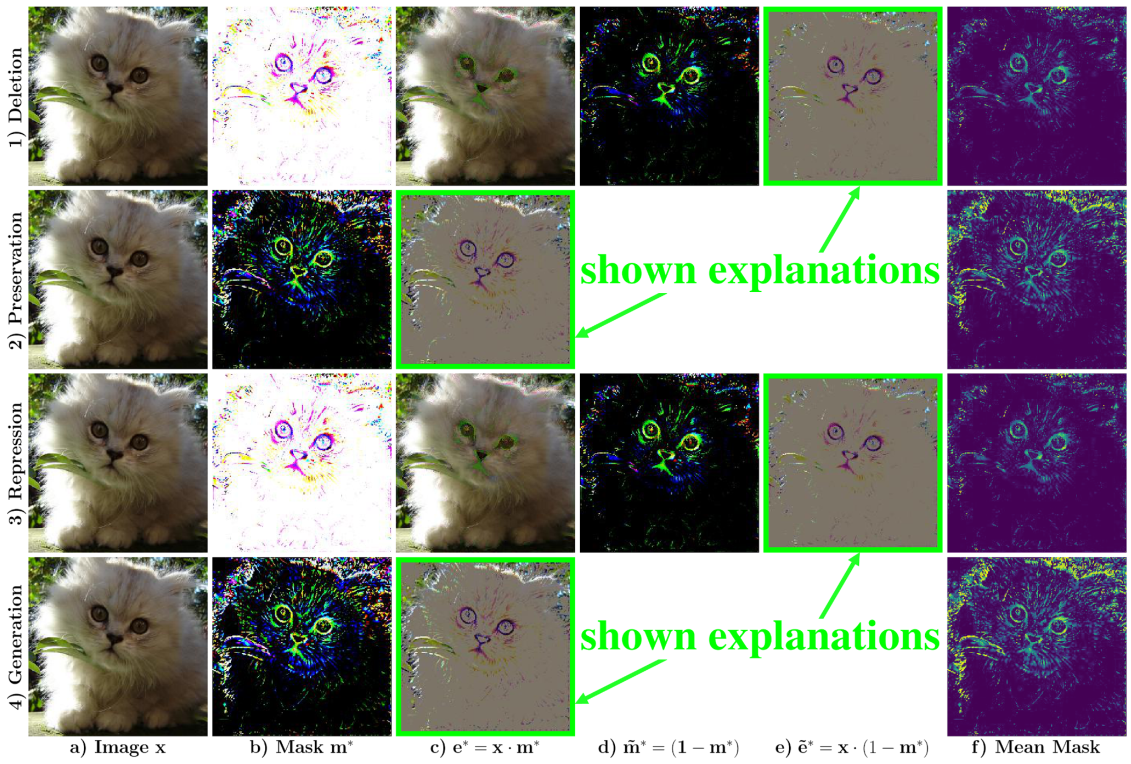


Figure 6.3: Visualizations calculated for *VGG* using different game types (*i.e. deletion game*, *preservation game*, *repression game*, and *generation game*). Subscript c_T ommited to ease readability. a) Input image (Russakovsky *et al.*, 2015). b) Mask obtained by the optimization. Colors in a *deletion / repression* mask are complementary to the image colors. c) Explanation directly obtained by the optimization. d) Complementary mask with a true-color representation for the *deletion game / repression game*. e) Explanation highlighting the important evidence for the *deletion game / repression game*. f) Mean mask: mask / complementary mask averaged over colors. To underline important evidence, we use ($\mathbf{e}$) for the explanation of the *preservation game / generation game* and ($\tilde{\mathbf{e}}$) for the *deletion game / repression game*. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

white; enforced by maximizing $\|\mathbf{m}_{c_T}\|_1$) and show only small values at pixels, which provide the most prominent evidence for the target class c_T. The colors in a mask of the *deletion game* are complementary to the image colors. To obtain a true-color representation analogous to the *preservation game*, one can alternatively visualize the complementary mask (see Figure 6.3 / d1):

$$\tilde{\mathbf{m}}^*_{c_T} = \mathbf{1} - \mathbf{m}^*_{c_T}. \tag{6.5}$$

A resulting explanation of the *deletion game*, as defined by Equation 6.4, is visualized in Figure 6.3 / c1. This explanation is visually very similar to the original image as only a few pixels need to be deleted to change the model output. For better visualization,

Game type		Objective	
		Preserve important evidence	Remove important evidence
Initial mask	Ones	*Preservation game*	*Deletion game*
	Zeros	*Generation game*	*Repression game*

Table 6.2: Overview of different game types (*i.e. deletion game*, *preservation game*, *repression game*, and *generation game*). The objective defines how the resulting visual explanation should differ from the input image. Evidence always refers to a particular target class c_T which is the focus of the explanation. The mask initialization defines the starting condition of the optimization. If the mask is initialized with ones, the first visual explanation is identical to the image. However, if the mask is initialized with zeros, the first visual explanation is empty and ideally contains no evidence towards any class.

we depict in the remainder of the chapter a modified version of the explanation for the *deletion game*:

$$\tilde{\mathbf{e}}^*_{c_T} = \mathbf{x} \cdot (\mathbf{1} - \mathbf{m}^*_{c_T}). \tag{6.6}$$

This explanation has the same properties as the one of the *preservation game*, since it only highlights the important evidence (see Figure 6.3 / e1).

To solve the optimization in Equation 6.3 and Equation 6.4, we utilize *Stochastic Gradient Descent* (SGD) and start with an explanation $\mathbf{e}^0_{c_T} = \mathbf{1} \cdot \mathbf{x}$ identical to the original image (*i.e.* a mask initialized with ones; $\mathbf{m}^0_{c_T} = \mathbf{1}$). The initial explanation is thus always faithful to the model and does not contain any adversarial evidence. As an alternative initialization of the masks, we additionally explore a zero initialization $\mathbf{m}^0_{c_T} = \mathbf{0}$. In this setting, the initial explanation contains no evidence towards any class and the optimization iteratively has to add relevant (*generation game*) or irrelevant, not supporting the class c_T, information (*repression game*). The explanation and mask of the *generation game* are visually comparable to those of the *preservation game*, the same holds for the *deletion game* and *repression game* (see Figure 6.3). To get comparable visualizations, we depict explanations $\mathbf{e}^*$ for the *preservation game* and *generation game* and the modified version of the explanations $\tilde{\mathbf{e}}^*$ for the *deletion game* and *repression game* in the remainder of this chapter. An overview of the four optimization settings (*i.e.* game types) is given in Table 6.2. Additional examples as well as a more detailed discussion of the game types is included in Section 6.5.3.

6.4 Defending Against Adversarial Evidence

CNNs have been proven to be susceptible to adversarial images (Szegedy *et al.*, 2014; Goodfellow *et al.*, 2015; Kurakin *et al.*, 2017). An adversarial image is a perturbed version of a correctly classified image crafted to fool a neural network. Due to the computational similarity of adversarial methods and optimization-based visual explanation approaches, adversarial noise is also a concern for the latter methods. In order to get a meaningful visual explanation, one has to make sure that an explanation is based on true evidence present in the image and not on false adversarial evidence introduced during optimization. This is particularly true for the *generation game* and *repression game* as their optimization start with $\mathbf{m}^0_{c_T} = \mathbf{0}$ and iteratively adds information.

Fong and Vedaldi (2017) were the first to show the vulnerability of optimization-based visual explanation methods to adversarial noise. To avoid adversarial evidence, they use stochastic operations, additional regularizations, and optimize on a low-resolution mask which is upsampled to match the dimensions of the input image. Similar methods are employed by Dabkowski and Gal (2017), Du *et al.* (2018), and Chang *et al.* (2019). Dabkowski and Gal (2017) additionally propose to use a regularizing surrogate model to counter adversarial evidence. In general, these operations impede the generation of adversarial noise by obscuring the gradient direction in which the model is susceptible to false evidence, or by constraining the search space for potential adversarials. These techniques help to reduce adversarial evidence, but also introduce new drawbacks:

1. The defense capabilities usually depend on human-tuned parameters (*e.g.* mask upsamling factor, weighting of additional regularizations).
2. Computed visual explanations are limited to being low resolution and / or smooth, which prevents fine-grained evidence from being visualized.

A novel Adversarial Defense. To avoid the generation of adversarial evidence, we propose a novel adversarial defense which filters gradients during backpropagation in a targeted way. The basic idea of our approach is:

A neuron within a CNN is only allowed to be activated by the visual explanation $\mathbf{e}_{c_T}$ if the same neuron is also activated by the original input image $\mathbf{x}$.

If we regard neurons as indicators for the existence of features (*e.g.* edges, objects) (Zhou *et al.*, 2015), the proposed constraint enforces that the explanation $\mathbf{e}_{c_T}$ only contains features which exist at the same location in the original image $\mathbf{x}$. If, for example, a face is present at a specific location of an input image, the corresponding explanation is only allowed to contain the same face at the same location. The explanation is however not allowed to contain a new, possibly sparser face, in another location. By ensuring that the allowed features in the explanation $\mathbf{e}_{c_T}$ are a subset of the features in the image $\mathbf{x}$ our defense prevents the generation of new evidence.

This defense technique can be integrated into the visual explanation method of Section 6.3 via an optimization constraint:

$$\begin{cases} 0 \leq h_i^l(\mathbf{e}_{c_T}) \leq h_i^l(\mathbf{x}), & \text{if } h_i^l(\mathbf{x}) \geq 0, \\ 0 \geq h_i^l(\mathbf{e}_{c_T}) \geq h_i^l(\mathbf{x}), & \text{otherwise}, \end{cases} \tag{6.7}$$

where h_i^l is the activation of the i-th neuron in the l-th layer of the network after the nonlinearity. For brevity, the index i references one specific feature at one spatial location in the activation map. This constraint is applied after all nonlinearity-layers (*e.g.* ReLU-layers (Maas *et al.*, 2013)) of the CNN, besides the final classification layer. It ensures that the absolute value of activations can only be reduced towards values representing lower information content (we assume that an activation of zero corresponds to the lowest information content as commonly applied in network pruning (Han *et al.*, 2015)). To solve the optimization with subject to Equation 6.7, one could incorporate the constraints via a penalty function in the optimization loss. This approach has the drawback of introducing one additional hyperparameter. Alternatively, one could add an additional layer $\bar{h}_i^l$ after each nonlinearity which ensures the validity of the constraint:

$$\begin{aligned} \bar{h}_i^l(\mathbf{e}_{c_T}) &= \min(b_u, \max(b_l, h_i^l(\mathbf{e}_{c_T}))), \\ b_u &= \max(0, h_i^l(\mathbf{x})), \\ b_l &= \min(0, h_i^l(\mathbf{x})), \end{aligned} \tag{6.8}$$

where $h_i^l(\mathbf{e}_{c_T})$ is the actual activation of the original nonlinearity-layer and $\bar{h}_i^l(\mathbf{e}_{c_T})$ the adjusted activation after ensuring the bounds b_u, b_l of the original input. For instance, for a ReLU nonlinearity, the upper bound b_u is equal to $h_i^l(\mathbf{x})$ and the lower bound b_l is zero. We are not applying this approach as it changes the architecture of the model which we try to explain. Instead, we clip gradients in the backward pass of the optimization, which lead to a further violation of Equation 6.7. This is equivalent to adding an additional clipping-layer after each nonlinearity which acts as the identity in the forward pass and uses the gradient update of Equation 6.8 in the backward pass. When backpropagating an error-signal $\bar{\gamma}_i^l$ through the clipping-layer, the gradient update rule for the resulting error γ_i^l is defined by:

$$\gamma_i^l = \bar{\gamma}_i^l \cdot [h_i^l(\mathbf{e}_{c_T}) \leq b_u] \cdot [h_i^l(\mathbf{e}_{c_T}) \geq b_l], \tag{6.9}$$

where $[\,\cdot\,]$ is the indicator function and b_l, b_u the bounds computed in Equation 6.8. This clipping only affects the gradients of the similarity metric $\varphi(\cdot\,,\cdot)$ which are propagated through the network. The proposed gradient clipping does not add hyperparameters and keeps the original structure of the model during the forward pass. Compared to other adversarial defense techniques (Dabkowski and Gal, 2017; Fong and Vedaldi, 2017; Chang *et al.*, 2019), it imposes no constraint on the explanation (*e.g.* resolution / smoothness constraints), enabling fine-grained visual explanations.

6.5 Experiments

6.5.1 Implementation Details

Unless otherwise specified, the visual explanations are computed for the most-likely class using SGD with a learning rate of 0.1, running for 500 iterations. To improve optimization and avoid instabilities, we initialize the masks with noise sampled for each pixel from a uniform distribution $\mathcal{U}(a, b)$. We use $\mathcal{U}(0, 0.01)$ for the *generation game* and *repression game* and $\mathcal{U}(0.99, 1)$ for the *preservation game* and *deletion game*. We normalize the gradient using its maximum value to prevent large changes of individual mask pixels. For the similarity metric $\varphi(\cdot, \cdot)$ we use the cross-entropy for the *generation game* and *preservation game* and the negative probability for the *deletion game* and *repression game*. During optimization the mask is clipped to the range $[0, 1]$ after each update step.

When computing a visual explanation for the most-likely class, we use a line-search for the parameter λ to determine its optimal value. Unless otherwise noted, we iteratively use 13 equally spaced λ values between 10^{-4} and 10^{-10} and stop when the resulting most-likely class of e_{ml} shifts (*deletion* and *repression game*) or achieves the highest probability among all classes (*preservation* and *generation game*). We use images of the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset and pre-trained model weights (Dieleman *et al.*, 2015). Faces, license plates, and signs are shown pixelated in the illustrations. Four common model architectures are utilized in our experiments: *AlexNet* (Krizhevsky *et al.*, 2012), *GoogleNet* (Szegedy *et al.*, 2015), *ResNet50* (He *et al.*, 2016a), and *VGG16* (Simonyan and Zisserman, 2015).

6.5.2 Validating the Adversarial Defense

Qualitative Evaluation

To qualitatively evaluate the performance of our defense, we compute an explanation for an adversarial class c_A for which there is no evidence in the image (*i.e.* it is visually not present in the image). We approximate c_A with the least-likely class c_{ll} considering only images which yield a very high predictive confidence for the true class $p(c_{true}) \geq 0.995$. The least-likely class should contain very little to no overlapping evidence with the true class, assuming a well-trained model with a high confidence and a dataset with a large number of diverse classes, such as *ImageNet*. Using c_{ll} as the target class, the resulting explanation method without defense is similar to an adversarial attack, the *Iterative Least-Likely Class Method* (Kurakin *et al.*, 2017).

A correct visual explanation for the adversarial class c_A sould be "empty", *i.e.* a grey image, and the model prediction of the visual explanation should not match the adversarial class c_A. If, on the other hand, the explanation method is susceptible to adversarial evidence (e.g.: when no adversarial defense mechanism is applied), the optimization procedure should be able to perfectly generate an explanation for any class. This behavior can be seen in Figure 6.4. When no defense mechanism is used, the explanation method is able to generate a faulty explanation for the adversarial class (c_A: *limousine*). The explanation for class c_A (see Figure 6.4 / c2) contains primarily sparse artificial structures and is classified with a probability of one as *limousine*. The corresponding explanation, computed while employing our proposed adversarial defense, contains no evidence for the adversarial class (see Figure 6.4 / b2). The explanation is grey and yields a zero

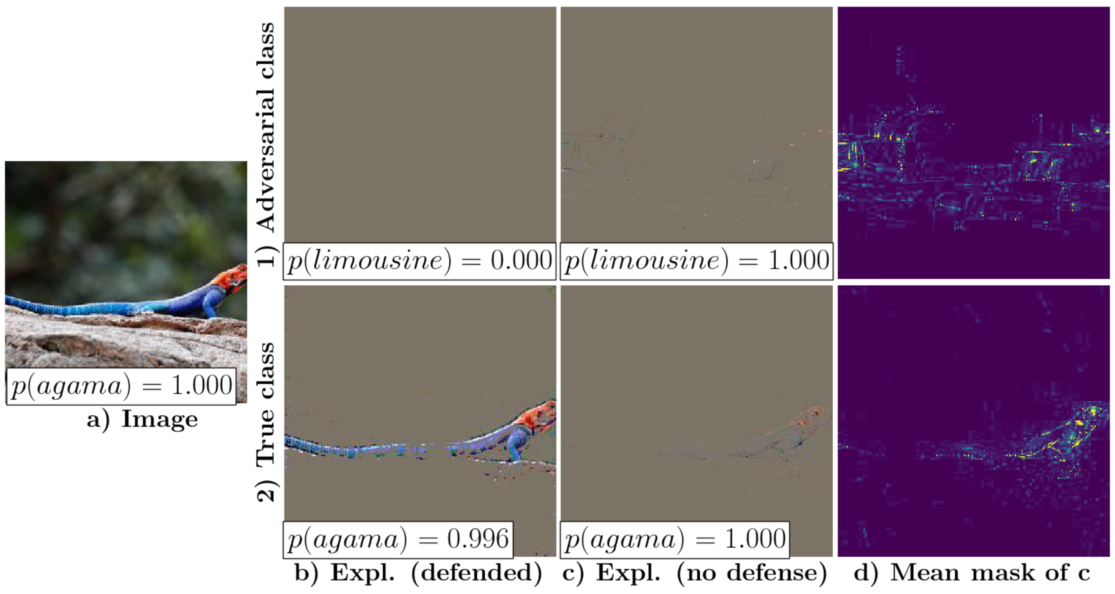


Figure 6.4: Visual explanations computed for the adversarial class *limousine* and the true class *agama* using the *generation game* (learning rate: 0.1; iterations: 500; λ: 1e$-$7) and *VGG16*. The explanations are computed once with and once without our adversarial defense. We approximate the adversarial class with the least-likely class considering an image with a very high confidence for the true class. The predicted softmax score of the target class is noted in the lower left corner of each image. An adversarial for class *limousine* can only be computed without our defense. Column (d) depicts mean masks enhanced by a factor of 7 to show small adversarial structures. The image containing the agama is from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

probability for the adversarial class. To better perceive sparse structures, we additionally show an enhanced mean mask in Figure 6.4 / d. As a reference, we also depict the visual explanation of the true class (c_{true}: *agama*). The optimization with our defense results in a meaningful explanation of the *agama* (see Figure 6.4 / b1). Without defense the explanation (see Figure 6.4 / c1) is much more sparse. Since there is no constraint to change pixel values arbitrarily, we assume that the optimization has introduced additional artificial structures to get a sparser visual explanation.

Figure 6.5 highlights equivalent results using an image of a *peacock* and the adversarial class *rotisserie*. An adversarial explanation for class *rotisserie* can only be computed without our defense.

Quantitative Evaluation

A quantitative evaluation of the proposed defense is reported in Table 6.3. We generate visual explanations for 1000 random *ImageNet* validation images and use the adversarial

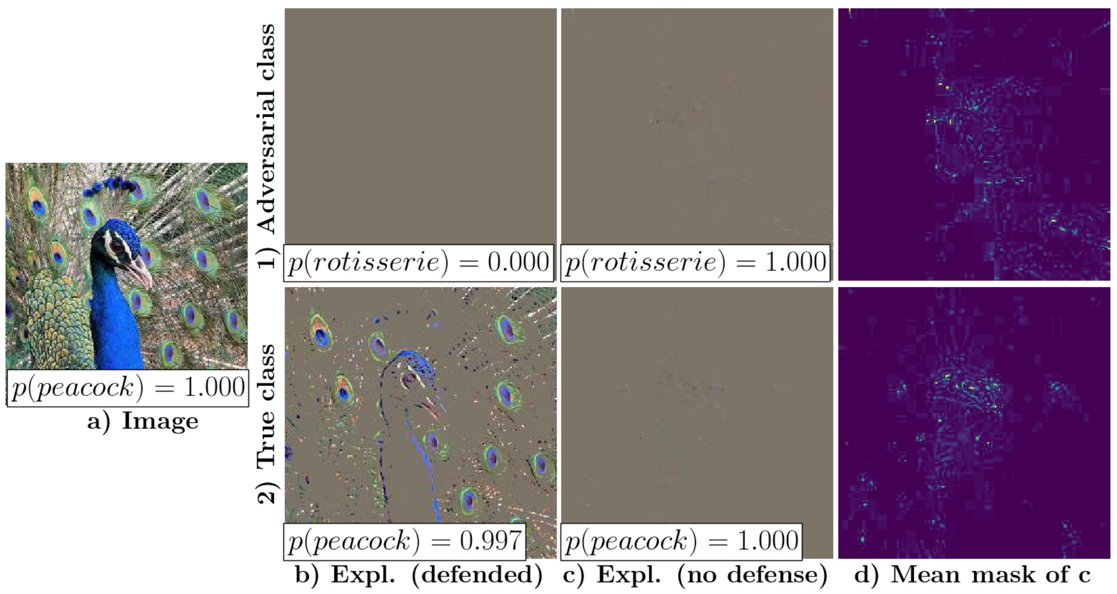


Figure 6.5: Visual explanations computed for the adversarial class *rotisserie* and the true class *peacock* using the *generation game* (learning rate: 0.1; iterations: 500; λ: 1e$-$7) and *VGG16*. The explanations are computed once with and once without our adversarial defense. We approximate the adversarial class with the least-likely class considering an image with a very high confidence for the true class. The predicted softmax score of the target class is noted in the lower left corner of each image. An adversarial for class *rotisserie* can only be computed without our defense. Column (d) depicts mean masks enhanced by a factor of 7 to show small adversarial structures. The input image containing the peacock is from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset.

Model	*GoogleNet*	*VGG16*	*AlexNet*	*ResNet50*
No defense	100 %	100 %	100 %	100 %
Defended	0.0 %	0.2 %	0.0 %	0.0 %

Table 6.3: Ratio how often an adversarial explanation was generated for 1000 random *ImageNet* validation images, using the *generation game* (learning rate: 0.1; iterations: 500; λ: 0). The sparsity loss is set to zero to ease the generation of adversarials. The statistics are computed once with and once without our adversarial defense. The evaluation is conducted using four common model architectures: *AlexNet* (Krizhevsky *et al.*, 2012), *GoogleNet* (Szegedy *et al.*, 2015), *ResNet50* (He *et al.*, 2016a), and *VGG16* (Simonyan and Zisserman, 2015). (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

class c_A as the explanation target. For the adversarial class c_A we use the least-likely class, according to the qualitative evaluation. We use the second least-likely class, if the least-likely class coincidentally matches the predicted class of the zero image (*i.e*. the initial explanation of the optimization). The images are chosen to have a high predictive confidence for the true class $p(c_{true} \geq 0.995)$ to ensure a small amount of overlapping evidence between the true- and least-likely class. To ease the generation of adversarial examples, we set the sparsity loss to zero and only use the similarity metric which tries to maximize the probability of the target class c_A. Without an employed defense technique,

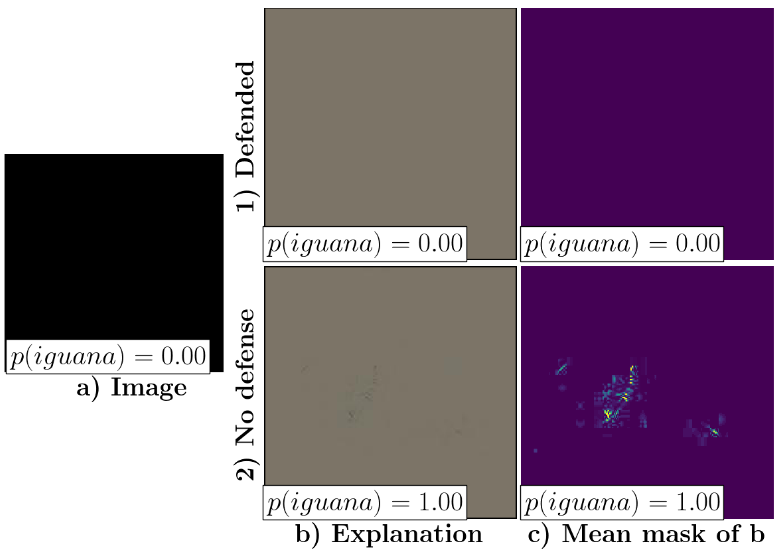


Figure 6.6: Visual explanations computed with the *genaration game* (learning rate: 0.1; iterations: 500; λ: 5e−4) for the adversarial class *iguana* using a black input image and *GoogleNet*. An adversarial explanation can only be computed without our defense. Column (c) depicts mean masks enhanced by a factor of 10. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

Model	*GoogleNet*	*VGG16*	*AlexNet*	*ResNet50*
No defense	100 %	100 %	100 %	100 %
Defended	0.0 %	0.1 %	0.1 %	0.0 %

Table 6.4: Ratio how often an adversarial example was generated from a black input image averaged over 998 *ImageNet* classes, using the *generation game* (learning rate: 0.1; iterations: 500; λ: 0). The sparsity loss is set to zero to ease the generation of adversarials. The statistics are computed once with and once without our proposed adversarial defense. (Wagner *et al.* (2019) / © 2019 IEEE)

the optimization is able to generate an adversarial explanation for 100% of the images. Applying our defense, the optimization nearly never was able to do so. The two adversarial examples generated using *VGG16* have a low confidence, so we assume that there has been some evidence for the chosen class c_A in the image. Our proposed technique is thus well suited to defend against adversarial evidence.

Explanining empty Images

To validate our proposed defense without relying on approximating the adversarial class with the least-likely class, we report an alternative version of the qualitative and quantitative evaluation in Figure 6.6 and Table 6.4, by only using a black image as input. For

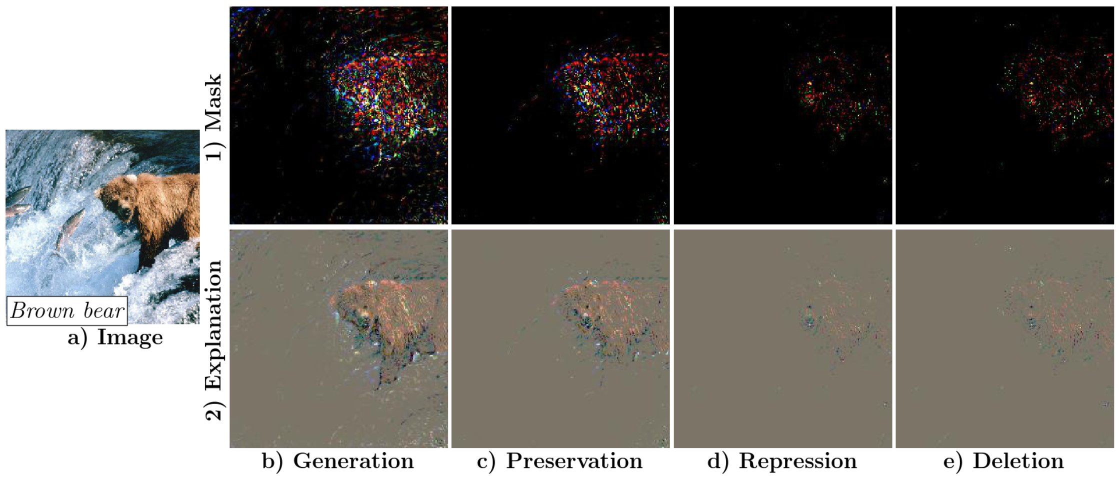


Figure 6.7: Visual explanations and masks computed using the different game types for *GoogleNet* using the target class *brown bear*. The complementary masks $(1 - \mathbf{m})$ are plotted for the *repression game* and *deletion game* to get comparable visualizations for all game types. The input image containing the brown bear is from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

the quantitative evaluation we create visual explanations for 998 *ImageNet* classes, using always the same black input image. We omit the predicted class of the black image and the class of the starting condition (image · zero mask). This experiment is based on the assumption that there is no evidence for the 998 *ImageNet* classes in the black input image. The results are comparable to the once using real input images, fortifying the effectiveness of our adversarial defense.

6.5.3 Comparison of Game Types

In Figure 6.7 and Figure 6.8, the different explanation types (*i.e.* game types; see Section 6.3) are visually compared using *GoogleNet*. To obtain true-color representations analogous to the *preservation game* and *generation game*, we depict the complementary mask for the *repression game* and *deletion game*.

Visual explanations of the *deletion game* and *repression game* are qualitatively similar. The similarity among the two game types is due to both using the same optimization with only a different starting condition: $\mathbf{m}^0_{c_T} = 0$ for the *repression game* versus $\mathbf{m}^0_{c_T} = 1$ for the *deletion game*. The same observation holds for the *generation game* and *preservation game* (see Table 6.2) where visual explanations are in general less sparse than in the *repression game* and *deletion game*. This is most likely due to the fact that only small

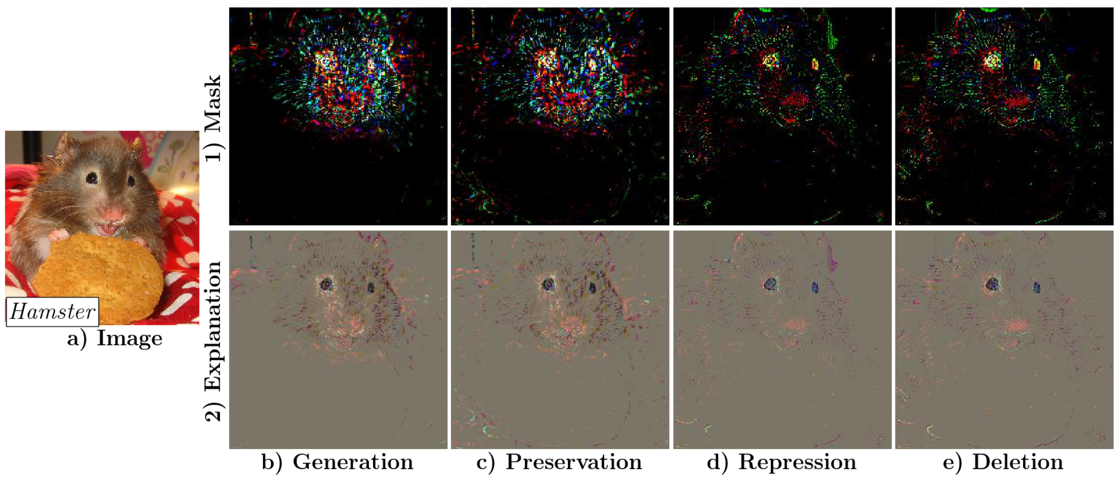


Figure 6.8: Visual explanations and masks computed using the different game types for *GoogleNet* using the target class *hamster*. The complementary masks $(1 - \mathbf{m})$ are plotted for the *repression game* and *deletion game* to get comparable visualizations for all game types. The input image containing the hamster is from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

parts of the image need to be suppressed to change the model output (*e.g.* shifting one breed of dog to another), though, to evoke a certain model output one needs to create sufficient amount of evidence for this class.

During the optimization only pixels containing evidence towards the target class need to be changed for the *generation game* and *deletion game*. The opposite is true for the *preservation game* and *repression game*. Usually only a small subset of the image pixels are relevant for the prediction of a model. Consequently, most of the mask pixels are zero for the *generation / preservation game* and one for the *deletion / repression game*. Due to the usage of complementary masks for the *deletion / repression game* (see Figure 6.7 and Figure 6.8) a value of one appears black in the corresponding mask visualizations.

In our experience the *deletion game* produces the most fine-grained visual explanations. Compared to the other game types it usually needs the least amount of optimization steps. This is caused by the fact that we start with a mask of ones ($\mathbf{m}^0_{c_T} = \mathbf{1}$) and comparatively few mask values need to be changed to delete the most prominent evidence for the target class.

6.5.4 Comparison of Methods

In Figure 6.9, we compare FGVis with other state-of-the-art visual explanation methods. The masks for FGVis are computed using the *deletion game* with the standard parametrization, as described in Section 6.5.1. To ensure that visualizations are comparable to the reference methods we use mean explanation masks.

Explanation masks generated by our method (FGVis) are the most fine-grained. Especially the two other *Perturbation-Based Method*s (PBMs) (Figure 6.9 / e: occlusion-based explanation (Zeiler and Fergus, 2014); Figure 6.9 / f: optimization-based explanation (Fong and Vedaldi, 2017)) as well as the *Activation-Based Method* (ABM) (Figure 6.9 / d: Grad-CAM (Selvaraju *et al.*, 2017)) produce coarser explanations and, therefore, tend to include additional non-essential evidence (*i.e.* background pixel which are not relevant for the prediction of the target class). FGVis mainly highlights pixels which form edges of the chosen target object. This observation is not true for the other methods.

The gradient-based explanations (Figure 6.9 / a (Simonyan *et al.*, 2014)) and the explanations generated by *Guided Backpropagation* (Figure 6.9 / b (Springenberg *et al.*, 2015)) are the most similar to ours. However, our explanations are optimized to be faithful to the model, meaning, by deleting the highlighted pixels from the corresponding images one can prevent the model from correctly classifying the images. This is not necessarily true for the other two methods. To verify the faithfulness of these methods one additionally has to utilize proxy measures or use pre-processing steps, which may falsify the result.

6.5.5 Class-Discriminative / Fine-Grained

A reliable visual explanation method should only highlight evidence that contributes to the prediction of a model. Consequently, an explanation method should be able to produce class-discriminative (*i.e.* focus on one object) and fine-grained explanations (Selvaraju *et al.*, 2017). The latter property is particularly important for domains where only small parts of an object have a major influence on the model prediction (*e.g.* the medical domain). For such domains, a method should only visualize the object parts relevant for the prediction and not the entire object.

To evaluate FGVis with respect to these two properties, we generate visual explanations for images containing two dominant objects (see Figure 6.10 and Figure 6.11). The images are selected to contain objects from highly different categories, ensuring little overlap of evidence between objects. For each image, we generate two explanations using the *deletion game* and *GoogleNet*. The corresponding target class is depicted in the lower left corner of each explanation. We employ the strategy described in Section 6.5.1

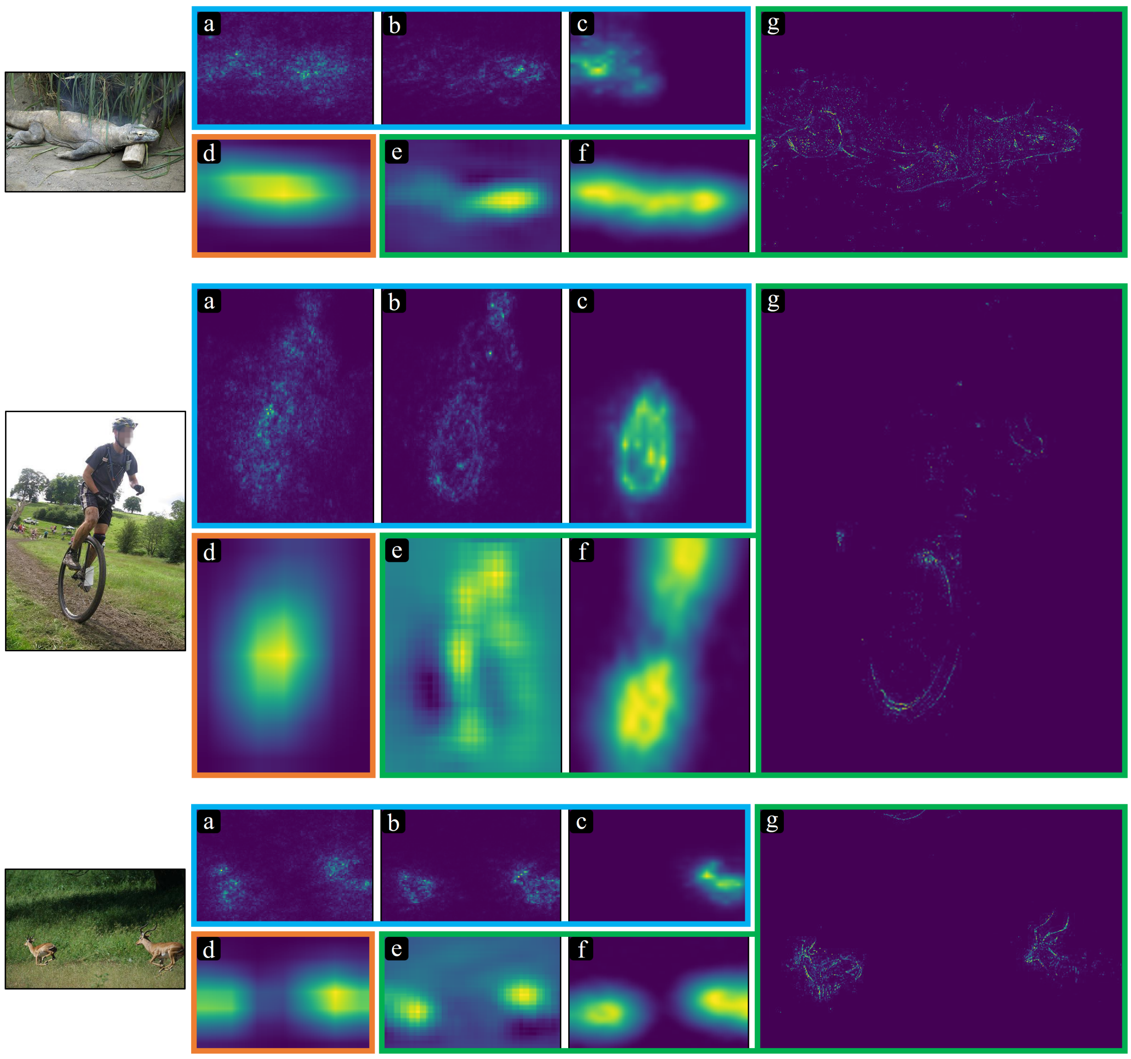


Figure 6.9: Mean explanation masks of different state-of-the-art explanation methods. The masks are computed for *GoogleNet* using the target class *komodo dragon* (top) / *unicycle* (middle) / *impala* (bottom). The three input images are from the *ImageNet* (Russakovsky *et al.*, 2015) dataset.
◇ *Perturbation-Based Methods* (PBMs): a) gradient-based explanation (Simonyan *et al.*, 2014), b) *Guided Backpropagation* (Springenberg *et al.*, 2015), c) *Contrastive Excitation Backprop* (Zhang *et al.*, 2016).
◇ *Activation-Based Methods* (ABMs): d) Grad-CAM (Selvaraju *et al.*, 2017).
◇ *Perturbation-Based Methods* (PBMs): e) occlusion-based explanation (Zeiler and Fergus, 2014), f) optimization-based explanation (Fong and Vedaldi, 2017), g) FGVis (ours).
The masks of all reference methods are based on work by Fong and Vedaldi (2017) (© 2017 IEEE). Due to the detailed and sparse nature of our masks, we plot them in a larger format. FGVis produces the most fine-grained explanations by deleting important pixel of the selected target class. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

to determine the parameter λ for explanations of the most-likely class. To explain the other class, the parameter λ is optimized to significantly reduce the corresponding softmax score.

FGVis is able to generate class-discriminative explanations and only highlights pixels of the chosen target class. Even partially overlapping objects, as the *soccer ball* and *norwegian elkhound* in Figure 6.10 / 1, the *schooner* and *suspension bridge* in Figure 6.11 / 3, or the *tennis balls* and *strainer* in Figure 6.11 / 4 are correctly discriminated. As also noted in Fong and Vedaldi (2017), especially faces seem to have a significant influence on the prediction of animal classes (see Figure 6.10 / c1 and Figure 6.10 / c3).

One major advantage of FGVis is its ability to visualize fine-grained details. This property is especially visible in Figure 6.11 / b2, which shows an explanation for the target class *chainlink fence*. Despite the fine structure of the fence, FGVis is able to compute a precise explanation which mainly contains fence pixels. The ability to generate fine-grained explanations is also beneficial when dealing with small objects such as the *traffic light* in Figure 6.10 / b2.

6.5.6 Investigating Biases of Training Data

Visual explanation methods can be used to identify a bias in the training data. Particularly in safety-critical, high-risk domains (*e.g.* autonomous driving), such a bias can lead to fatal system failures if a model is not able to generalize to real-world scenarios. The likelihood of a collision can be significantly increased if, for example, an autonomous car uses only the behavior of other traffic participants to derive the state of a traffic light. Using insights from visual explanation methods, a domain expert can identify and assess the severity of a bias and its impact on the real world to derive countermeasures (*e.g.* record additional data to increase the diversity of the dataset).

In this section, we investigate two common data biases: the coexistence of objects as well as the bias of certain objects towards specific colors.

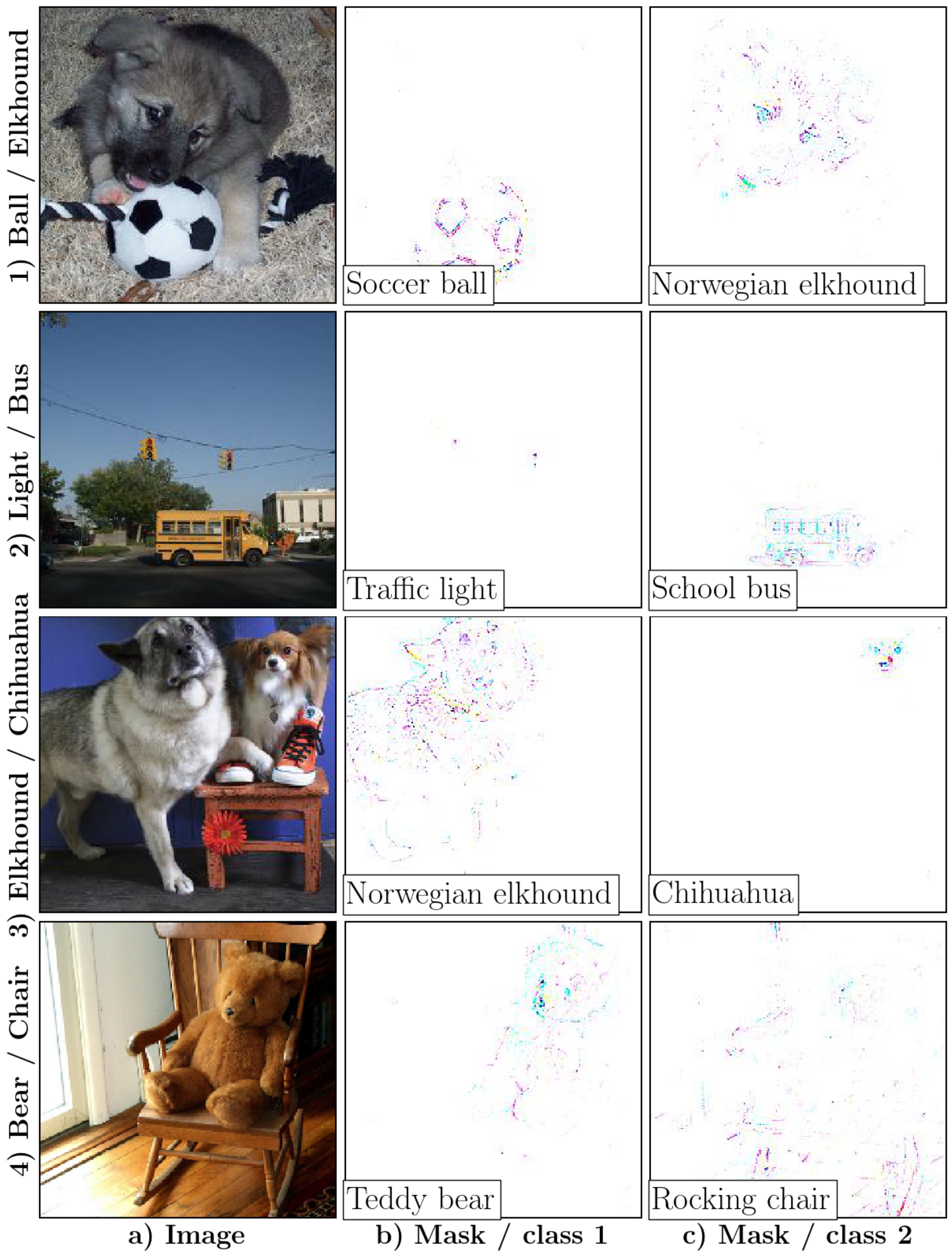


Figure 6.10: Visual explanation masks computed for images with multiple objects from distinct classes. For each image we generate two masks using the *deletion game* and *GoogleNet*. The corresponding target class is depicted in the lower left corner of each mask. The input images are from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. FGVis produces class-discriminative explanations, even when objects partially overlap.

(Modified from Wagner *et al.* (2019) / © 2019 IEEE)

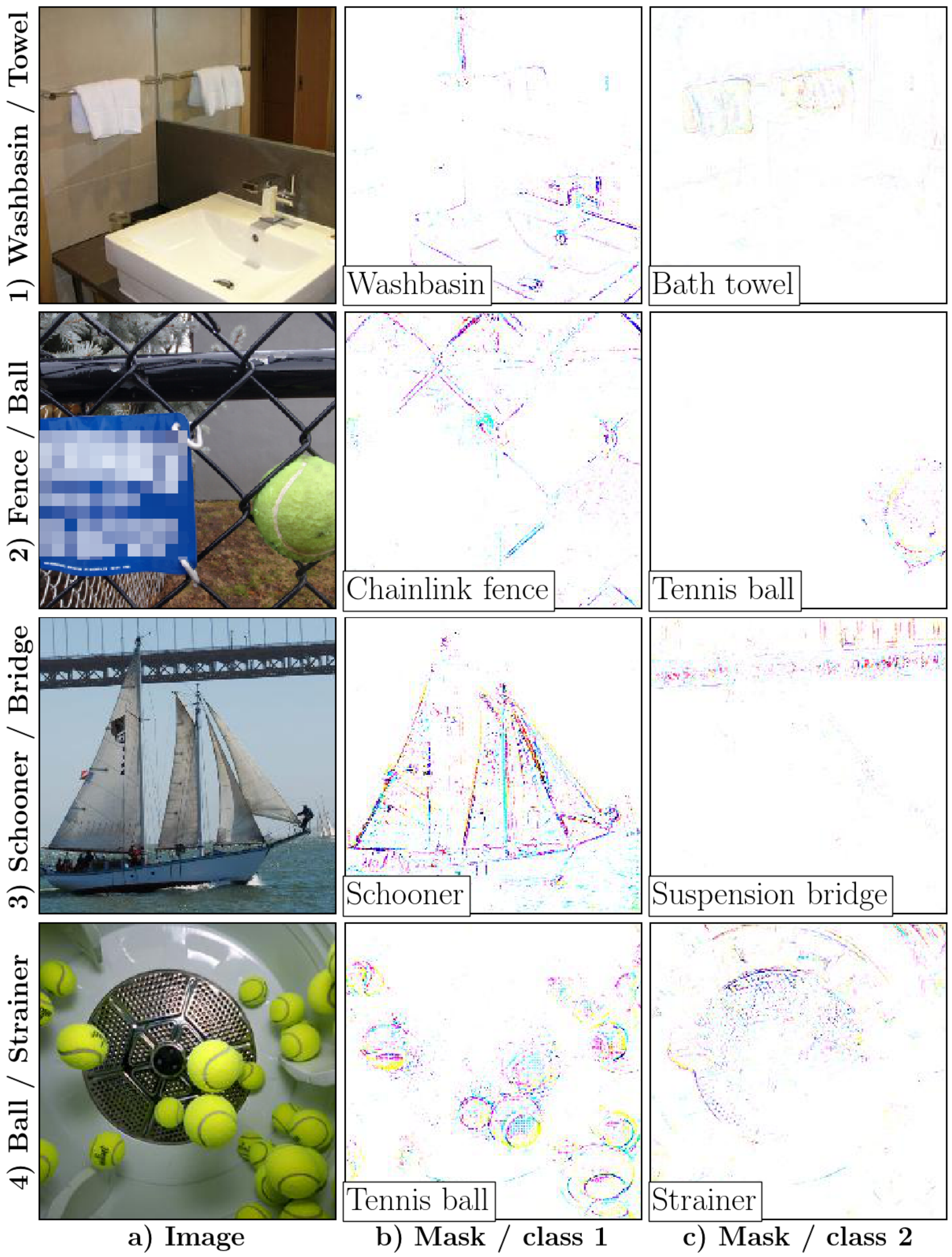


Figure 6.11: Visual explanation masks computed for images with multiple objects from distinct classes. For each image we generate two masks using the *deletion game* and *GoogleNet*. The corresponding target class is depicted in the lower left corner of each mask. The input images are from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. FGVis produces class-discriminative explanations, even when objects partially overlap.
(Modified from Wagner *et al.* (2019) / © 2019 IEEE)

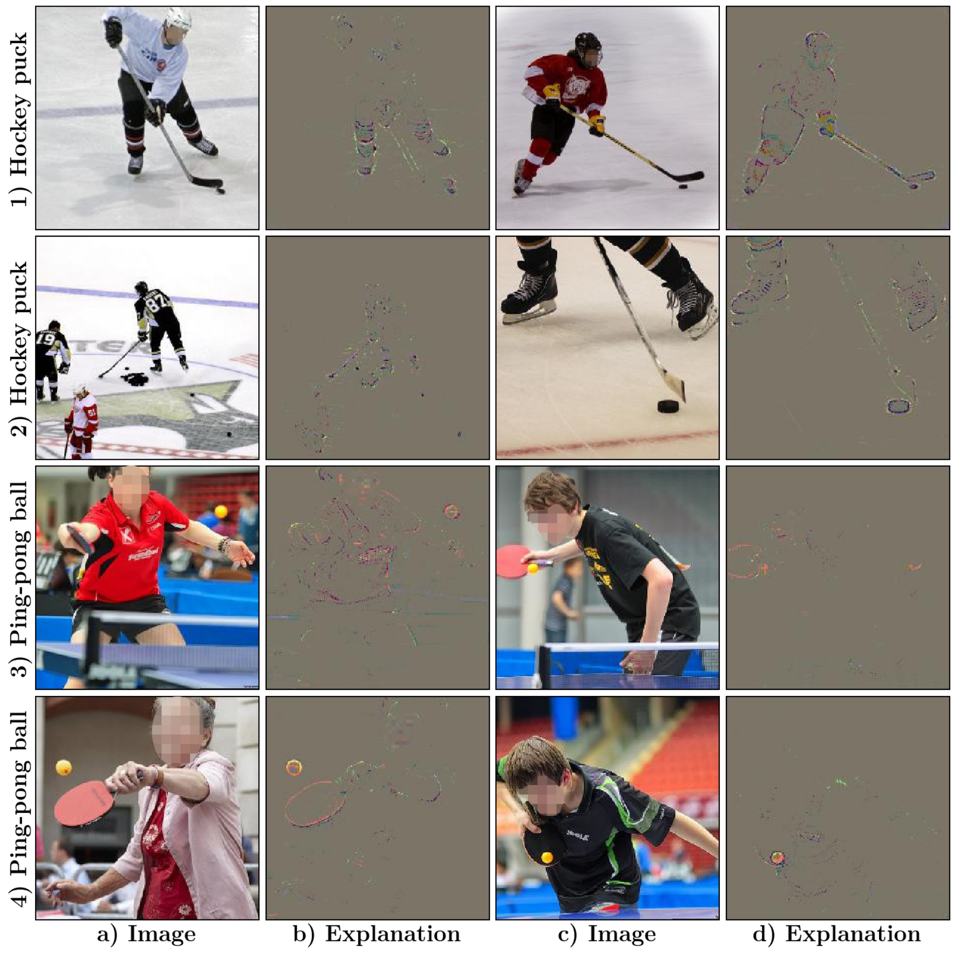


Figure 6.12: Visual explanations computed using the *deletion game* for *GoogleNet*. For both classes (*hockey puck* and *ping-pong ball*) our visual explanation method additionally has to delete pixels of the players and the table tennis bat / ice-hockey stick to shift the prediction of the *GoogleNet* model. The input images are from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

Coexistence of Objects

The coexistence of objects often leads to models learning a bias. In Figure 6.12, we depict such a bias for *GoogleNet* trained on *ImageNet*.

Sports equipment like *hockey pucks* or *ping-pong balls* frequently appear in combination with players in images. This bias is learned by the neural network and results in visual explanations that also contain pixels belonging to the players. Without deleting these pixels, the *deletion game* is not able to shift the class of the evaluated images. In addition to the player, the table tennis bat / ice-hockey stick seems to also contribute to the prediction of the *GoogleNet* model.

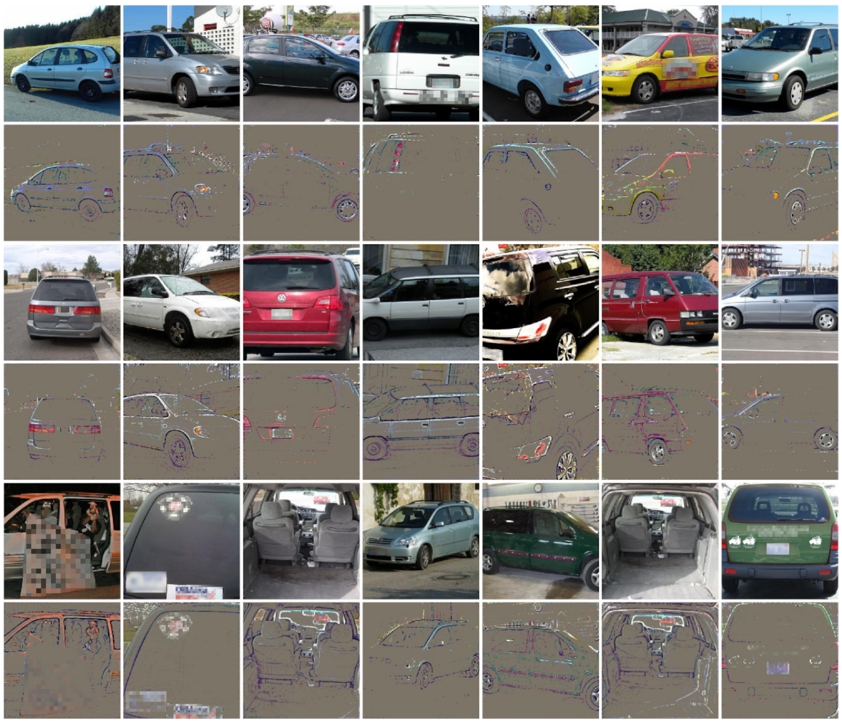


Figure 6.13: Explanations of *ImageNet* (Russakovsky *et al.*, 2015) validation images containing *minivans* computed using the *preservation game* for *VGG16*. Explanations of class *minivan* focus on edges, hardly preserving coloured areas. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

Color Bias

Objects are often biased towards specific colors. FGVis generates color dependent explanations and thus provides a first visual indication for the importance of different color channels.

We investigate if a *VGG16* model trained on *ImageNet* shows such a bias using the *preservation game*. We focus on images of *school buses* and *minivans* and depict visual explanations for all correctly classified images of the *ImageNet* validation dataset in Figure 6.13 and Figure 6.14, respectively. Visual explanations of *minivans* mainly focus on edges, not consistently preserving the original color of the cars. Especially for white or grey *minivans* the color visible in the explanation is primarily reduced to a greenish-blue color. The explanations of *school buses* are strongly dominated by the color yellow, matching the main color of *school buses* in the *ImageNet* training dataset. Compared to *minivans*, explanations of *school buses* do not mainly focus on edges and also contain a relatively large amount of yellow surface areas. These observations are a first indication for the importance of color for the prediction of *school buses*.

To verify the qualitative finding, we quantitatively estimate the color bias in Table 6.5. For all correctly classified images of the *ImageNet* validation dataset, we swap each of the three color channels *BGR* to either *RBG* or *GRB* and calculate the ratio of maintained correct classifications. For class *minivan* 83.3% of the 21 correctly classified images keep their class label, for class *school bus* only 8.3% of 42 images. The two reported metrics result from averaging the accuracies of both color representations *RBG* and *GRB* (see Table 6.5). The quantitative results thus support our qualitative findings for class *school bus*. For 80 *ImageNet* classes at least 75% of images are no longer correctly classified after the color swap. We show the results for the most and least affected 19 classes as well as the classes *minivan* and *school bus* in Table 6.5.

To the best of our knowledge, FGVis is the first method used to highlight color channel importance.

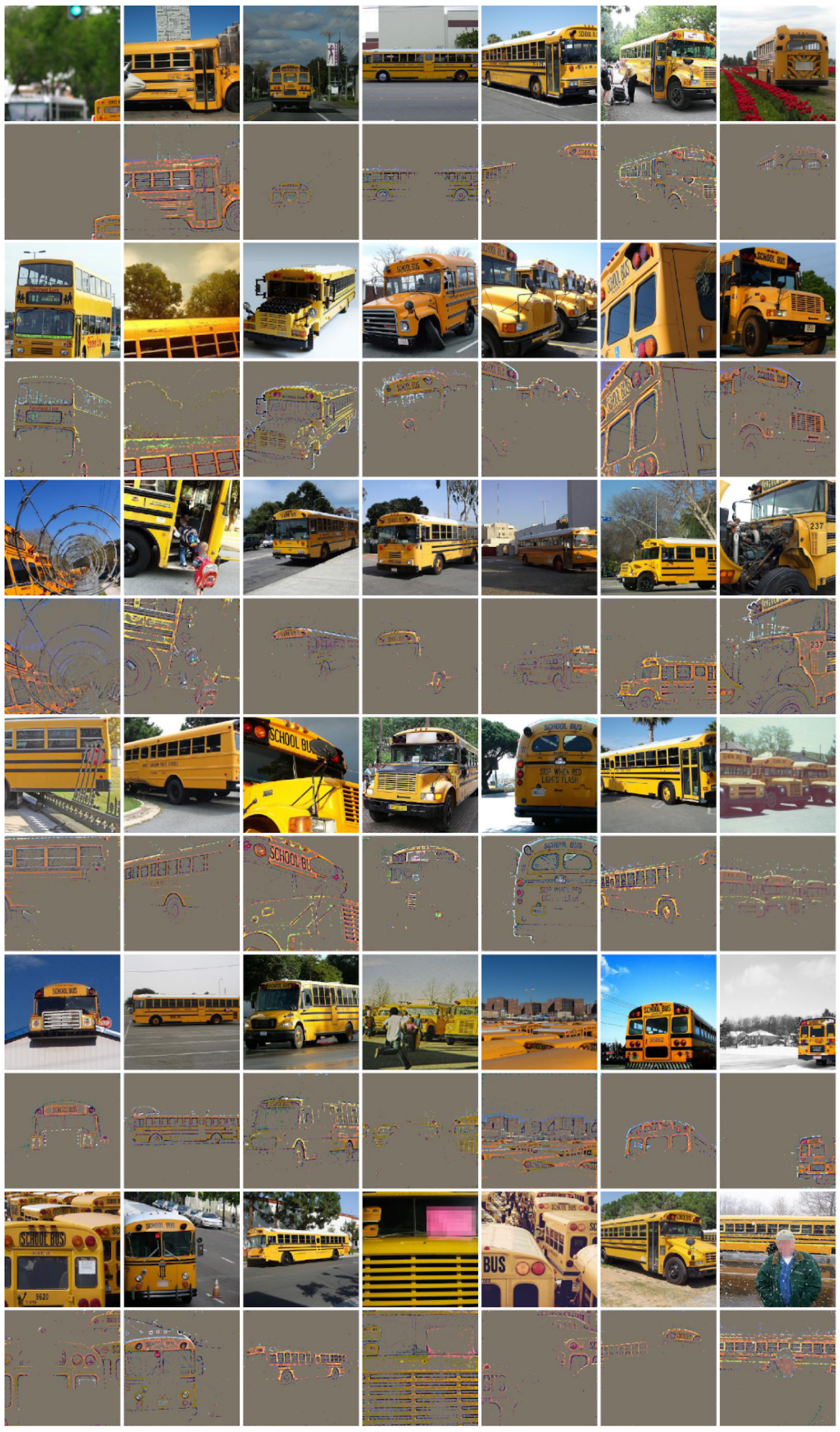


Figure 6.14: Explanations of *ImageNet* (Russakovsky *et al.*, 2015) validation images containing *school buses* computed using the *preservation game* for *VGG16*. Explanations of class *school bus* are dominated by the color yellow.
(Modified from Wagner *et al.* (2019) / © 2019 IEEE)

ID	Class name	#Images	Avg(RBG, GRB)	RBG	GRB
168	redbone	31	0.00 %	0.00 %	0.00 %
964	potpie	28	0.00 %	0.00 %	0.00 %
159	Rhodesian ridgeback	35	0.00 %	0.00 %	0.00 %
930	French loaf	27	0.00 %	0.00 %	0.00 %
234	Rottweiler	42	1.19 %	0.00 %	2.38 %
214	Gordon setter	36	1.39 %	2.78 %	0.00 %
963	pizza, pizza pie	35	1.43 %	2.86 %	0.00 %
950	orange	35	1.43 %	2.86 %	0.00 %
184	Irish terrier	33	1.52 %	0.00 %	3.03 %
962	meat loaf, meatloaf	29	1.72 %	3.45 %	0.00 %
984	rapeseed	47	2.13 %	4.26 %	0.00 %
211	vizsla, Hungarian pointer	35	2.86 %	2.86 %	2.86 %
11	goldfinch, Carduelis carduelis	48	3.12 %	0.00 %	6.25 %
934	hotdog, hot dog, red hot	40	3.75 %	2.50 %	5.00 %
218	Welsh springer spaniel	39	3.85 %	2.56 %	5.13 %
191	Airedale, Airedale terrier	37	5.41 %	5.41 %	5.41 %
163	bloodhound, sleuthhound	18	5.56 %	5.56 %	5.56 %
961	dough	15	6.67 %	0.00 %	13.33 %
263	Pembroke, Pembroke Welsh corgi	41	7.32 %	7.32 %	7.32 %
...	...	...	...	...	...
779	school bus	42	8.33 %	9.52 %	7.14 %
...	...	...	...	...	...
656	minivan	21	83.33 %	71.43 %	95.24 %
...	...	...	...	...	...
528	dial telephone, dial phone	36	95.83 %	91.67 %	100.00 %
866	tractor	37	95.95 %	91.89 %	100.00 %
572	goblet	26	96.15 %	96.15 %	96.15 %
47	African chameleon, Chamaeleo chamaeleon	40	96.25 %	95.00 %	97.50 %
302	ground beetle, carabid beetle	27	96.30 %	96.30 %	96.30 %
463	bucket, pail	27	96.30 %	96.30 %	96.30 %
717	pickup, pickup truck	28	96.43 %	100.00 %	92.86 %
178	Weimaraner	44	96.59 %	93.18 %	100.00 %
669	mosquito net	44	96.59 %	97.73 %	95.45 %
661	Model T	46	96.74 %	97.83 %	95.65 %
769	rule, ruler	36	97.22 %	100.00 %	94.44 %
771	safe	40	97.50 %	97.50 %	97.50 %
829	streetcar, tram, tramcar, trolley, ...	41	97.56 %	97.56 %	97.56 %
713	photocopier	44	97.73 %	100.00 %	95.45 %
916	web site, website, internet site, site	47	97.87 %	100.00 %	95.74 %
423	barber chair	31	98.39 %	96.77 %	100.00 %
190	Sealyham terrier, Sealyham	39	98.72 %	97.44 %	100.00 %
340	zebra	47	100.00 %	100.00 %	100.00 %
545	electric fan, blower	37	100.00 %	100.00 %	100.00 %

Table 6.5: Ratio of maintained correct classifications on the validation set of *ImageNet* after swapping color channels *BGR* to either *RBG* or *GRB*. The most and least affected 19 classes as well as the classes *minivan* and *school bus* are included in the table. The class ID, class name, number of correctly classified images before the color swap (#Images), and percentage of maintained correct classifications after the swap are reported. (Wagner *et al.* (2019) / © 2019 IEEE)

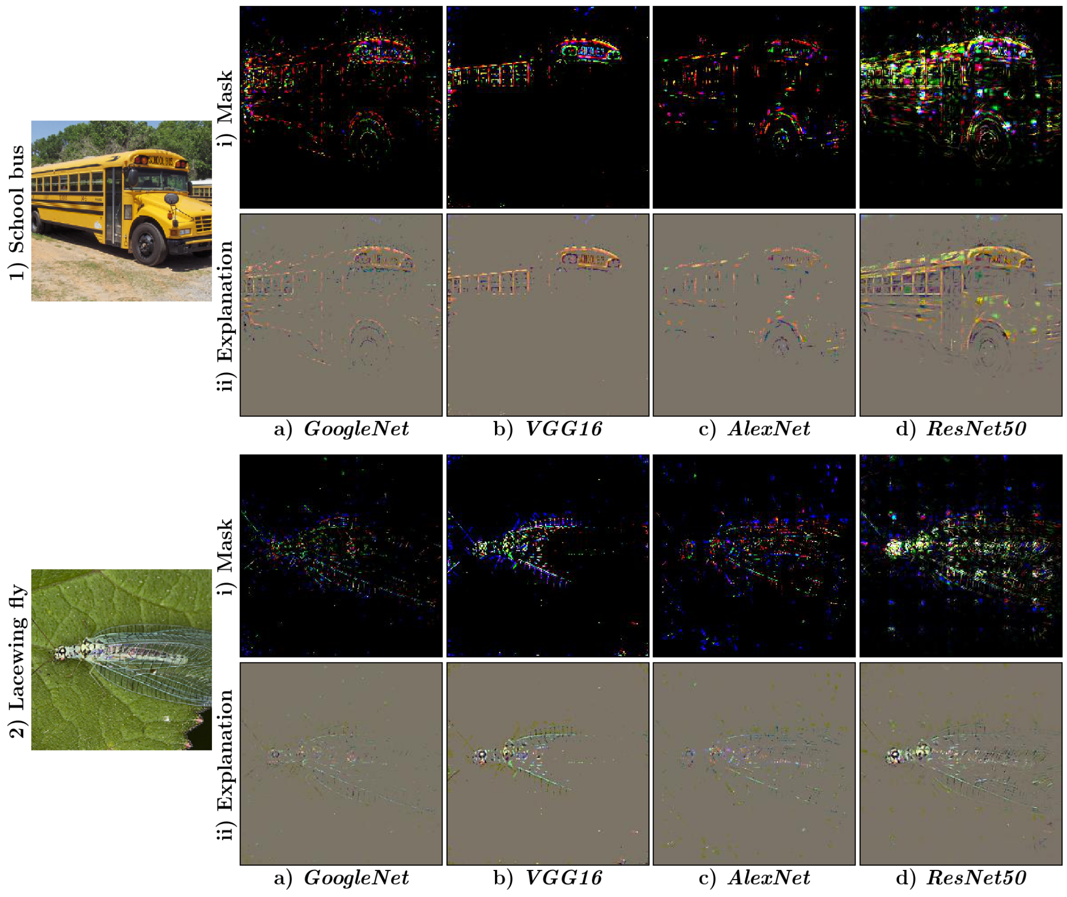


Figure 6.15: Visual explanations and masks computed using the *preservation game* for different network architectures. The images are from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

6.5.7 Comparison of Models

In Figure 6.15 and Figure 6.16, we compare the evidence on which different network architectures base their prediction using the *preservation* and *deletion game*. Our evaluation examines four architectures: *AlexNet* (Krizhevsky *et al.*, 2012), *GoogleNet* (Szegedy *et al.*, 2015), *ResNet50* (He *et al.*, 2016a), and *VGG16* (Simonyan and Zisserman, 2015). All four architectures are trained on the *ImageNet* training dataset and the images used for comparison are taken from the *ImageNet* validation dataset.

Visual explanations of the *ResNet50* architecture are more dense, meaning more pixels have to be deleted / preserved to change / retain the prediction of the model. The explanations of *AlexNet* and *GoogleNet* primarily contain fine structures and are not so

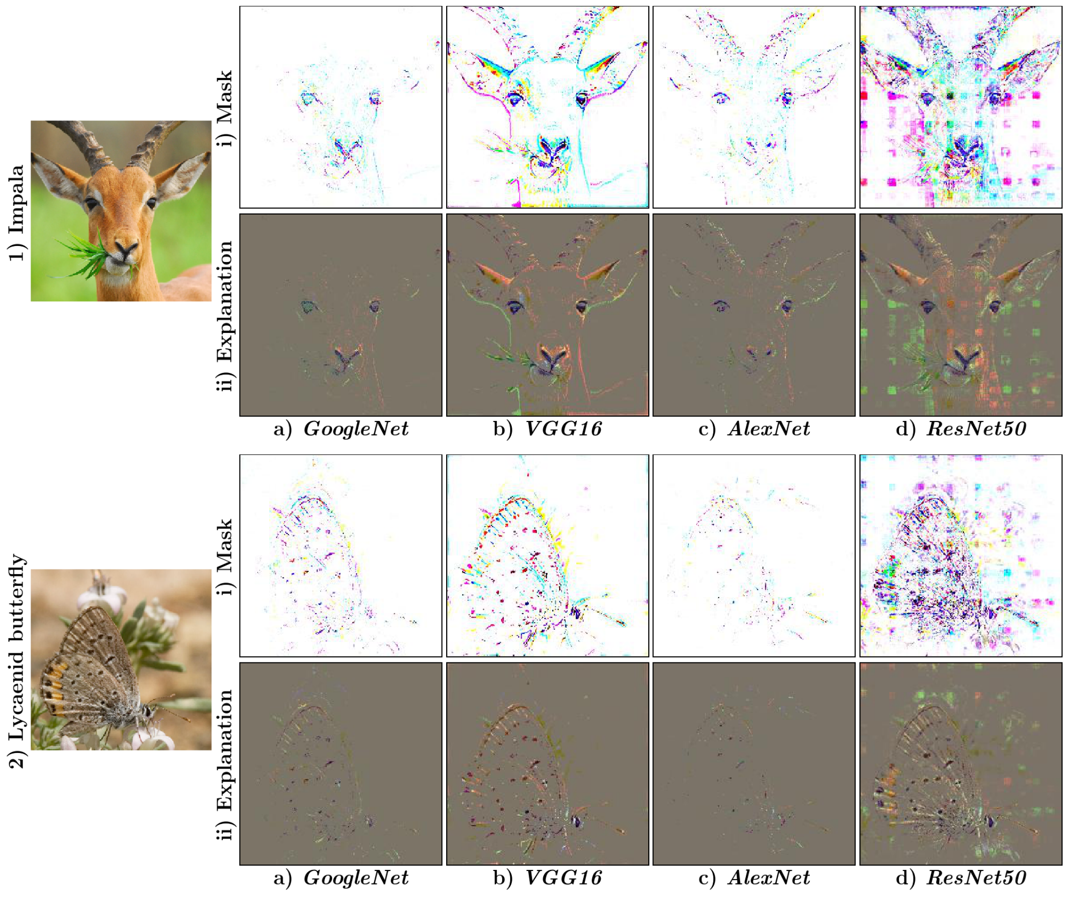
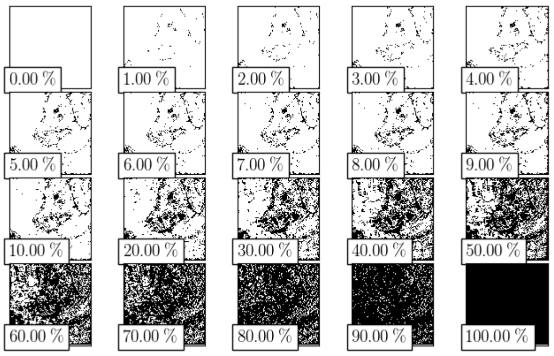


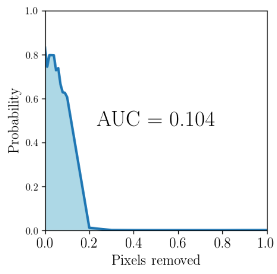
Figure 6.16: Visual explanations and masks computed using the *deletion game* for different network architectures. The images are from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

pronounced compared to *VGG16* and *ResNet50*. This is particularly visible in the explanations of the *impala* in Figure 6.16 / 1. *VGG16* has more pronounced edges in the explanations compared to the other networks. In addition, pixels at the image edges are highlighted more often. Visual explanations of *ResNet50* show a grid-like pattern. Interestingly, the grid pattern is not only visible within the target object, but also to a lesser extent outside the object. A similar observation was also reported in Nie *et al.* (2018) for *ResNet50*. A detailed investigation of this phenomenon is left to future research.

(a) Input image



(b) Deletion masks: binary masks with fraction of pixels removed



(c) Deletion curve

Figure 6.17: Subset of the binary deletion masks and resulting deletion curve for one input image. The deletion curve (c) is computed by successively deleting pixels (b) from the image (a) according to their importance and measuring the resulting probability of class c_{ml}. The input image containing the *black-and-tan coonhound* is from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. (Wagner *et al.* (2019) / © 2019 IEEE)

6.5.8 Faithfulness of Explanations

A visual explanation should be faithful to the underlying neural network, meaning, an explanation should only highlight evidence which actually contributes to the prediction of a network. Petsiuk *et al.* (2018) proposed causal metrics to quantitatively compare the faithfulness of visual explanation methods. These metrics do not depend on human labels (*i.e*. they are not biased towards human perception) and are thus well suited to verify if a visual explanation correctly represents the true evidence on which a model bases its decisions.

Using the deletion metric of Petsiuk *et al.* (2018), we evaluate the faithfulness of visual explanations generated by FGVis. The deletion metric measures how the removal of evidence affects the prediction of the used model. The metric assumes that an importance map is given, which ranks all image pixels with respect to their evidence for the predicted class c_{ml} (*i.e*. the most-likely class according to the model). We use the mean mask as a proxy for the pixel-wise importance map. The mean mask is computed for all images of the *ImageNet* validation dataset using the *deletion game* with a learning rate of 0.3, running for 250 (*VGG16*) / 400 (*ResNet50*) iterations. A line-search is used to determine the parameter λ. We iteratively use four equally spaced λ values between 10^{-7} and 10^{-10} and stop when $y_e^{c_{ml}} < 0.02 \cdot y_x^{c_{ml}}$, where $y_e^{c_{ml}}$ is the softmax score of class c_{ml} given the explanation, and $y_x^{c_{ml}}$ the corresponding score given the input image.

Method	*VGG16*	*ResNet50*
Grad-Cam (Selvaraju *et al.*, 2017)	0.109	0.123
Sliding Window (Zeiler and Fergus, 2014)	0.116	0.142
LIME (Ribeiro *et al.*, 2016)	0.101	0.122
RISE (Petsiuk *et al.*, 2018)	0.098	0.108
FGVis (ours)	**0.064**	**0.064**

Table 6.6: Deletion metric of two model architectures computed on the *ImageNet* validation dataset (lower is better). The results for all reference methods are from Petsiuk *et al.* (2018). FGVis outperforms the other explanation methods by a large margin. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

Using the importance map, the deletion curve is generated by successively removing pixels from the input image according to their importance and measuring the resulting probability of class c_{ml}. The removed pixels are set to zero, as proposed by Petsiuk *et al.* (2018). In Figure 6.17 / b, we illustrate a subset of the binary deletion masks for an input image. The deletion metric is computed by measuring the *Area Under the Curve* (AUC) of the deletion curve (Figure 6.17 / c).

In Table 6.6, we report the resulting deletion metric of FGVis, computed on the validation split of *ImageNet* using the *VGG16* and *ResNet50* model. FGVis outperforms the other visual explanation methods on both models by a large margin. This performance increase can most likely be attributed to the ability of FGVis to visualize fine-grained evidence. All other approaches are limited to coarse explanations, either due to computational constraints or due to the used measures to avoid adversarial evidence.

6.5.9 Further Examples

In Figure 6.18 and Figure 6.19, we depict further explanation masks computed using our proposed FGVis method. The masks in Figure 6.18 are computed using the *repression game* for *VGG16*, whereas the ones in Figure 6.19 are computed using the *preservation game* for *ResNet50*.

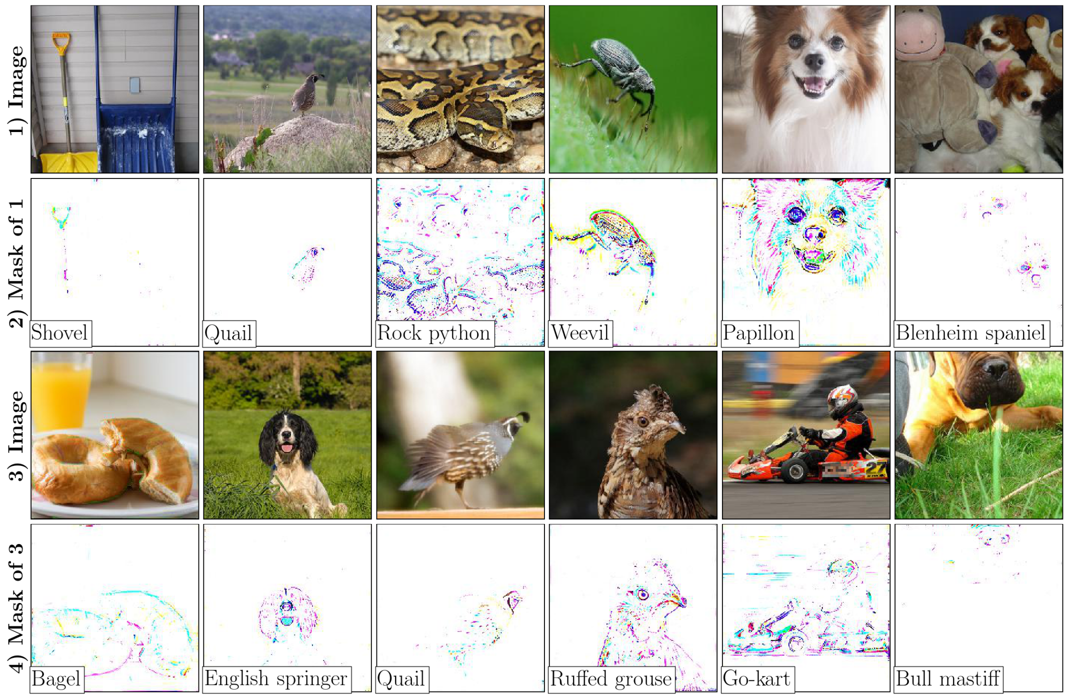


Figure 6.18: Visual explanation masks computed using the *repression game* for *VGG16*. The target class is noted in the lower left corner of each explanation mask. The images are from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

6.6 Conclusion

In the last decade, deep learning methods led to significant advances in a wide range of disciplines, including computer vision (Krizhevsky *et al.*, 2012; Girshick *et al.*, 2014; Long *et al.*, 2015), natural language processing (Torfi *et al.*, 2020), and behavior prediction (Rudenko *et al.*, 2020). A disadvantage of many deep learning models is their low level of transparency and interpretability. Particularly in safety-critical applications, such as self-driving cars, it is advantageous to have insights into the decision-making process of a model. Developers can use these insights to debug and validate models and they can increase the user's trust in the technology (McAllister *et al.*, 2017). In the case of a serious error, authorities may require a review and explanation of model decisions which would benefit from additional knowledge of a model's decision-making process. One can increase the transparency of a model in the design phase by, for example, incorporating intermediate representations which are human-interpretable. In the previous chapter, we adopted this idea and designed an interpretable representation filter that stabilizes the

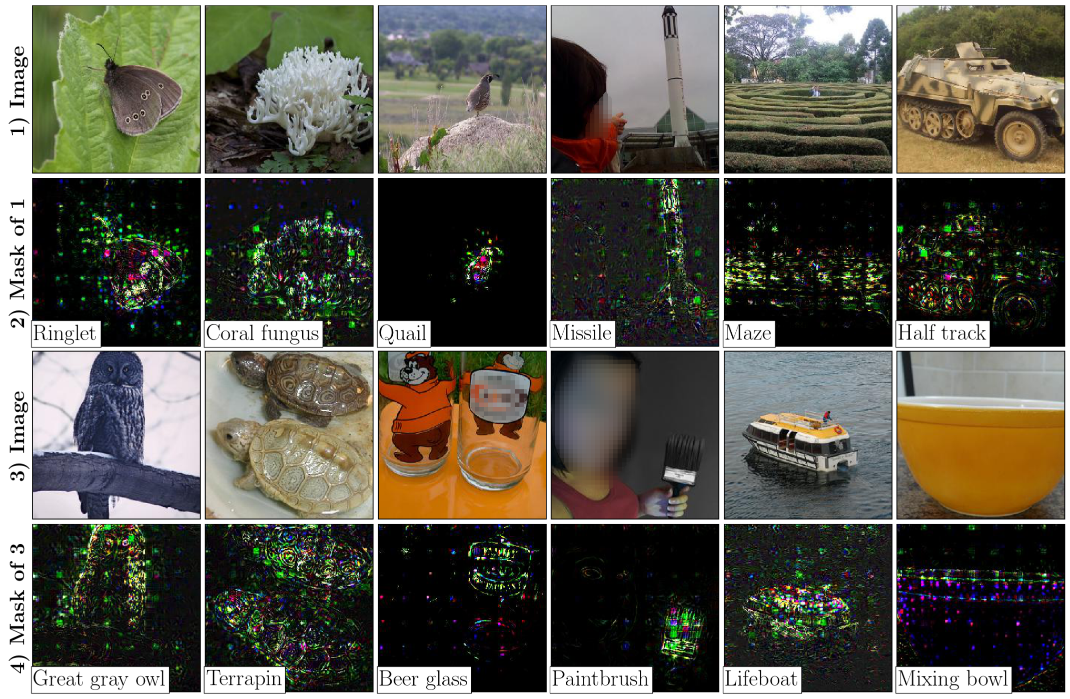


Figure 6.19: Visual explanation masks computed using the *preservation game* for *ResNet50*. The target class is noted in the lower left corner of each explanation mask. The images are from the *ImageNet* (Russakovsky *et al.*, 2015) validation dataset. (Modified from Wagner *et al.* (2019) / © 2019 IEEE)

prediction of a semantic segmentation network. If a black-box model is given, post-hoc explanation methods can be used to inspect its behavior.

In this chapter, we proposed a post-hoc, optimization-based explanation method (FGVis), which generates fine-grained and class-discriminative visual explanations in the image space. To prevent the generation of adversarial evidence (*i.e.* faulty evidence due to artifacts), our method uses a novel adversarial defense which filters gradients during optimization. Compared to other adversarial defense techniques, ours does not depend on human-tuned hyperparameters and imposes no further constraints (*e.g.* a reduced resolution) on the explanation. The visual explanations created by FGVis are intuitively interpretable, well suited for visualizing detailed evidence, and can be tested as they are valid model inputs. Additionally, they preserve the characteristics of images, like edges and colors, enabling richer insights. We showed the effectiveness of our defense technique using different model architectures, compared our visual explanations to other methods, and quantitatively evaluated their faithfulness. Moreover, we underlined the strength in producing class-discriminative visualizations and pointed to characteristics in explanations of the *ResNet50* architecture.

After the publication of the content of this chapter, a variety of scientific papers have been published on the topic of *Explainable Artificial Intelligence* (XAI) (Arrieta *et al.*, 2020; Zablocki *et al.*, 2021; Linardatos *et al.*, 2021; Samek *et al.*, 2021; Fan *et al.*, 2021). Numerous works extended existing post-hoc, local explanation methods (Fong *et al.*, 2019; Kim *et al.*, 2019; Qi *et al.*, 2019; Wang *et al.*, 2020a) or developed novel local approaches (Goyal *et al.*, 2019; Schulz *et al.*, 2020).

Fong *et al.* (2019) improve their optimization-based explanation method introduced in Fong and Vedaldi (2017) by replacing two terms of the objective function with corresponding optimization constraints — an *area constraint* that fixes the sparsity of the explanation mask to a given value as well as a *smoothness constraint*. The proposed changes increase the usability of the explanation method by removing hyperparameters. In addition, they extend their method to support the investigation of intermediate network representations. The *area constraint* of Fong *et al.* (2019) could be used as an alternative to the sparsity loss term (see Equation 6.3) in our visual explanation method FGVis. Khakzar *et al.* (2020) propose to improve gradient-based attribution methods via input-specific network pruning (Khakzar *et al.*, 2021). They intensively evaluate and verify their approach *PruneGrad* using sanity checks (Adebayo *et al.*, 2018b) and evaluation metrics (Hooker *et al.*, 2019). In addition, they highlight a connection between pruning of neurons and our proposed adversarial defense (Khakzar *et al.*, 2020). Other methods like *Score-CAM* (Wang *et al.*, 2020a) or *Smooth Grad-CAM++* (Omeiza *et al.*, 2019) can be seen as extensions of Grad-CAM (Selvaraju *et al.*, 2017). *Score-CAM* replaces the gradient-based weighting of activation maps used in Grad-CAM with a weighting approach relying on a *Channel-Wise Increase of Confidence* (CIC) score (Wang *et al.*, 2020a). Their experiments show a superior performance of *Score-CAM* in localization and recognition metrics compared to other CAM-based approaches. Omeiza *et al.* (2019) propose to integrate the gradient averaging technique of *SmoothGrad* (Smilkov *et al.*, 2017) into *Grad-CAM++* (Chattopadhay *et al.*, 2018) to improve the resulting visual explanations in terms of sharpness and object localization. Compared to our method, these two approaches are not well suited for visualizing fine-grained evidence. A more detailed discussion of the recent XAI literature, which covers local explanation methods, global methods, as well as other aspects (*e.g.* model-agnostic vs. model-specific methods, approaches for other data types and machine learning models) can be found in Arrieta *et al.* (2020), Samek *et al.* (2021), and Linardatos *et al.* (2021).

Explanation methods are also being increasingly used in the field of autonomous driving, due to the safety-critical nature of such systems. Bojarski *et al.* (2018) propose a post-hoc explanation method called *VisualBackProp* and use it to examine *PilotNet* (Bojarski *et al.*, 2016), a CNN which predicts the steering commands of a vehicle given the image of a front-facing camera. Their qualitative results show that the model bases its decisions on visual cues which are consistent with those a human driver would use, *e.g*. pavement-grass boundaries, parked cars (Bojarski *et al.*, 2017; Bojarski *et al.*, 2020). Liu *et al.* (2020) extend the perturbation-based explanation method of Fong and Vedaldi (2017) to spatio-temporal models and inspect different architectures for scene-based driving behavior classification. Sauer *et al.* (2018) propose a neural network that predicts a human-interpretable, abstract representation of an autonomous vehicle's environment based on data from a camera. To debug the behavior of their model, they use the explanation method *Gradient-Weighted Class Activation Mapping* (Grad-CAM). A more detailed evaluation of explainability in the field of vision-based autonomous driving is presented by Zablocki *et al.* (2021).

We believe that explanation methods are a valuable tool to accelerate the development and deployment of deep learning models in real-world applications. However, there are still many limitations and challenges to be addressed, such as robust as well as consistent metrics / concepts for evaluating methods, deeper theoretical understanding of XAI approaches, and more intuitive concepts for communicating model behavior.

7 Conclusion

Due to their promise to disrupt the transportation sector, autonomous vehicles have attracted great attention in both academic and industrial research. Such systems are expected to increase road safety, improve quality of life, reduce resource consumption, and thus change society in a lasting way. However, to achieve large-scale deployment in the real world without human supervision, there are still many technical, social, and legal challenges to overcome.

A key technical challenge is the robust perception and interpretation of the environment. Perception systems must exhibit a high degree of reliability under all possible environment conditions and traffic situations. In addition, they must cope with the large variability of the scene, sensor interference, as well as short sensor failures. In this thesis, we particularly focused on reducing aleatoric failures, *i.e.* failures which cannot be eliminated by using a more advanced perception model or additional training data. To solve such failures, one has to enhance the information provided to the perception system. We investigated two such approaches: sensor data fusion and temporal integration of information.

Low or rapidly changing lighting conditions as well as bad weather conditions are a challenge for pedestrian detectors operating on images of the visible spectrum. To increase the robustness of such detectors, we proposed to fuse the information of a visible and a thermal camera using deep learning-based models. An early- and a late-fusion CNN were investigated for fusion. Building upon the R-CNN detection framework, we created a multispectral pedestrian detector and analyzed its performance on the *KAIST* benchmark (Hwang *et al.*, 2015). Our late-fusion-based detector with additional pre-training on visible images outperformed prior detectors on the *KAIST* benchmark significantly.

To the best of our knowledge, this is the first work to use a deep learning-based approach for multispectral pedestrian detection.

A complementary approach to sensor data fusion is the temporal filtering of information. The filter task can be divided into two subtasks: a prediction subtask and an update subtask. In a first step, we focused on the prediction aspect and proposed an approach to create semantic forecasting models. Our approach is based on a predictive transformation, which converts non-predictive feed-forward models into predictive ones. The transformation relies on a recurrent predictive module as well as a teacher-student training strategy. Training was conducted in a self-supervised manner, eliminating the need for costly labeled data. Our experiments showed the ability of the transformed architectures to model the dynamics of the scene (*e.g.* the interaction and motion of objects), enabling meaningful predictions of future environment states.

Building on the knowledge gained from our predictive transformation approach, we designed a parameter efficient temporal filtering concept for FC-DenseNet. The resulting temporal model RFC-DenseNet recurrently filters feature representations on all abstraction levels in a hierarchical manner, while conceptually decoupling scene representation from temporal dependencies. Compared to other single-image baselines as well as other temporal architectures, our model showed improved performance, especially in challenging scenarios.

As a second field of research, we examined two approaches to increase the transparency and interpretability of deep learning-based models: *interpretability by design* and post-hoc explanation methods. Such approaches are particularly important for models used in safety-critical applications. Insights into the decision-making process of a model facilitate debugging, they help to better understand shortcomings as well as limitations of a model, and enable an easy introduction of prior knowledge. A high-level of transparency and interpretability may also play an important role in solving some of the social and legal challenges of future autonomous vehicles. Users tend to trust a model more if they are able to anticipate and verify results. Thus, explanations tailored to end users can contribute to building trust, leading to a broader social acceptance of the technology. Transparency and interpretability of models facilitate validation, certification, and failure analysis, which may be required by legal authorities.

In a subsequent chapter, we again used a recurrent filter to increase the robustness of a segmentation model. Compared to the filters used to create RFC-DenseNet, we designed this filter to be more transparent and interpretable. The filter consists of multiple submodules, which can be debugged, pre-trained, and tested independently. Human interpretable intermediate representations were introduced to offer insights into the behavior of the filter. In addition, we used these representations to include auxiliary training losses. By replacing individual submodules with physical models, we were able to integrate domain knowledge into the filter. Our experiments especially highlighted the advantages of our interpretable and transparent filter design and showed the ability of the resulting segmentation model to cope with missing information.

Finally, we proposed FGVis, a post-hoc, local explanation method. FGVis creates visual explanations in the image space by optimizing for a perturbed version of the input image in which all irrelevant or most relevant pixels are removed. To prevent the generation of adversarial evidence (*i.e.* faulty evidence due to artifacts), we introduced a novel adversarial defense which filters gradients during optimization. The defense imposes no constraints on explanations (*e.g.* a reduced resolution) and does not depend on human-tuned hyperparameters. Explanations created by FGVis are fine-grained, class-discriminative, and preserve the characteristics of images, such as edges and colors. Additionally, they are valid model inputs and can thus be tested. We believe that visual explanation methods, such as FGVis, are a valuable tool to accelerate the development and deployment of black-box models.

The methods and models of this thesis contribute to achieving an unrestricted and large-scale deployment of autonomous vehicles in the real world without human supervision. In the following, we outline outstanding challenges and potential future research directions.

To increase the robustness of perception systems, we investigated two complementary approaches, which are especially beneficial for challenging scenarios: sensor data fusion (Chapter 2) and temporal integration of information (Chapter 4 and 5). An autonomous vehicle designed with a safety-first principle (Mueller *et al.*, 2020) should employ both measures concurrently to ensure a safe functioning in all possible situations. Further complementary measures are the integration of context information (*e.g.* map data) as well as the exchange of data with other road users and the infrastructure (*i.e.* *vehicle-to-everything* (V2X) communication (Hobert *et al.*, 2015; Wang *et al.*, 2020b)). The joint integration of all data streams in a deep learning-based perception model is a challenging and open area of research.

The models developed in this thesis were trained and evaluated using classical perception losses and metrics independent of the autonomous vehicle's overall architecture. Such a separate consideration of the perception component facilitates development, but does not guarantee optimal performance with respect to the final task of autonomous driving (Herman *et al.*, 2021; Casas *et al.*, 2020). Thus, a holistic development approach with system-level model training and evaluation is beneficial. One approach are end-to-end models which directly predict the steering commands of the autonomous vehicle from raw sensory data (Bojarski *et al.*, 2016; Tampuu *et al.*, 2020). However, such models have a low level of transparency and interpretability, resulting in the disadvantages already mentioned. The development of end-to-end trainable models with a modular structure and human interpretable intermediate representations is another interesting area of research. A promising concept for the development of such a model is proposed by Casas *et al.* (2021).

As discussed in Chapter 6, explanation methods are a useful tool for accelerating the development and deployment of deep learning-based models. However, there are still many limitations and challenges to address, such as reliable as well as consistent concepts and metrics for evaluating methods, deeper theoretical understanding of XAI approaches, and more intuitive concepts for communicating model behavior. In addition, method development should be more focused on user-centric design, taking into account the target audience (*e.g.* user, developer, regulator) and the overall goal (*e.g.* building trust, failure analysis) to obtain more informative explanations (Weller, 2019).

List of Abbreviations

ABM	*Activation-Based Method*
ACF	*Aggregated Channel Features*
AUC	*Area Under the Curve*
BBM	*Backpropagation-Based Method*
BN	*Batch Normalization*
BNN	*Bayesian Neural Network*
BPTT	*Backpropagation Through Time*
CAM	*Class Activation Mapping*
CIC	*Channel-Wise Increase of Confidence*
CNN	*Convolutional Neural Network*
CRF	*Conditional Random Field*
DB	*Dense Block*
DenseNet	*Densely Connected Convolutional Network*
DSSD	*Deconvolutional Single Shot Detector*
DU	*Dense Unit*
FC-DenseNet	*Fully Convolutional DenseNet*
FCN	*Fully Convolutional Network*
FGVis	*Fine-Grained Visual Explanation Method*
FM	*Filter Module*
Grad-CAM	*Gradient-Weighted Class Activation Mapping*
GRU	*Gated Recurrent Unit*
HOG	*Histogram of Oriented Gradient*
IMM	*Interacting Multiple Model*
IoU	*Intersection over Union*

LIME	*Local Interpretable Model-Agnostic Explanations*
LSTM	*Long Short-Term Memory*
MPSPNet	*Multi-Task Pyramid Scene Parsing Network*
MSDS-RCNN	*Multispectral Simultaneous Detection and Segmentation R-CNN*
NN	*Neural Network*
PBM	*Perturbation-Based Method*
PGP	*Predictive Gating Pyramid*
PSPNet	*Pyramid Scene Parsing Network*
R-CNN	*Regions with CNN features*
rCNN	*Recurrent Convolutional Neural Network*
RDB	*Recurrent Dense Block*
ReLU	*Rectified Linear Unit*
ResNet	*Residual Network*
RFC-DenseNet	*Recurrent Fully Convolutional DenseNet*
RNN	*Recurrent Neural Network*
SGD	*Stochastic Gradient Descent*
SVM	*Support Vector Machine*
TD	*Transition Down*
TU	*Transition Up*
V2X	*vehicle-to-everything*
XAI	*Explainable Artificial Intelligence*

List of Tables

List of Figures

Bibliography

Adebayo, J., J. Gilmer, I. Goodfellow, and B. Kim (2018a). “Local Explanation Methods for Deep Neural Networks Lack Sensitivity to Parameter Values”. In: *International Conference on Learning Representations (ICLR), Workshop Track Proceedings*.

Adebayo, J., J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim (2018b). “Sanity Checks for Saliency Maps”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 31. Curran Associates, Inc.

Alahi, A., K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese (2016). “Social LSTM: Human Trajectory Prediction in Crowded Spaces”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971. DOI: 10.1109/CVPR.2016.110.

Angelova, A., A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson (2015). “Real-Time Pedestrian Detection With Deep Network Cascades”. In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 32.1–32.12. DOI: 10.5244/C.29.32.

Arrieta, A. B., N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, and F. Herrera (2020). “Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI”. In: *Information Fusion* 58, pp. 82–115. DOI: 10.1016/j.inffus.2019.12.012.

Ba, J. and R. Caruana (2014). “Do Deep Nets Really Need to be Deep?” In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 27. Curran Associates, Inc.

Bach, S., A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek (2015). “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PLoS ONE* 10.7. DOI: 10.1371/journal.pone.0130140.

Baehrens, D., T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller (2010). “How to Explain Individual Classification Decisions”. In: *Journal of Machine Learning Research* 11.61, pp. 1803–1831.

Bahram, M. (2017). “Interactive Maneuver Prediction and Planning for Highly Automated Driving Functions”. PhD thesis. Technische Universität München.

Behnke, S. (2003). *Hierarchical Neural Networks for Image Interpretation*. Vol. 2766. Lecture Notes in Computer Science. Springer. DOI: 10.1007/b11963.

Benenson, R., M. Mathias, R. Timofte, and L. Van Gool (2012). “Pedestrian Detection at 100 Frames Per Second”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2903–2910. DOI: 10.1109/CVPR.2012.6248017.

Benenson, R., M. Mathias, T. Tuytelaars, and L. Van Gool (2013). “Seeking the Strongest Rigid Detector”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3666–3673. DOI: 10.1109/CVPR.2013.470.

Benenson, R., M. Omran, J. Hosang, and B. Schiele (2014). “Ten Years of Pedestrian Detection, What Have We Learned?” In: *Computer Vision – ECCV 2014 Workshops*. Springer International Publishing, pp. 613–627. DOI: 10.1007/978-3-319-16181-5_47.

Bertozzi, M., A. Broggi, M. Felisa, G. Vezzoni, and M. Del Rose (2006). “Low-Level Pedestrian Detection by Means of Visible and Far Infra-Red Tetra-Vision”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 231–236. DOI: 10.1109/IVS.2006.1689633.

Blom, H. A. P. (1984). “An Efficient Filter for Abruptly Changing Systems”. In: *The 23rd IEEE Conference on Decision and Control*, pp. 656–658. DOI: 10.1109/CDC.1984.272089.

Bojarski, M., D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba (2016). *End to End Learning for Self-Driving Cars*. DOI: 10.48550/arXiv.1604.07316. arXiv: 1604.07316 [cs.CV].

Bojarski, M., P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller (2017). *Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car*. DOI: 10.48550/arXiv.1704.07911. arXiv: 1704.07911 [cs.CV].

Bojarski, M., A. Choromanska, K. Choromanski, B. Firner, L. J. Ackel, U. Muller, P. Yeres, and K. Zieba (2018). “VisualBackProp: Efficient Visualization of CNNs for Autonomous Driving”. In: *Proceedings of the IEEE International Conference on*

Robotics and Automation (ICRA), pp. 4701–4708. DOI: 10.1109/ICRA.2018.8461053.

Bojarski, M., C. Chen, J. Daw, A. Değirmenci, J. Deri, B. Firner, B. Flepp, S. Gogri, J. Hong, L. Jackel, Z. Jia, B. Lee, B. Liu, F. Liu, U. Muller, S. Payne, N. K. N. Prasad, A. Provodin, J. Roach, T. Rvachov, N. Tadimeti, J. van Engelen, H. Wen, E. Yang, and Z. Yang (2020). *The NVIDIA PilotNet Experiments*. DOI: 10.48550/arXiv.2010.08776. arXiv: 2010.08776 [cs.CV].

Bourdev, L. and J. Brandt (2005). "Robust Object Detection via Soft Cascade". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 236–243. DOI: 10.1109/CVPR.2005.310.

Brostow, G. J., J. Shotton, J. Fauqueur, and R. Cipolla (2008). "Segmentation and Recognition Using Structure From Motion Point Clouds". In: *Computer Vision – ECCV 2008*. Springer International Publishing, pp. 44–57. DOI: 10.1007/978-3-540-88682-2_5.

Brostow, G. J., J. Fauqueur, and R. Cipolla (2009). "Semantic Object Classes in Video: A High-Definition Ground Truth Database". In: *Pattern Recognition Letters* 30.2, pp. 88–97. DOI: 10.1016/j.patrec.2008.04.005.

Brouwer, N., H. Kloeden, and C. Stiller (2016). "Comparison and Evaluation of Pedestrian Motion Models for Vehicle Safety Systems". In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2207–2212. DOI: 10.1109/ITSC.2016.7795912.

Bucilua, C., R. Caruana, and A. Niculescu-Mizil (2006). "Model Compression". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 535–541. DOI: 10.1145/1150402.1150464.

Cai, Z., M. Saberian, and N. Vasconcelos (2015). "Learning Complexity-Aware Cascades for Deep Pedestrian Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3361–3369. DOI: 10.1109/ICCV.2015.384.

Casas, S., C. Gulino, S. Suo, and R. Urtasun (2020). "The Importance of Prior Knowledge in Precise Multimodal Prediction". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2295–2302. DOI: 10.1109/IROS45743.2020.9341199.

Casas, S., A. Sadat, and R. Urtasun (2021). "MP3: A Unified Model to Map, Perceive, Predict and Plan". In: *Proceedings of the IEEE/CVF Conference on Computer Vi-*

sion and Pattern Recognition (CVPR), pp. 14398–14407. DOI: 10.1109/CVPR46437.2021.01417.

Chang, C.-H., E. Creager, A. Goldenberg, and D. Duvenaud (2019). “Explaining Image Classifiers by Counterfactual Generation”. In: *International Conference on Learning Representations (ICLR)*.

Chattopadhay, A., A. Sarkar, P. Howlader, and V. N. Balasubramanian (2018). “Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 839–847. DOI: 10.1109/WACV.2018.00097.

Chen, L., G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille (2015). “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs”. In: *International Conference on Learning Representations (ICLR)*.

Chiu, H.-K., E. Adeli, and J. C. Niebles (2020). “Segmenting the Future”. In: *IEEE Robotics and Automation Letters* 5.3, pp. 4202–4209. DOI: 10.1109/LRA.2020.2992184.

Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014). “Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Choi, E.-J. and D.-J. Park (2010). “Human Detection Using Image Fusion of Thermal and Visible Image With New Joint Bilateral Filter”. In: *5th International Conference on Computer Sciences and Convergence Information Technology*, pp. 882–885. DOI: 10.1109/ICCIT.2010.5711182.

Conaire, C. Ó., E. Cooke, N. E. O’Connor, N. Murphy, and A. F. Smeaton (2005). “Background Modelling in Infrared and Visible Spectrum Video for People Tracking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. DOI: 10.1109/CVPR.2005.419.

Cordts, M., M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele (2016). “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3213–3223. DOI: 10.1109/CVPR.2016.350.

Couprie, C., P. Luc, and J. Verbeek (2019). “Joint Future Semantic and Instance Segmentation Prediction”. In: *Computer Vision – ECCV 2018 Workshops*. Springer International Publishing, pp. 154–168. DOI: 10.1007/978-3-030-11015-4_14.

Dabkowski, P. and Y. Gal (2017). “Real Time Image Saliency for Black Box Classifiers”. In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 30. Curran Associates, Inc.

Dalal, N. and B. Triggs (2005). “Histograms of Oriented Gradients for Human Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 886–893. DOI: 10.1109/CVPR.2005.177.

Dieleman, S., J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, D. M. de Almeida, B. McFee, H. Weideman, G. Takács, P. de Rivaz, J. Crall, G. Sanders, K. Rasul, C. Liu, G. French, and J. Degrave (2015). *Lasagne: First Release.* DOI: 10.5281/zenodo.27878.

Dollár, P., Z. Tu, P. Perona, and S. Belongie (2009a). “Integral Channel Features”. In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 91.1–91.11.

Dollár, P., C. Wojek, B. Schiele, and P. Perona (2009b). “Pedestrian Detection: A Benchmark”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 304–311. DOI: 10.1109/CVPR.2009.5206631.

Dollár, P., S. Belongie, and P. Perona (2010). “The Fastest Pedestrian Detector in the West”. In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 68.1–68.11. DOI: 10.5244/C.24.68.

Dollár, P., C. Wojek, B. Schiele, and P. Perona (2012). “Pedestrian Detection: An Evaluation of the State of the Art”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4, pp. 743–761. DOI: 10.1109/TPAMI.2011.155.

Dollár, P., R. Appel, S. Belongie, and P. Perona (2014). “Fast Feature Pyramids for Object Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.8, pp. 1532–1545. DOI: 10.1109/TPAMI.2014.2300479.

Dosovitskiy, A., P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox (2015). “FlowNet: Learning Optical Flow with Convolutional Networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2758–2766. DOI: 10.1109/ICCV.2015.316.

Dosovitskiy, A., P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox (2016). “Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.9, pp. 1734–1747. DOI: 10.1109/TPAMI.2015.2496141.

Du, M., N. Liu, Q. Song, and X. Hu (2018). “Towards Explanation of DNN-Based Prediction with Guided Feature Inversion”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1358–1367. DOI: 10.1145/3219819.3220099.

Du, M., N. Liu, and X. Hu (2019). “Techniques for Interpretable Machine Learning”. In: *Communications of the ACM* 63.1, pp. 68–77. DOI: 10.1145/3359786.

Eigen, D., C. Puhrsch, and R. Fergus (2014). “Depth Map Prediction From a Single Image using a Multi-Scale Deep Network”. In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 27. Curran Associates, Inc.

Eitel, A., J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard (2015). “Multimodal Deep Learning for Robust RGB-D Object Recognition”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 681–687.

Elnagar, A. (2001). “Prediction of Moving Objects in Dynamic Environments Using Kalman Filters”. In: *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 414–419. DOI: 10.1109/CIRA.2001.1013236.

Everingham, M., S. M. Eslami, L. Gool, C. K. Williams, J. Winn, and A. Zisserman (2015). “The Pascal Visual Object Classes Challenge: A Retrospective”. In: *International Journal of Computer Vision (IJCV)* 111.1, pp. 98–136. DOI: 10.1007/s11263-014-0733-5.

Fagnant, D. J. and K. Kockelman (2015). “Preparing a Nation for Autonomous Vehicles: Opportunities, Barriers and Policy Recommendations”. In: *Transportation Research Part A: Policy and Practice* 77, pp. 167–181. DOI: 10.1016/j.tra.2015.04.003.

Fan, F.-L., J. Xiong, M. Li, and G. Wang (2021). “On Interpretability of Artificial Neural Networks: A Survey”. In: *IEEE Transactions on Radiation and Plasma Medical Sciences* 5.6, pp. 741–760. DOI: 10.1109/TRPMS.2021.3066428.

Fayyaz, M., M. H. Saffar, M. Sabokrou, M. Fathy, F. Huang, and R. Klette (2016). “STFCN: Spatio-Temporal Fully Convolutional Neural Network for Semantic Segmentation of Street Scenes”. In: *Computer Vision – ACCV 2016 Workshops*. Springer International Publishing, pp. 493–509. DOI: 10.1007/978-3-319-54407-6_33.

Felzenszwalb, P., D. Mcallester, and D. Ramanan (2008). “A Discriminatively Trained, Multiscale, Deformable Part Model”. In: *Proceedings of the IEEE/CVF Conference* *on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2008.4587597.

Felzenszwalb, P. F., R. B. Girshick, D. McAllester, and D. Ramanan (2010). "Object Detection with Discriminatively Trained Part-Based Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9, pp. 1627–1645. DOI: 10.1109/TPAMI.2009.167.

Fong, R. C. and A. Vedaldi (2017). "Interpretable Explanations of Black Boxes by Meaningful Perturbation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3449–3457. DOI: 10.1109/ICCV.2017.371.

Fong, R. C., M. Patrick, and A. Vedaldi (2019). "Understanding Deep Networks via Extremal Perturbations and Smooth Masks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2950–2958. DOI: 10.1109/ICCV.2019.00304.

Frosst, N. and G. Hinton (2017). "Distilling a Neural Network Into a Soft Decision Tree". In: *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017*.

Fu, C.-Y., W. Liu, A. Ranga, A. Tyagi, and A. C. Berg (2017). *DSSD: Deconvolutional Single Shot Detector*. DOI: 10.48550/arXiv.1701.06659. arXiv: 1701.06659 [cs.CV].

Gadde, R., V. Jampani, and P. V. Gehler (2017). "Semantic Video CNNs Through Representation Warping". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4463–4472. DOI: 10.1109/ICCV.2017.477.

Gade, R. and T. B. Moeslund (2014). "Thermal Cameras and Applications: A Survey". In: *Machine Vision Applications* 25.1, pp. 245–262. DOI: 10.1007/s00138-013-0570-5.

Gal, Y. and Z. Ghahramani (2016a). "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 29. Curran Associates, Inc.

Gal, Y. and Z. Ghahramani (2016b). "Bayesian Convolutional Neural Networks With Bernoulli Approximate Variational Inference". In: *International Conference on Learning Representations (ICLR), Workshop Track Proceedings*.

Ganin, Y. and V. Lempitsky (2015). “Unsupervised Domain Adaptation by Backpropagation”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML).*

Garcia-Garcia, A., S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez (2018). “A Survey on Deep Learning Techniques for Image and Video Semantic Segmentation”. In: *Applied Soft Computing* 70, pp. 41–65. DOI: 10.1016/j.asoc.2018.05.018.

Geiger, A., P. Lenz, and R. Urtasun (2012). “Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

Ghazvinian Zanjani, F. and M. van Gerven (2016). “Improving Semantic Video Segmentation by Dynamic Scene Integration”. In: *Proceedings of the Netherlands Conference on Computer Vision (NCCV).*

Ghorbani, A., J. Wexler, J. Y. Zou, and B. Kim (2019). “Towards Automatic Concept-Based Explanations”. In: *Advances in Neural Information Processing Systems (NeurIPS).* Vol. 32. Curran Associates, Inc.

Girshick, R., J. Donahue, T. Darrell, and J. Malik (2014). “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

Girshick, R. (2015). “Fast R-CNN”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

Goldhammer, M., K. Doll, U. Brunsmann, A. Gensler, and B. Sick (2014). “Pedestrian’s Trajectory Forecast in Public Traffic With Artificial Neural Networks”. In: *Proceedings of the 22nd International Conference on Pattern Recognition*, pp. 4110–4115. DOI: 10.1109/ICPR.2014.704.

Goodfellow, I., J. Shlens, and C. Szegedy (2015). “Explaining and Harnessing Adversarial Examples”. In: *International Conference on Learning Representations (ICLR).*

Goyal, Y., Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee (2019). “Counterfactual Visual Explanations”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML).*

Graves, A. (2014). *Generating Sequences With Recurrent Neural Networks*. DOI: 10.48550/arXiv.1308.0850. arXiv: 1308.0850 [cs.NE].

Greff, K., R. K. Srivastava, and J. Schmidhuber (2017). "Highway and Residual Networks Learn Unrolled Iterative Estimation". In: *International Conference on Learning Representations (ICLR)*.

Grigorescu, S., B. Trasnea, T. Cocias, and G. Macesanu (2020). "A Survey of Deep Learning Techniques for Autonomous Driving". In: *Journal of Field Robotics* 37.3, pp. 362–386. DOI: 10.1002/rob.21918.

Gu, J., X. Yang, S. De Mello, and J. Kautz (2017). "Dynamic Facial Analysis: From Bayesian Filtering to Recurrent Neural Network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1531–1540. DOI: 10.1109/CVPR.2017.167.

Guan, D., Y. Cao, J. Yang, Y. Cao, and M. Y. Yang (2019). "Fusion of Multispectral Data Through Illumination-Aware Deep Neural Networks for Pedestrian Detection". In: *Information Fusion* 50, pp. 148–157. DOI: 10.1016/j.inffus.2018.11.017.

Guo, C., G. Pleiss, Y. Sun, and K. Q. Weinberger (2017). "On Calibration of Modern Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Han, S., J. Pool, J. Tran, and W. Dally (2015). "Learning Both Weights and Connections for Efficient Neural Network". In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 28. Curran Associates, Inc.

Hao, M. and T. Yamamoto (2018). "Shared Autonomous Vehicles: A Review Considering Car Sharing and Autonomous Vehicles". In: *Asian Transport Studies* 5.1, pp. 47–63.

Hazirbas, C., L. Ma, C. Domokos, and D. Cremers (2016). "FuseNet: Incorporating Depth Into Semantic Segmentation Via Fusion-Based CNN Architecture". In: *Asian Conference on Computer Vision (ACCV)*, pp. 213–228. DOI: 10.1007/978-3-319-54181-5_14.

He, K., X. Zhang, S. Ren, and J. Sun (2016a). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

He, K., X. Zhang, S. Ren, and J. Sun (2016b). “Identity Mappings in Deep Residual Networks”. In: *Computer Vision – ECCV 2016*. Springer International Publishing, pp. 630–645. DOI: 10.1007/978-3-319-46493-0_38.

Herman, M., J. Wagner, V. Prabhakaran, N. Möser, H. Ziesche, W. Ahmed, L. Bürkle, E. Kloppenburg, and C. Gläser (2021). “Pedestrian Behavior Prediction for Automated Driving: Requirements, Metrics, and Relevant Features”. In: *IEEE Transactions on Intelligent Transportation Systems*. DOI: 10.1109/TITS.2021.3135136.

Hinton, G., O. Vinyals, and J. Dean (2014). “Distilling the Knowledge in a Neural Network”. In: *NIPS Deep Learning and Representation Learning Workshop*.

Hobert, L., A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs (2015). “Enhancements of V2X Communication in Support of Cooperative Autonomous Driving”. In: *IEEE Communications Magazine* 53.12, pp. 64–70. DOI: 10.1109/MCOM.2015.7355568.

Hooker, S., D. Erhan, P.-J. Kindermans, and B. Kim (2019). “A Benchmark for Interpretability Methods in Deep Neural Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 32. Curran Associates, Inc.

Hosang, J., M. Omran, R. Benenson, and B. Schiele (2015). “Taking a Deeper Look at Pedestrians”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4073–4082. DOI: 10.1109/CVPR.2015.7299034.

Hu, A., F. Cotter, N. Mohan, C. Gurau, and A. Kendall (2020). “Probabilistic Future Prediction for Video Scene Understanding”. In: *Computer Vision – ECCV 2020*. Springer International Publishing, pp. 767–785. DOI: 10.1007/978-3-030-58517-4_45.

Huang, G., Z. Liu, L. Van Der Maaten, and K. Q. Weinberger (2017). “Densely Connected Convolutional Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.

Hwang, S., J. Park, N. Kim, Y. Choi, and I. S. Kweon (2015). “Multispectral Pedestrian Detection: Benchmark Dataset and Baselines”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1037–1045. DOI: 10.1109/CVPR.2015.7298706.

Ilg, E., N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox (2017). “FlowNet 2.0: Evolution of Optical Flow Estimation With Deep Networks”.

In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1647–1655. DOI: 10.1109/CVPR.2017.179.

Ioffe, S. and C. Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Jain, S., X. Wang, and J. E. Gonzalez (2019). "Accel: A Corrective Fusion Network for Efficient Semantic Segmentation on Video". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8858–8867. DOI: 10.1109/CVPR.2019.00907.

Jégou, S., M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio (2017). "The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1175–1183. DOI: 10.1109/CVPRW.2017.156.

Ji, S., W. Xu, M. Yang, and K. Yu (2013). "3D Convolutional Neural Networks for Human Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1, pp. 221–231. DOI: 10.1109/TPAMI.2012.59.

Jia, Y., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell (2014). "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *Proceedings of the 22nd ACM International Conference on Multimedia*, pp. 675–678.

Jin, X., H. Xiao, X. Shen, J. Yang, Z. Lin, Y. Chen, Z. Jie, J. Feng, and S. Yan (2017a). "Predicting Scene Parsing and Motion Dynamics in the Future". In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 30. Curran Associates, Inc.

Jin, X., X. Li, H. Xiao, X. Shen, Z. Lin, J. Yang, Y. Chen, J. Dong, L. Liu, Z. Jie, J. Feng, and S. Yan (2017b). "Video Scene Parsing with Predictive Feature Learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5581–5589. DOI: 10.1109/ICCV.2017.595.

Karasev, V., A. Ayvaci, B. Heisele, and S. Soatto (2016). "Intent-Aware Long-Term Prediction of Pedestrian Motion". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2543–2549. DOI: 10.1109/ICRA.2016.7487409.

Kendall, A., V. Badrinarayanan, and R. Cipolla (2017). "Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding”. In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 57.1–57.12. DOI: 10.5244/C.31.57.

Kendall, A. and Y. Gal (2017). “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 30. Curran Associates, Inc.

Kendall, A., Y. Gal, and R. Cipolla (2018). “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)*, pp. 7482–7491. DOI: 10.1109/CVPR.2018.00781.

Khakzar, A., S. Baselizadeh, S. Khanduja, C. Rupprecht, S. T. Kim, and N. Navab (2020). *Improving Feature Attribution Through Input-Specific Network Pruning*. DOI: 10.48550/arXiv.1911.11081. arXiv: 1911.11081 [cs.CV].

Khakzar, A., S. Baselizadeh, S. Khanduja, C. Rupprecht, S. T. Kim, and N. Navab (2021). “Neural Response Interpretation Through the Lens of Critical Pathways”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13528–13538. DOI: 10.1109/CVPR46437.2021.01332.

Kim, B., J. Seo, S. Jeon, J. Koo, J. Choe, and T. Jeon (2019). “Why are Saliency Maps Noisy? Cause of and Solution to Noisy Saliency Maps”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 4149–4157. DOI: 10.1109/ICCVW.2019.00510.

Kingma, D. P. and J. Ba (2015). “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations (ICLR)*.

Kiureghian, A. D. and O. Ditlevsen (2009). “Aleatory or Epistemic? Does It Matter?” In: *Structural Safety* 31.2, pp. 105–112. DOI: 10.1016/j.strusafe.2008.06.020.

König, D., M. Adam, C. Jarvers, G. Layher, H. Neumann, and M. Teutsch (2017). “Fully Convolutional Region Proposal Networks for Multispectral Person Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 243–250. DOI: 10.1109/CVPRW.2017.36.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 25. Curran Associates, Inc.

Krotosky, S. J. and M. M. Trivedi (2008). “Person Surveillance Using Visual and Infrared Imagery”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 18.8, pp. 1096–1105. DOI: 10.1109/TCSVT.2008.928217.

Krueger, D., T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. C. Courville, and C. J. Pal (2017). “Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations”. In: *International Conference on Learning Representations (ICLR)*.

Kruse, T., A. K. Pandey, R. Alami, and A. Kirsch (2013). “Human-Aware Robot Navigation: A Survey”. In: *Robotics and Autonomous Systems* 61.12, pp. 1726–1743. DOI: 10.1016/j.robot.2013.05.007.

Kundu, A., V. Vineet, and V. Koltun (2016). “Feature Space Optimization for Semantic Video Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3168–3175. DOI: 10.1109/CVPR.2016.345.

Kurakin, A., I. Goodfellow, and S. Bengio (2017). “Adversarial Examples in the Physical World”. In: *International Conference on Learning Representations (ICLR), Workshop Track Proceedings*.

St-Laurent, L., X. Maldague, and D. Prévost (2007). “Combination of Colour and Thermal Sensors for Enhanced Object Detection”. In: *10th International Conference on Information Fusion*, pp. 1–8. DOI: 10.1109/ICIF.2007.4408003.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). “Gradient-Based Learning Applied to Document Recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Lee, J. H., J.-S. Choi, E. S. Jeon, Y. G. Kim, T. T. Le, K. Y. Shin, H. C. Lee, and K. R. Park (2015). “Robust Pedestrian Detection by Combining Visible and Thermal Infrared Cameras”. In: *Sensors* 15.5, pp. 10580–10615. DOI: 10.3390/s150510580.

Lee, Y., T. D. Bui, and J. Shin (2018). “Pedestrian Detection Based on Deep Fusion Network Using Feature Correlation”. In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 694–699. DOI: 10.23919/APSIPA.2018.8659688.

Lefèvre, S., D. Vasquez, and C. Laugier (2014). “A Survey on Motion Prediction and Risk Assessment for Intelligent Vehicles”. In: *ROBOMECH Journal* 1.1. DOI: 10.1186/s40648-014-0001-z.

Lei, P. and S. Todorovic (2016). “Recurrent Temporal Deep Field for Semantic Video Labeling”. In: *Computer Vision – ECCV 2016*. Springer International Publishing, pp. 302–317. DOI: 10.1007/978-3-319-46454-1_19.

Leykin, A. and R. Hammoud (2006). “Robust Multi-Pedestrian Tracking in Thermal-Visible Surveillance Videos”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. DOI: 10.1109/CVPRW.2006.175.

Leykin, A., Y. Ran, and R. Hammoud (2007). “Thermal-Visible Video Fusion for Moving Target Tracking and Pedestrian Classification”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2007.383444.

Li, C., D. Song, R. Tong, and M. Tang (2018). “Multispectral Pedestrian Detection via Simultaneous Detection and Segmentation”. In: *Proceedings of the British Machine Vision Conference (BMVC)*.

Li, C., D. Song, R. Tong, and M. Tang (2019). “Illumination-Aware Faster R-CNN for Robust Multispectral Pedestrian Detection”. In: *Pattern Recognition* 85, pp. 161–171. DOI: 10.1016/j.patcog.2018.08.005.

Liggins, M. E., D. L. Hall, and J. Llinas (2008). *Handbook of Multisensor Data Fusion: Theory and Practice; Second Edition*. Electrical Engineering and Applied Signal Processing Series. Taylor & Francis. DOI: 10.1201/9781420053098.

Linardatos, P., V. Papastefanopoulos, and S. Kotsiantis (2021). “Explainable AI: A Review of Machine Learning Interpretability Methods”. In: *Entropy* 23.1. DOI: 10.3390/e23010018.

Liu, Y.-C., Y.-A. Hsieh, M.-H. Chen, C.-H. H. Yang, J. Tegner, and Y.-C. J. Tsai (2020). “Interpretable Self-Attention Temporal Reasoning for Driving Behavior Understanding”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2338–2342. DOI: 10.1109/ICASSP40776.2020.9053783.

Liu, J., S. Zhang, S. Wang, and D. Metaxas (2016a). “Multispectral Deep Neural Networks for Pedestrian Detection”. In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 73.1–73.13. DOI: 10.5244/C.30.73.

Liu, W., D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg (2016b). “SSD: Single Shot Multibox Detector”. In: *Computer Vision – ECCV 2016*. Springer International Publishing, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2.

Long, J., E. Shelhamer, and T. Darrell (2015). “Fully Convolutional Networks for Semantic Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

Lotter, W., G. Kreiman, and D. Cox (2016). “Unsupervised Learning of Visual Structure Using Predictive Generative Networks”. In: *International Conference on Learning Representations (ICLR)*.

Luber, M., J. A. Stork, G. D. Tipaldi, and K. O. Arras (2010). “People Tracking With Human Motion Predictions From Social Forces”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 464–469. DOI: 10.1109/ROBOT.2010.5509779.

Luc, P., N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun (2017). “Predicting Deeper into the Future of Semantic Segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 648–657. DOI: 10.1109/ICCV.2017.77.

Luc, P., C. Couprie, Y. Lecun, and J. Verbeek (2018). “Predicting Future Instance Segmentation by Forecasting Convolutional Features”. In: *Computer Vision – ECCV 2018*, pp. 593–608. DOI: 10.1007/978-3-030-01240-3_36.

Luo, P., Y. Tian, X. Wang, and X. Tang (2014). “Switchable Deep Network for Pedestrian Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 899–906. DOI: 10.1109/CVPR.2014.120.

Maas, A. L., A. Y. Hannun, and A. Y. Ng (2013). “Rectifier Nonlinearities Improve Neural Network Acoustic Models”. In: *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Mahjourian, R., M. Wicke, and A. Angelova (2017). “Geometry-Based Next Frame Prediction From Monocular Video”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 1700–1707. DOI: 10.1109/IVS.2017.7995953.

Mahjourian, R., M. Wicke, and A. Angelova (2018). “Unsupervised Learning of Depth and Ego-Motion From Monocular Video Using 3D Geometric Constraints”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5667–5675. DOI: 10.1109/CVPR.2018.00594.

Mathieu, M., C. Couprie, and Y. LeCun (2016). “Deep Multi-Scale Video Prediction Beyond Mean Square Error”. In: *International Conference on Learning Representations (ICLR)*.

McAllister, R., Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. V. Weller (2017). "Concrete Problems for Autonomous Vehicle Safety: Advantages of Bayesian Deep Learning". In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4745–4753. DOI: 10.24963/ijcai.2017/661.

McCormac, J., A. Handa, S. Leutenegger, and A. J. Davison (2017). "SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-Training on Indoor Segmentation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2697–2706. DOI: 10.1109/ICCV.2017.292.

Mees, O., A. Eitel, and W. Burgard (2016). "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)*, pp. 151–156. DOI: 10.1109/IROS.2016.7759048.

Michalski, V., R. Memisevic, and K. Konda (2014). "Modeling Deep Temporal Dependencies with Recurrent Grammar Cells". In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 27. Curran Associates, Inc.

Mueller, A. S., J. B. Cicchino, and D. S. Zuby (2020). "What Humanlike Errors Do Autonomous Vehicles Need to Avoid to Maximize Safety?" In: *Journal of Safety Research* 75, pp. 310–318. DOI: 10.1016/j.jsr.2020.10.005.

Nabavi, S. S., M. Rochan, and Y. Wang (2018). "Future Semantic Segmentation with Convolutional LSTM". In: *Proceedings of the British Machine Vision Conference (BMVC)*.

Nair, V. and G. E. Hinton (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*.

Nguyen, A., A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune (2016). "Synthesizing the Preferred Inputs for Neurons in Neural Networks via Deep Generator Networks". In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 29. Curran Associates, Inc.

NHTSA (2008). *National Motor Vehicle Crash Causation Survey: Report to Congress*. Report No. DOT HS 811 059. U.S. Department of Transportation, National Highway Traffic Safety Administration (NHTSA).

Nie, W., Y. Zhang, and A. Patel (2018). "A Theoretical Explanation for Perplexing Behaviors of Backpropagation-Based Visualizations". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Nilsson, D. and C. Sminchisescu (2018). “Semantic Video Segmentation by Gated Recurrent Flow Propagation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6819–6828. DOI: 10.1109/CVPR.2018.00713.

Omeiza, D., S. Speakman, C. Cintas, and K. Weldermariam (2019). *Smooth Grad-CAM++: An Enhanced Inference Level Visualization Technique for Deep Convolutional Neural Network Models*. DOI: 10.48550/arXiv.1908.01224. arXiv: 1908.01224 [cs.CV].

Ondruska, P. and I. Posner (2016). “Deep Tracking: Seeing Beyond Seeing Using Recurrent Neural Networks”. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Oprea, S., P. Martinez-Gonzalez, A. Garcia-Garcia, J. A. Castro-Vargas, S. Orts-Escolano, J. Garcia-Rodriguez, and A. Argyros (2020). “A Review on Deep Learning Techniques for Video Prediction”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. DOI: 10.1109/TPAMI.2020.3045007.

Ouali, Y., C. Hudelot, and M. Tami (2020). *An Overview of Deep Semi-Supervised Learning*. DOI: 10.48550/arXiv.2006.05278. arXiv: 2006.05278 [cs.LG].

Ouyang, W. and X. Wang (2013). “Joint Deep Learning for Pedestrian Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2056–2063. DOI: 10.1109/ICCV.2013.257.

Papageorgiou, C. P., M. Oren, and T. Poggio (1998). “A General Framework for Object Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 555–562. DOI: 10.1109/ICCV.1998.710772.

Patraucean, V., A. Handa, and R. Cipolla (2015). *Spatio-Temporal Video Autoencoder With Differentiable Memory*. DOI: 10.48550/arXiv.1511.06309. arXiv: 1511.06309 [cs.LG].

Pavel, M. S., H. Schulz, and S. Behnke (2015). “Recurrent Convolutional Neural Networks for Object-Class Segmentation of RGB-D Video”. In: *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280820.

Pavel, M. S., H. Schulz, and S. Behnke (2017). “Object Class Segmentation of RGB-D Video Using Recurrent Convolutional Neural Networks”. In: *Neural Networks* 88, pp. 105–113. DOI: 10.1016/j.neunet.2017.01.003.

Pellegrini, S., A. Ess, K. Schindler, and L. Van Gool (2009). “You’ll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 261–268. DOI: 10.1109/ICCV.2009.5459260.

Petsiuk, V., A. Das, and K. Saenko (2018). “RISE: Randomized Input Sampling for Explanation of Black-Box Models”. In: *Proceedings of the British Machine Vision Conference (BMVC)*.

Pfeuffer, A., K. Schulz, and K. Dietmayer (2019). “Semantic Segmentation of Video Sequences with Convolutional LSTMs”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 1441–1447. DOI: 10.1109/IVS.2019.8813852.

Pfeuffer, A. and K. Dietmayer (2019). “Separable Convolutional LSTMs for Faster Video Segmentation”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1072–1078. DOI: 10.1109/ITSC.2019.8917487.

Pfeuffer, A. and K. Dietmayer (2020). “Robust Semantic Segmentation in Adverse Weather Conditions by Means of Fast Video-Sequence Segmentation”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294554.

Qi, Z., S. Khorram, and F. Li (2019). “Visualizing Deep Networks by Optimizing with Integrated Gradients”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Radwan, N., A. Valada, and W. Burgard (2018). “VLocNet++: Deep Multitask Learning for Semantic Visual Localization and Odometry”. In: *IEEE Robotics and Automation Letters* 3.4, pp. 4407–4414. DOI: 10.1109/LRA.2018.2869640.

Ranzato, M., A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra (2014). *Video (Language) Modeling: A Baseline for Generative Models of Natural Videos*. DOI: 10.48550/arXiv.1412.6604. arXiv: 1412.6604 `[cs.LG]`.

Rasmus, A., M. Berglund, M. Honkala, H. Valpola, and T. Raiko (2015). “Semi-Supervised Learning with Ladder Networks”. In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 28. Curran Associates, Inc.

Raza, S. H., M. Grundmann, and I. Essa (2013). “Geometric Context From Videos”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3081–3088. DOI: 10.1109/CVPR.2013.396.

Redmon, J., S. Divvala, R. Girshick, and A. Farhadi (2016). “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

Rehder, E. and H. Kloeden (2015). “Goal-Directed Pedestrian Prediction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 139–147. DOI: 10.1109/ICCVW.2015.28.

Ren, S., K. He, R. Girshick, and J. Sun (2015). “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 28. Curran Associates, Inc.

Ribeiro, M. T., S. Singh, and C. Guestrin (2016). “Why Should I Trust You?: Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. DOI: 10.1145/2939672.2939778.

Richter, S. R., V. Vineet, S. Roth, and V. Koltun (2016). “Playing for Data: Ground Truth From Computer Games”. In: *Computer Vision – ECCV 2016*. Springer International Publishing, pp. 102–118. DOI: 10.1007/978-3-319-46475-6_7.

Rudenko, A., L. Palmieri, and K. O. Arras (2017). “Predictive Planning for a Mobile Robot in Human Environments”. In: *Proceedings of the Workshop on AI Planning and Robotics: Challenges and Methods (at ICRA 2017), Singapore*.

Rudenko, A., L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras (2020). “Human Motion Trajectory Prediction: A Survey”. In: *The International Journal of Robotics Research* 39.8, pp. 895–935. DOI: 10.1177/0278364920917446.

Ruder, S. (2017). *An Overview of Multi-Task Learning in Deep Neural Networks*. DOI: 10.48550/arXiv.1706.05098. arXiv: 1706.05098 `[cs.LG]`.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2015). “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3, pp. 211–252.

Saemann, T., K. Amende, S. Milz, and H.-M. Gross (2019). “Leverage Temporal Consistency for Robust Semantic Video Segmentation”. In: *ICML Workshop on Uncertainty and Robustness in Deep Learning*.

Samek, W., G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller (2021). "Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications". In: *Proceedings of the IEEE* 109.3, pp. 247–278. DOI: 10.1109/JPROC.2021.3060483.

Sánchez Pérez, J., E. Meinhardt-Llopis, and G. Facciolo (2013). "TV-L1 Optical Flow Estimation". In: *Image Processing On Line* 3, pp. 137–150.

Saric, J., M. Orsic, T. Antunovic, S. Vrazic, and S. Segvic (2020). "Warp to the Future: Joint Forecasting of Features and Feature Motion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10645–10654. DOI: 10.1109/CVPR42600.2020.01066.

Sauer, A., N. Savinov, and A. Geiger (2018). "Conditional Affordance Learning for Driving in Urban Environments". In: *Conference on Robot Learning (CoRL)*, pp. 237–252.

Schneider, N. and D. M. Gavrila (2013). "Pedestrian Path Prediction with Recursive Bayesian Filters: A Comparative Study". In: *Pattern Recognition*, pp. 174–183. DOI: 10.1007/978-3-642-40602-7_18.

Schulz, K., L. Sixt, F. Tombari, and T. Landgraf (2020). "Restricting the Flow: Information Bottlenecks for Attribution". In: *International Conference on Learning Representations (ICLR)*.

Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra (2017). "Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

Seo, D., K. Oh, and I.-S. Oh (2019). "Regional Multi-Scale Approach for Visually Pleasing Explanations of Deep Neural Networks". In: *IEEE Access* 8, pp. 8572–8582. DOI: 10.1109/ACCESS.2019.2963055.

Sermanet, P., K. Kavukcuoglu, S. Chintala, and Y. Lecun (2013). "Pedestrian Detection with Unsupervised Multi-stage Feature Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3626–3633. DOI: 10.1109/CVPR.2013.465.

Shelhamer, E., K. Rakelly, J. Hoffman, and T. Darrell (2016). "Clockwork Convnets for Video Semantic Segmentation". In: *Computer Vision – ECCV 2016 Workshops*. Springer International Publishing, pp. 852–868. DOI: 10.1007/978-3-319-49409-8_69.

Shrikumar, A., P. Greenside, and A. Kundaje (2017). “Learning Important Features Through Propagating Activation Differences”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Simonyan, K., A. Vedaldi, and A. Zisserman (2014). “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *International Conference on Learning Representations (ICLR), Workshop Track Proceedings*.

Simonyan, K. and A. Zisserman (2014). “Two-Stream Convolutional Networks for Action Recognition in Videos”. In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 27. Curran Associates, Inc.

Simonyan, K. and A. Zisserman (2015). “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations (ICLR)*.

Singh, S. (2015). *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*. Traffic Safety Facts Crash Stats, Report No. DOT HS 812 115. Washington, DC: National Highway Traffic Safety Administration.

Smilkov, D., N. Thorat, B. Kim, F. Viégas, and M. Wattenberg (2017). “SmoothGrad: Removing Noise by Adding Noise”. In: *ICML Workshop on Visualization for Deep Learning*.

Springenberg, J. T., A. Dosovitskiy, T. Brox, and M. Riedmiller (2015). “Striving for Simplicity: The All Convolutional Net”. In: *International Conference on Learning Representations (ICLR), Workshop Track Proceedings*.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). “Dropout: A Simple Way to Prevent Neural Networks From Overfitting”. In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958.

Srivastava, N., E. Mansimov, and R. Salakhudinov (2015). “Unsupervised Learning of Video Representations using LSTMs”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Suard, F., A. Rakotomamonjy, A. Bensrhair, and A. Broggi (2006). “Pedestrian Detection Using Infrared Images and Histograms of Oriented Gradients”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 206–212. DOI: 10.1109/IVS.2006.1689629.

Sundararajan, M., A. Taly, and Q. Yan (2017). “Axiomatic Attribution for Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2014). “Intriguing Properties of Neural Networks”. In: *International Conference on Learning Representations (ICLR)*.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015). “Going Deeper With Convolutions”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2015.7298594.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer Science & Business Media. DOI: 10.1007/978-3-030-34372-9.

Tampuu, A., T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad (2020). “A Survey of End-to-End Driving: Architectures and Training Methods”. In: *IEEE Transactions on Neural Networks and Learning Systems*. DOI: 10.1109/TNNLS.2020.3043505.

Tang, Z., D. Wang, and Z. Zhang (2016). “Recurrent Neural Network Training With Dark Knowledge Transfer”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5900–5904. DOI: 10.1109/ICASSP.2016.7472809.

Terwilliger, A., G. Brazil, and X. Liu (2019). “Recurrent Flow-Guided Semantic Forecasting”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1703–1712. DOI: 10.1109/WACV.2019.00186.

Teutsch, M., T. Mueller, M. Huber, and J. Beyerer (2014). “Low Resolution Person Detection with a Moving Thermal Infrared Camera by Hot Spot Classification”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 209–216. DOI: 10.1109/CVPRW.2014.40.

Tian, Y., P. Luo, X. Wang, and X. Tang (2015). “Pedestrian Detection Aided by Deep Learning Semantic Tasks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5079–5087. DOI: 10.1109/CVPR.2015.7299143.

Torfi, A., R. A. Shirvani, Y. Keneshloo, N. Tavaf, and E. A. Fox (2020). *Natural Language Processing Advancements By Deep Learning: A Survey*. DOI: 10.48550/arXiv.2003.01200. arXiv: 2003.01200 [cs.CL].

Torresan, H., B. Turgeon, C. Ibarra-Castanedo, P. Hebert, and X. P. Maldague (2004). “Advanced Surveillance Systems: Combining Video and Thermal Imagery for Pedestrian Detection”. In: *Thermosense XXVI*. Vol. 5405, pp. 506–515.

Tran, D., L. Bourdev, R. Fergus, L. Torresani, and M. Paluri (2015). “Learning Spatiotemporal Features with 3D Convolutional Networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4489–4497. DOI: 10.1109/ICCV.2015.510.

Tran, D., L. Bourdev, R. Fergus, L. Torresani, and M. Paluri (2016). “Deep End2End Voxel2Voxel Prediction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 402–409. DOI: 10.1109/CVPRW.2016.57.

Ummenhofer, B., H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox (2017). “DeMoN: Depth and Motion Network for Learning Monocular Stereo”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5622–5631. DOI: 10.1109/CVPR.2017.596.

Valipour, S., M. Siam, M. Jagersand, and N. Ray (2017). “Recurrent Fully Convolutional Networks for Video Segmentation”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 29–36. DOI: 10.1109/WACV.2017.11.

Veit, A., M. J. Wilber, and S. Belongie (2016). “Residual Networks Behave Like Ensembles of Relatively Shallow Networks”. In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 29. Curran Associates, Inc.

Verma, A., R. Hebbalaguppe, L. Vig, S. Kumar, and E. Hassan (2015). “Pedestrian Detection via Mixture of CNN Experts and Thresholded Aggregated Channel Features”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 555–563. DOI: 10.1109/ICCVW.2015.78.

Vijayanarasimhan, S., S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki (2017). *SfM-Net: Learning of Structure and Motion From Video*. DOI: 10.48550/arXiv.1704.07804. arXiv: 1704.07804 [cs.CV].

Viola, P. and M. Jones (2001). “Rapid Object Detection Using a Boosted Cascade of Simple Features”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2001.990517.

Viola, P. and M. Jones (2004). “Robust Real-Time Face Detection”. In: *International Journal of Computer Vision (IJCV)* 57.2, pp. 137–154. DOI: 10.1023/B:VISI.0000013087.49260.fb.

Vondrick, C., H. Pirsiavash, and A. Torralba (2016). “Anticipating Visual Representations From Unlabeled Video”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 98–106. DOI: 10.1109/CVPR.2016.18.

Vu, T. H., W. Choi, S. Schulter, and M. Chandraker (2019). “Memory Warps for Long-Term Online Video Representations and Anticipation”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1156–1165. DOI: 10.1109/WACV.2019.00128.

Wagner, J., V. Fischer, M. Herman, and S. Behnke (2016). “Multispectral Pedestrian Detection using Deep Fusion Convolutional Neural Networks”. In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 509–514.

Wagner, J., V. Fischer, M. Herman, and S. Behnke (2017). “Learning Semantic Prediction using Pretrained Deep Feedforward Networks”. In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 565–570.

Wagner, J., V. Fischer, M. Herman, and S. Behnke (2018a). “Functionally Modular and Interpretable Temporal Filtering for Robust Segmentation”. In: *Proceedings of the British Machine Vision Conference (BMVC)*.

Wagner, J., V. Fischer, M. Herman, and S. Behnke (2018b). “Hierarchical Recurrent Filtering for Fully Convolutional DenseNets”. In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 49–54.

Wagner, J., J. M. Köhler, T. Gindele, L. Hetzel, J. T. Wiedemer, and S. Behnke (2019). “Interpretable and Fine-Grained Visual Explanations for Convolutional Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9089–9099. DOI: 10.1109/CVPR.2019.00931.

Wang, H., Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu (2020a). “Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 111–119. DOI: 10.1109/CVPRW50498.2020.00020.

Wang, T.-H., S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun (2020b). "V2VNet: Vehicle-to-Vehicle Communication for Joint Perception and Prediction". In: *Computer Vision – ECCV 2020*. Springer International Publishing, pp. 605–621. DOI: 10.1007/978-3-030-58536-5_36.

Weller, A. (2019). "Transparency: Motivations and Challenges". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, pp. 23–40. DOI: 10.1007/978-3-030-28954-6_2.

Werbos, P. J. (1990). "Backpropagation Through Time: What It Does and How to Do It". In: *Proceedings of the IEEE* 78.10, pp. 1550–1560. DOI: 10.1109/5.58337.

Xingjian, S., Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo (2015). "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 28. Curran Associates, Inc.

Xu, Y.-S., T.-J. Fu, H.-K. Yang, and C.-Y. Lee (2018). "Dynamic Video Segmentation Network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6556–6565. DOI: 10.1109/CVPR.2018.00686.

Yang, B., J. Yan, Z. Lei, and S. Z. Li (2015). "Convolutional Channel Features". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 82–90. DOI: 10.1109/ICCV.2015.18.

Yang, G. and D. Ramanan (2021). "Learning to Segment Rigid Motions From Two Frames". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1266–1275. DOI: 10.1109/CVPR46437.2021.00132.

Yi, S., H. Li, and X. Wang (2015). "Understanding Pedestrian Behaviors From Stationary Crowd Groups". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3488–3496. DOI: 10.1109/CVPR.2015.7298971.

Yin, Z. and J. Shi (2018). "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)*, pp. 1983–1992. DOI: 10.1109/CVPR.2018.00212.

Yosinski, J., J. Clune, Y. Bengio, and H. Lipson (2014). "How Transferable are Features in Deep Neural Networks?" In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 27. Curran Associates, Inc.

Yu, F. and V. Koltun (2016). “Multi-Scale Context Aggregation by Dilated Convolutions”. In: *International Conference on Learning Representations (ICLR)*.

Yurdakul, E. E. and Y. Yemez (2017). “Semantic Segmentation of RGBD Videos with Recurrent Fully Convolutional Neural Networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 367–374. DOI: 10.1109/ICCVW.2017.51.

Zablocki, É., H. Ben-Younes, P. Pérez, and M. Cord (2021). *Explainability of Vision-Based Autonomous Driving Systems: Review and Challenges*. DOI: 10.48550/arXiv.2101.05307. arXiv: 2101.05307 [cs.CV].

Zeiler, M. D. and R. Fergus (2014). “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision – ECCV 2014*. Springer International Publishing, pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53.

Zhang, H., K. Jiang, Y. Zhang, Q. Li, C. Xia, and X. Chen (2014). “Discriminative Feature Learning for Video Semantic Segmentation”. In: *Proceedings of the International Conference on Virtual Reality and Visualization (ICVRV)*, pp. 321–326. DOI: 10.1109/ICVRV.2014.65.

Zhang, J., Z. Lin, J. Brandt, X. Shen, and S. Sclaroff (2016). “Top-Down Neural Attention by Excitation Backprop”. In: *Computer Vision – ECCV 2016*. Springer International Publishing, pp. 543–559. DOI: 10.1007/978-3-319-46493-0_33.

Zhang, L., B. Wu, and R. Nevatia (2007). “Pedestrian Detection in Infrared Images Based on Local Shape Features”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2007.383452.

Zhang, L., X. Zhu, X. Chen, X. Yang, Z. Lei, and Z. Liu (2019). “Weakly Aligned Cross-Modal Learning for Multispectral Pedestrian Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5126–5136. DOI: 10.1109/ICCV.2019.00523.

Zhang, Q. and S.-C. Zhu (2018). “Visual Interpretability for Deep Learning: A Survey”. In: *Frontiers of Information Technology & Electronic Engineering* 19.1, pp. 27–39. DOI: 10.1631/FITEE.1700808.

Zhao, H., J. Shi, X. Qi, X. Wang, and J. Jia (2017). “Pyramid Scene Parsing Network”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6230–6239. DOI: 10.1109/CVPR.2017.660.

Zhao, H., X. Qi, X. Shen, J. Shi, and J. Jia (2018). "ICNet for Real-Time Semantic Segmentation on High-Resolution Images". In: *Computer Vision – ECCV 2018*, pp. 418–434. DOI: 10.1007/978-3-030-01219-9_25.

Zhou, B., A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba (2015). "Object Detectors Emerge in Deep Scene CNNs". In: *International Conference on Learning Representations (ICLR)*.

Zhou, B., A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba (2016). "Learning Deep Features for Discriminative Localization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2921–2929. DOI: 10.1109/CVPR.2016.319.

Zhou, K., L. Chen, and X. Cao (2020). "Improving Multispectral Pedestrian Detection by Addressing Modality Imbalance Problems". In: *Computer Vision – ECCV 2020*. Springer International Publishing, pp. 787–803. DOI: 10.1007/978-3-030-58523-5_46.

Zhou, T., M. Brown, N. Snavely, and D. G. Lowe (2017). "Unsupervised Learning of Depth and Ego-Motion From Video". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6612–6619. DOI: 10.1109/CVPR.2017.700.

Zhu, X., Y. Xiong, J. Dai, L. Yuan, and Y. Wei (2017). "Deep Feature Flow for Video Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4141–4150. DOI: 10.1109/CVPR.2017.441.

Zhuang, J., Z. Wang, and B. Wang (2021). "Video Semantic Segmentation With Distortion-Aware Feature Correction". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.8, pp. 3128–3139. DOI: 10.1109/TCSVT.2020.3037234.

Ziebart, B. D., N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa (2009). "Planning-Based Prediction for Pedestrians". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3931–3936. DOI: 10.1109/IROS.2009.5354147.

Zilke, J. R., E. L. Mencía, and F. Janssen (2016). "DeepRed–Rule Extraction From Deep Neural Networks". In: *International Conference on Discovery Science*, pp. 457–473. DOI: 10.1007/978-3-319-46307-0_29.

Zin, T. T., H. Takahashi, T. Toriu, and H. Hama (2011). "Fusion of Infrared and Visible Images for Robust Person Detection". In: *Image Fusion*. IntechOpen. Chap. 12.