TUM School of Computation, Information and Technology
Technische Universität München

Failure Prediction for Autonomous Systems

Christopher B. Kuhn, M.Sc.

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:
Prof. Dr. Reinhard Heckel

Prüfer der Dissertation:
1. Prof. Dr.-Ing. Eckehard Steinbach
2. Prof. Abhishek Dubey

Die Dissertation wurde am 31.03.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 18.11.2022 angenommen.

Abstract

Failures in autonomous systems are inevitable. While the performance of automated and assisted driving systems has improved over the last years, such systems will not be able to handle every situation on their own in the foreseeable future. Failures can be caused by ambiguous or compromised input, by an insufficiently trained or ill-designed model, or by out-of-distribution data that the system has not been trained for. In safety-critical tasks such as driving, the resulting failures can have severe consequences. The concept of failure prediction can be used to avoid dangerous mistakes. While a system is not always capable of making the right decision, it can still be possible to detect incorrect decisions. Then, the system can either try to resolve the failure itself, switch to a safer backup strategy, or return control to a human operator. In this thesis, we investigate how failures of autonomous systems can be predicted. We consider the entire pipeline of a general autonomous driving system and propose three main additions: system-level failure prediction, component-level failure prediction, and applications of failure prediction to correct or avoid failures.

System-level failures describe a state where the system cannot safely execute the desired autonomous behavior, regardless of which component is at fault. Such failures result in a disengagement of the system and require a human operator to resolve the situation. We propose a failure prediction framework that allows to predict disengagements up to seven seconds in advance. We apply the idea of introspection and use previously recorded system failures as training data to learn how to predict new failures. Training a classifier with sequences of state data achieves an average failure prediction accuracy of 78.8 %, using sequences of planned trajectories achieves 79.6 %, and using image sequences results in an accuracy of 85.1 %. Finally, we propose a late fusion approach of combining all three input types, achieving an overall accuracy of 89.1 %. Even seven seconds in advance, our fusion approach predicts over 86 % of failures, at a false positive rate of less than 15 %.

A system-level failure is typically caused by one or more component-level failures. As our second main contribution, we propose a failure prediction method for the individual perception component of semantic segmentation. We again apply the concept of introspection and use pixel-wise errors made by a given semantic segmentation model as training labels. We then train a separate model to classify each pixel of an input image as failure or success. We outperform the state of the art by more than 7 % in a precision-recall analysis. Additionally, we propose to use sequences of previous predicted errors to extrapolate into the future, allowing to accurately predict future semantic segmentation errors up to 0.5 s ahead.

Finally, we use failure prediction concepts to correct or avoid failures. Inspired by human perception, we design a two-stage approach for detecting and reclassifying regions which a state-of-the-art semantic segmentation model misclassified. Applied to a driving data set, our system correctly classifies 63 % of the road participants in the failure regions. To correct failure pixels directly, we propose the idea of reverse error modeling. We train an autoencoder to reconstruct the correct semantic map from the erroneous predictions of a given model, improving the performance by 0.7 % for uncompressed images and by over 8 % for compressed input. Lastly, instead of correcting mistakes, we propose a sensor monitoring approach that avoids mistakes by preemptively reducing the influence of problematic sensor input. We train an introspective monitoring model for each sensor and then weight the fusion of all sensor inputs with the predicted success probability. By focusing on sensors with a low failure probability, we improve the performance of a state-of-the-art multimodal object detection network by 4.6 %.

Acknowledgment

The work presented in this dissertation was carried out as a member of both the academic staff at the Chair of Media Technology (LMT) at the Technical University of Munich (TUM) and the Automated and Autonomous Driving division at the BMW Group. Many people supported me during the last three years, both professionally and personally.

First of all, I would like to sincerely thank Prof. Dr.-Ing. Eckehard Steinbach for the supervision of this project at TUM. I am deeply grateful for the guidance, support, and invaluable insights he provided at every step of this work. He always had time for fruitful discussions, where he never failed to offer new ideas and inspiration.

Furthermore, I would like to thank Prof. Abhishek Dubey for agreeing to become the second examiner. I would like to especially thank Prof. Dubey for the opportunity to work with his team at Vanderbilt University for the last four months of this project, which was a deeply enjoyable and enriching experience.

Sincere thanks also go to my supervisor and mentor at BMW, Dr. Goran Petrovic. He initiated my research project in the first place and always provided me with support and insights to ensure a successful collaboration between BMW and TUM. His enthusiastic and optimistic attitude made working on this project a pleasure, both on a professional and a personal level.

While I highly value all of my former colleagues, I feel obliged to highlight several people. Most of all, I am grateful to my colleague, office mate, and friend Markus Hofbauer for our exceptional collaboration and fruitful discussions on a daily basis. Working together has made every day of this project more enjoyable and I am looking forward to our future collaboration. I would also like to extend thanks to Dr. Christoph Bachhuber for great discussions and collaborations even after he had left LMT. Special thanks also go to Dr. Tamay Aykut for interesting and joyful talks during our time at LMT as well as during our collaboration afterwards. I look forward to the opportunity to continue working together in the future.

Next, I would like to thank Dr. Rahul Chaudhari for many engaging discussions both inside and outside of LMT. I would like to thank Basak Gülecyüz and Andreas Noll for being a pleasure to work with on the Digital Signal Processing lecture, especially when preparing our first online-only semester. Further thanks go to Ermin Sakic for many valuable conversations while in home office, Martin Piccolrovazzi and Yuankai Wu for always making the time at LMT more enjoyable, and to Furkan Kaynar for the interesting discussions beyond our projects. I am also grateful to all my former students who contributed to this project, especially Ziqin Xu, Ma Bowen, and Lukas Habermayr.

My appreciation also goes to the administrative support provided by Marta Giunta, Etienne Meyer, Dr. Martin Maier, and Simon Krapf. From my time at Vanderbilt University, special thanks go to Shreyas Ramakrishna for being both a great colleague and friend, making my stay there a highly rewarding experience. From my BMW colleagues, I would like to thank Sebastian Wirkert and Sebastian Schmidt for always making my time there more interesting and enjoyable. Special thanks go to Greta Kölln, for being the best source of motivation anyone could ask for.

Last but not least, I would like to express my deepest gratitude to my family, in particular my beloved parents as well as my brothers Alexander and Sebastian, who always encouraged me and believed in my abilities.

Christopher Kuhn, Nashville, TN, March 22, 2022

Contents

Abbreviations

ADAS	Advanced Driver-Assistance System	1
ASPP	Atrous Spatial Pooling Pyramid	12
AUC	Area Under Curve	127
AVOD	Aggregate View Object Detection	128
BEV	Bird's Eye View	128
CAM	Class Activation Map	124
CARLA	Car Learning to Act	77
CE	Cross Entropy	9
CNN	Convolutional Neural Network	123
DAE	Denoising Autoencoder	12
DAVID	Densely Annotated Video Driving	127
ERA	Error-Reversing Autoencoder	128
FC	Fully Connected	123
FPR	False Positive Rate	40
GAN	Generative Adversarial Network	13
grad-CAM	gradient-weighted Class Activation Map	46
HD	High Definition	73
IoU	Intersection over Union	128
JPEG	Joint Photographic Experts Group	126
LIDAR	Light Detection and Ranging	123
LSTM	Long Short-Term Memory	127
MAE	Mean Average Error	46
mAP	mean Average Precision	127
MC	Monte Carlo	127
OOD	Out-of-Distribution	123
ReLU	Rectified Linear Unit	99
RNN	Recurrent Neural Network	36
ROC	Receiver Operating Characteristic	124
ROI	Region of Interest	125
SAE	Society of Automotive Engineers	123
SVM	Support Vector Machine	56
TOD	Teleoperated Driving	1
TPR	True Positive Rate	40
VAE	Variational Autoencoder	26
YOLO	You Only Look Once	10

Symbols

General Symbols

P_{fail}	Failure probability of component or system	$\in \mathbb{R}$
P_{thresh}	Threshold probability for triggering takeover	$\in \mathbb{R}$
N	Number of classes	$\in \mathbb{N}$
c	Semantic class	$\in \mathbb{N}$
P_c	Softmax score of class c	$\in \mathbb{R}$
z_c	Output logit of class c	$\in \mathbb{R}$
l	Semantic class label	$\in \mathbb{N}$
L	One-hot encoded class vector	$\in \mathbb{N}^{N\times 1}$
T	Temperature scaling factor	$\in \mathbb{R}$
P_{cal}	Calibrated softmax score	$\in \mathbb{R}$

System-Level Symbols

F	Car state feature vector	$\in \mathbb{R}^{5\times 1}$
v	Speed	$\in \mathbb{R}$
θ	Steering angle	$\in \mathbb{R}$
a_x	Frontal acceleration	$\in \mathbb{R}$
a_y	Lateral acceleration	$\in \mathbb{R}$
ω	Angular speed	$\in \mathbb{R}$
S	10 s driving sequence	
L	Length of input sample s	$\in \mathbb{N}$
s	Input sample consisting of L vectors F	$\in \mathbb{R}^{5\times \mathrm{L}}$
s_{n_F}	Input sample from failure sequence	$\in \mathbb{R}^{5\times \mathrm{L}}$
n_F	Failure sequence index	$\in \mathbb{N}$
n_S	Success sequence index	$\in \mathbb{N}$
s_{n_S}	Input sample from success sequence	$\in \mathbb{R}^{5\times \mathrm{L}}$
N_F	Number of disengagements	$\in \mathbb{N}$
N_S	Number of successful sequences	$\in \mathbb{N}$

D_{state}	Data set consisting of all input samples s_{n_S} and s_{n_F}	
C_{state}	Classification model for state-based input samples	
P_{state}	Failure probability of sample s	$\in \mathbb{R}$
H	Output filter horizon	$\in \mathbb{N}$
t	Time point within driving sequence	$\in \mathbb{N}$
I_t	RGB image at time t	$\in \mathbb{R}^{224\times224\times3}$
$I_{d,t}$	Difference image created from images I_t and $I_{t-1\,\mathrm{s}}$	$\in \mathbb{R}^{224\times224\times1}$
$I_{dd,t}$	Dynamic difference image at time t	$\in \mathbb{R}^{224\times224\times3}$
I_{dd,n_F}	Dynamic difference image from failure sequence	$\in \mathbb{R}^{224\times224\times3}$
I_{dd,n_S}	Dynamic difference image from success sequence	$\in \mathbb{R}^{224\times224\times3}$
D_{img}	Data set consisting of all dynamic difference images I_{dd,n_F} and I_{dd,n_S}	
C_{img}	Classification model for image-based input samples	
P_{img}	Failure probability of dynamic difference image I_{dd}	$\in \mathbb{R}$
p_i	i-th point in planned trajectory T_t^n	$\in \mathbb{R}^2$
T_t^n	t-th planned trajectory in sequence S_n	$\in \mathbb{R}^{2\times30}$
S_n	Sequence consisting of 100 consecutive planned trajectories T_t^n	$\in \mathbb{R}^{2\times30\times100}$
s^{traj}	Input sample consisting of L consecutive planned trajectories T_t^n	$\in \mathbb{R}^{2\times30\times L}$
$s_{n_F}^{\mathrm{traj}}$	Input sample from failure trajectory sequence	$\in \mathbb{R}^{2\times30\times L}$
$s_{n_S}^{\mathrm{traj}}$	Input sample from success trajectory sequence	$\in \mathbb{R}^{2\times30\times L}$
D_{traj}	Data set consisting of all planned trajectory samples $s_{n_F}^{\mathrm{traj}}$ and $s_{n_S}^{\mathrm{traj}}$	
C_{traj}	Classification model for trajectory-based input samples	
$\theta_{i,traj}$	Angle between the lines $\overline{p_i, p_{i+1}}$ and $\overline{p_{i+1}, p_{i+2}}$	$\in \mathbb{R}$

Component-Level Symbols

S_{semseg}	Baseline semantic image segmentation model
I	Input image
L_{GT}	Ground-truth pixel-wise labels of I
L_{pred}	Labels predicted by S_{semseg} for I
E_{GT}	Ground-truth errors made by S_{semseg} on I
D_{semseg}	Data set for training the baseline model S_{semseg}
$L_{semseg,GT}$	Ground-truth pixel-wise labels of I_{semseg}
$L_{semseg,pred}$	Labels predicted by S_{semseg} for I_{semseg}

N	Number of semantic classes	$\in \mathbb{N}$
S_{intro}	Single-image introspective failure prediction model	
D_{intro}	Data set for training the introspective model S_{intro}	
$L_{intro,GT}$	Ground-truth pixel-wise labels of I_{intro}	
$L_{intro,pred}$	Labels of I_{intro} predicted by S_{semseg}	
$E_{intro,GT}$	Ground-truth errors made by by S_{semseg} on I_{intro}	
I_{test}	Image set for testing the introspective model S_{intro}	
$L_{test,GT}$	Ground-truth pixel-wise labels of I_{test}	
$L_{test,pred}$	Labels of I_{test} predicted by S_{semseg}	
$E_{test,GT}$	Ground-truth errors made by by S_{semseg} on I_{test}	
$E_{test,pred}$	Errors predicted by S_{intro} on I_{test}	
S_{video}	Baseline semantic video segmentation model	
$S_{intro,video}$	Spatio-temporal introspection model	
I_t	RGB input image at time t in video sequence	
$L_{t,GT}$	Ground-truth pixel-wise labels of image I_t	
$L_{t,pred}$	Labels predicted by S_{video} for image I_t	
$E_{t,GT}$	Ground truth pixel-wise error map at time t	
$E_t, pred$	Predicted pixel-wise segmentation error map at time t	
$E_{t+n,GT}$	Ground-truth pixel-wise error map n frames in the future	
S_{future}	Future pixel-wise failure prediction model	

Application Symbols

M	Mask used for erosion and dilation	
k	Number of k-means clusters	$\in \mathbb{N}$
M_{LIDAR}	Monitoring model for LIDAR input	
I_i	i-th image of given data set	
L_i	Pixel-wise semantic labels of image I_i	
S	Semantic segmentation model trained with all images I_i	
P_{pred}	Softmax output of the semantic segmentation model S	
$L_{i,pred}$	Predicted pixel-wise labels of image I_i	
E_S	Error function generated by model S	
E_S^{-1}	Inverse error function of model S	
M_{RGB}	Monitoring model for RGB input	

S_i	i-th sensor of multimodal system	
M_i	i-th monitoring model for sensor S_i	
n	Number of sensors in multimodal system	$\in \mathbb{N}$
R_i	Raw sensor data recorded by sensor S_i	
D_i	Object detection model trained on data R_i	
D_{Fusion}	Fusion-based object detection model trained with all n sensors	
$M_{i,intro}$	Sensor monitoring model for sensor S_i using introspection	
$M_{i,conf}$	Sensor monitoring model for sensor S_i based on confidence score	
$M_{RGB,intro}$	Camera monitoring model using introspection	
$M_{LIDAR,conf}$	LIDAR monitoring model based on confidence score	
P_i	Detection performance of model D_i	$\in \mathbb{R}$
F_i	Features extracted from sensor data R_i	
$P_{i,pred}$	Predicted detection performance of model D_i	$\in \mathbb{R}$
F_{fused}	Fusion of all features F_i	
F_{intro}	Fusion of all features F_i weighted with performance P_i	

1 Introduction

Autonomous systems are increasingly taking over tasks previously performed by humans, ranging from autonomous factory robots [1] to assisted and automated driving [2]. In the context of driving, entirely autonomous systems are not a reality yet. Available automated driving functions are restricted to assistance functions such as lane keeping, cruise control, and automatic emergency braking [3]. Both assisted driving systems and potential future autonomous functions are referred to as a vehicle's Advanced Driver-Assistance System (ADAS). While the progression from assisted to autonomous driving is incremental and fluent, the Society of Automotive Engineers (SAE) has classified the stages from purely manual driving to fully automated driving into six broad levels. A summary of these six levels is shown in Figure 1.1. Current commercial systems are considered to reach Level 2. On Level 3, the car performs all driving and the human is not expected to pay attention to the road, but still needs to be able to take control again if requested. On Level 4, no human driving is needed anymore, but the system is restricted to specific operational design domains. Level 5 describes systems that are capable of driving completely autonomously under any circumstance, removing any need of human interaction.

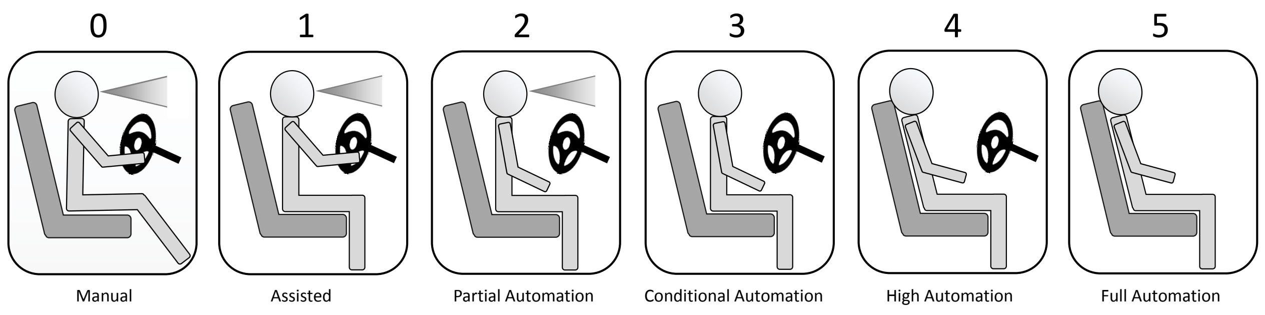


Figure 1.1 Visualization of the six autonomous driving levels as defined by the Society of Automotive Engineers (SAE) [3].

With increasing levels of autonomy, the vehicle is required to make increasingly complex and safety-critical decisions. Despite continuous advances in relevant fields such as machine learning and computer vision, failures in autonomous driving are inevitable. Several studies have discussed public accidents caused by current ADAS functions [4] and the subsequent question of whether automated driving functions are sufficiently safe [2]. Incidents such as the fatal crash of an Uber test vehicle in 2018 driving in autonomous mode serve as constant reminders of how severe failures in autonomous driving can be [5].

A potential solution to resolve situations where the car would make a mistake is to bring a human driver back into the loop. An autonomous vehicle designed to rely on human intervention when needed would be considered Level 3. Its autonomy is conditional on the presence of a human driver. The human can take back control either from inside the car or remotely in the form of Teleoperated Driving (TOD). TOD is especially useful for applications such as autonomous taxi services [6], where human operators inside the car can thus be entirely removed. Regardless of the location of the human driver, this solution requires a mechanism that decides when to switch control from the car to the human. Studies have shown that it can take up to 40 s for human drivers to regain control of vehicles that were previously driving autonomously [7]. Merely reacting to an already challenging driving scenario might not be enough for avoiding a dangerous situation. This leads to a challenging question: How can a failure of an autonomous system be predicted ahead of time?

The focus of this thesis lies on answering that question. While some specific complex situations such as police officers regulating traffic can be anticipated, it is not feasible to enumerate all potentially risky situations. A more general approach is required. In a dynamic task such as driving, detecting a failure after it has occurred can already be too late for ensuring safety. A failure prediction system that can detect failures predictively as well as automatically is therefore desirable. Developing such a system can improve safety by requesting human support early in advance. It can also be used to increase the performance of the system by running countermeasures against a predicted failure, keeping the failure from happening in the first place.

The concept of failure prediction has been approached in various ways before. To better understand the ideas of failure prediction, it is useful to consider why failures occur in the first place. There are three main sources of failures in autonomous driving: epistemic uncertainty, aleatoric uncertainty, and Out-of-Distribution (OOD) data. Epistemic uncertainty, also called model uncertainty [8], refers to uncertainty caused by model imperfections. Insufficient models are a critical failure source in autonomous driving, since the complexity of driving makes designing a perfectly functioning model extremely challenging. Aleatoric uncertainty, also called input uncertainty [8], is a source of failures inherent to the model input. It refers to uncertainties caused by ambiguous or corrupted sensor readings. A perfectly designed model will still not be able to make the correct decisions if its perception of the scene is compromised. Lastly, OOD data describes input that is significantly different to the operational design domain of the system [9]. If the car has not been trained and prepared for the current scene, even a well-designed model with clean sensor input will fail. In this thesis, we mostly consider the first two sources of failures. While it is important to be able to detect OOD input, we aim for predicting the failures inherent to a given model. OOD input can make any model fail and the detection thereof is largely independent of the model itself. Due to its importance, the field of OOD detection is still discussed in this thesis, but not used in the proposed contributions.

In the literature, the implementation of failure prediction is commonly approached using uncertainty estimation. The field of uncertainty estimation offers a wide range of methods to assess how confident a model is in the current situation [10, 12, 11]. The idea of combining Bayesian theory with deep learning led to the development of Bayesian neural networks [13]. Instead of relying on deterministic parameters, the weights of a Bayesian neural network are defined by probability distributions. This allows to obtain an output distribution and thus a variance associated with the prediction [14]. Since a high uncertainty is likely to lead to incorrect decisions and thus an overall failure of the system, an accurate quantification of model uncertainty can be useful. However, there are several issues with using uncertainty estimation to obtain a failure prediction in autonomous systems. Firstly, a low model uncertainty is not guaranteed to avoid failures. A system can be confident and certain, but still wrong in case a challenging or ambiguous traffic scenario is encountered. Secondly, existing uncertainty estimation methods only apply to individual models. While existing methods can detect some failures of individual components, a failure of the entire system consisting of multiple interacting components is not straightforward to predict this way. Only few works exist that explicitly predict the failure of an entire autonomous system. Most notably, Fridman et al. [15] proposed an approach for predicting disengagements of the Tesla autopilot system five seconds in advance at an accuracy of 90 %. However, the Tesla autopilot is a highway-only Level 2 driving function. In contrast, we consider general failures of systems at Level 3 in both highway and urban environments.

Both the task of failure prediction for individual components and for the entire system are addressed in this thesis. In the literature, most related approaches are focused on improving a model to keep it from making mistakes. In this thesis, all contributions are based on the assumption that a given system or model will inevitably make mistakes in practice, regardless of how well designed it is. Following this assumption, we design explicit failure prediction methods that are tailored to the system or model at hand. Then, we use the insights gained from the developed failure prediction methods to correct the system automatically, or adjust it to avoid failures in the first place. We summarize the resulting major contributions made in this work next.

1.1 Major Contributions

This thesis contains three main contributions. They can be combined into a comprehensive failure prediction framework that can be added to a typical autonomous system pipeline as shown in Figure 1.2. The individual failure probabilities P_{fail} obtained from each proposed method can be aggregated and a takeover request by a human driver can be triggered if the aggregated probability crosses a threshold P_{thresh}. If every failure probability stayed below such a threshold, then no human intervention would need to be requested and the autonomous system could continue its functions as before.

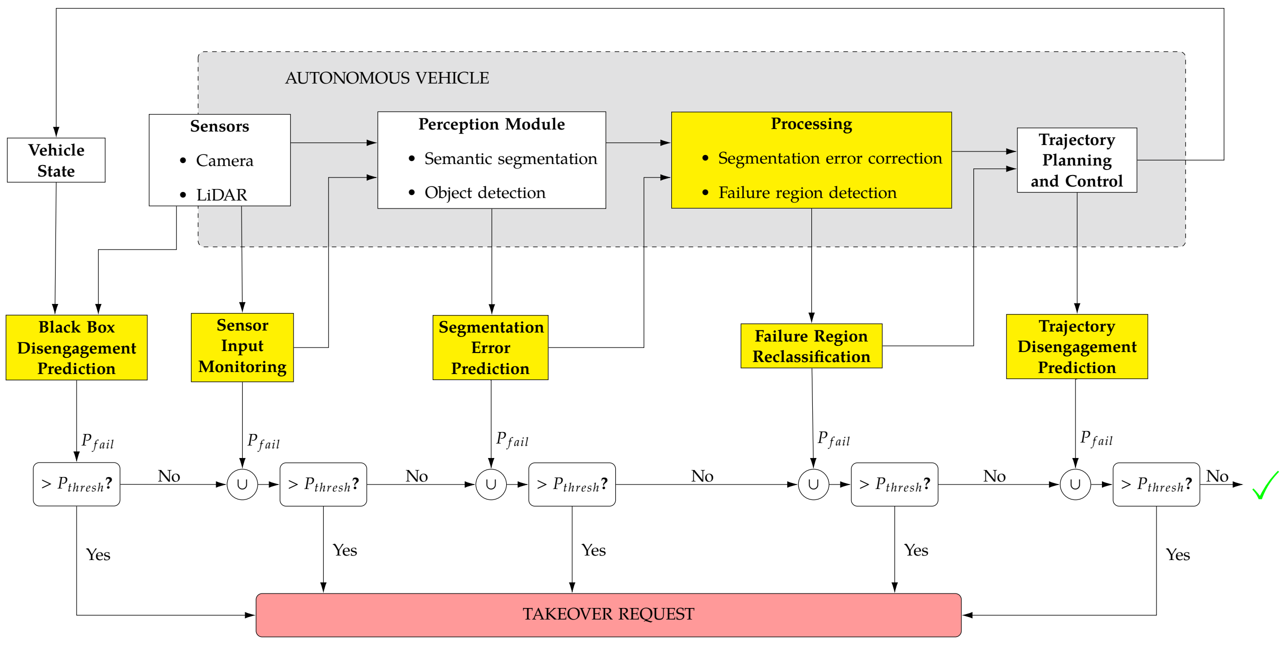

AUTONOMOUS VEHICLE
Vehicle State
Sensors
• Camera
• LiDAR
Perception Module
• Semantic segmentation
• Object detection
Processing
• Segmentation error correction
• Failure region detection
Trajectory Planning and Control
Black Box Disengagement Prediction
Sensor Input Monitoring
Segmentation Error Prediction
Failure Region Reclassification
Trajectory Disengagement Prediction
P_{fail}
$> P_{thresh}$?
No
Yes
∪
✓
TAKEOVER REQUEST

Figure 1.2 Overview of a general autonomous vehicle pipeline and the contributions of this thesis, marked in yellow. Black box and trajectory disengagement prediction is introduced in Chapter 3 and segmentation error prediction is presented in Chapter 4. Failure region detection and reclassification as well as segmentation error correction is discussed in Chapter 5.

The first contribution is focused on predicting system-level failures early in advance. We introduce the concepts proposed for system-level failures in Chapter 3. In the second contribution, we design a failure prediction approach for component-level failures. The proposed methods are summarized in Chapter 4. The third contribution consists of several applications of the failure prediction concepts developed in the first two contributions. They demonstrate that failure prediction can also be used to improve the performance of the system. The components of this third contribution are presented in Chapter 5. Next, we summarize each main contribution in more detail.

1. **System-level failure prediction** System-level failures in autonomous driving lead to disengagements of the automated driving functions, meaning the human driver has to take over again. To predict these critical points in advance, we develop an introspective failure prediction framework. A model is trained with recorded disengagements from six months of test drives in autonomous mode, provided by the BMW group. By training with both the disengagements and successful driving scenes, the model becomes capable of classifying a new scene as either success or failure. To allow early prediction, we use the ten seconds of recordings before each disengagement, as well as ten second sequences of successful driving. Three different sources of information about the scene are investigated as input for the failure prediction model.

- **State-based**: The first implementation of the proposed concept uses sequences of state data as input. Those sequences are labeled as *Failure* or *Success*, depending on whether they end in a disengagement or not. The temporal state vector sequences are then used as training data for a Long Short-Term Memory (LSTM) classifier. During testing, the classifier assigns the current state sequence a failure probability in real time. This black box approach does not pose any requirements on the inspected system and outperforms the state of the art [15, 16] by over 25 %.
- **Image-based**: Next, we consider image sequences as a richer scene representation. Difference images are created across multiple time points to capture spatio-temporal information and then used to train a Convolutional Neural Network (CNN) classifier. While more computationally expensive than the state-based approach, the failure prediction accuracy is significantly higher.
- **Trajectory-based**: Thirdly, we analyze the planned trajectories generated by the inspected system to predict the failures of the system. Since the planned trajectories capture the intended future behavior of the car, they are a useful source of information about future failures as well. Since changes in the planned trajectories can take seconds to result in changes in the car state, this approach outperforms the state-based approach in early failure prediction.
- **Late fusion**: Finally, all three sources of information are combined using a late fusion approach of averaging the three individual failure probabilities. The fusion model outperforms the state-based model by another 13 % and achieves the overall best performance of predicting failures seven seconds in advance, at an accuracy of over 89 %.

2. **Component-level failure prediction** Next, we address failures of individual components, specifically failures of semantic segmentation. The concept of introspection is applied to learn from recorded pixel-wise errors of the inspected model. Based on this concept, we develop failure prediction methods both for the current frame and for future frames.
   - First, pixel-wise failures made for the current input image are predicted. For each training image, the pixel-wise errors made by the inspected model are recorded and used as target labels for an introspective failure prediction model. The failure prediction model therefore performs semantic segmentation of the input image, but assigns each pixel a failure probability instead of a semantic class probability. By learning from the explicit failures made by one specific model, this approach outperforms generic uncertainty estimation approaches such as Monte Carlo (MC) dropout [10] by more than 7 %.
   - With a failure prediction for each frame available, it is possible to extrapolate into the future to predict pixel-wise failures for future frames as well. An autoencoder structure using convolutional LSTMs to capture spatio-temporal features is employed. We use sequences of predicted error maps as training input and the actual error map of a future frame as the target. This approach allows us to accurately predict future pixel-wise failures up to five frames into the future. To evaluate this method on a large-scale data set, we introduce the Densely Annotated Video Driving (DAVID) data set generated in Car Learning to Act (CARLA) [17]. It is around 9 times larger than comparable public video data sets [18, 19] with pixel-wise annotations for every frame.

3. **Failure prediction applications** Finally, we propose three methods to improve system performance based on concepts from the developed failure prediction methods.
   - First, a two-stage approach inspired by human perception is developed. Human perception consists of an initial low-resolution assessment of the entire scene, followed by a second, focused look at each of the most relevant areas [20]. This concept can be applied to computer vision as well. Semantic segmentation corresponds to the first global assessment.

Then, we use the proposed pixel-wise failure prediction to detect failure regions within a semantic prediction. The failure regions are locally reclassified and the failure pixels are updated accordingly. This corresponds to the focused second look of human perception. By reclassifying failure regions, previously overlooked objects such as pedestrians can be recovered.

- Instead of first extracting failure regions and then reclassifying them, the second approach directly learns to reverse the errors introduced by a given model. This is achieved by training an autoencoder with the erroneous semantic prediction from the inspected model as input and using the ground-truth pixel-wise labels as the target. The autoencoder thus learns to correct the mistakes made by the inspected model. When applied to compressed images, the mean Intersection over Union (IoU) can be improved by over 8 % this way.
- The third concept uses failure prediction to monitor each individual sensor in a fusion-based object detection network. By training a single-sensor object detector for each sensor modality, introspection is used to predict the detection performance each sensor can achieve on its own. Then, sensors with a low predicted performance are assigned lower weights in the fusion network to allow the network to focus on the most useful sensor inputs. By predicting which sensors would lead to failures and adjusting the system accordingly, the mean IoU of a state-of-the-art multimodal object detector [21] is improved by 4.6 %.

1.2 Thesis Organization

This thesis is structured as follows. **Chapter** 3 introduces the theoretical background necessary for understanding the contributions made in this thesis and how they were developed. The most relevant related work in the field of failure prediction is summarized. Strengths and limitations of existing methods are outlined to motivate the introduction of the methods proposed in this work. **Chapter** 3 introduces the system-level black box failure prediction approach that allows to predict disengagements of an autonomous system up to seven seconds in advance. Three different input modalities for the proposed failure prediction model are discussed and ultimately combined in a fusion-based approach. Next, **Chapter** 4 presents an approach for predicting component-level failures, specifically the failures of semantic segmentation as one of the key perception models in autonomous systems. Both an approach for predicting the failures of the current scene and an extension for predicting segmentation failures of future frames are proposed. In **Chapter** 5, the insights from the previous chapters are used to design several applications of failure prediction. Two methods for correcting failures of semantic segmentation are shown. A third method preemptively avoids failures made by a multimodal object detection module by adjusting the system based on the predicted failure probability of each individual sensor. Finally, we conclude this thesis in **Chapter** 6. The key results are summarized and the limitations of the proposed contributions are discussed. Based on these results, we make several suggestions for possible future research directions.

Parts of the work presented in this thesis have been published in international, peer-reviewed scientific journals and conferences [22, 23, 24, 25, 26, 27, 28, 29].

2 Background and Related Work

In this chapter, we summarize the theoretical background of this thesis. First, we discuss the key components of an autonomous system and why failures occur. Then, we give an overview of state-of-the-art uncertainty estimation methods, a field that is closely connected to failure prediction. Finally, we summarize methods for predicting future failures before they happen.

2.1 Autonomous Driving Systems

Autonomous driving systems have been actively researched for decades. In 2005, an autonomous vehicle managed to successfully drive the entire 212 km track of the DARPA Grand Challenge for the first time [30]. The advance of deep learning has further sped up development of self-driving systems. Assisted driving functions based on neural networks, such as the Tesla autopilot introduced in 2014 [31], have received significant public attention. In 2017, Waymo launched an autonomous vehicle ride-hailing service [6]. Despite this progress, failures and accidents remain a common issue [4]. The question of how safe autonomous driving can be is an ongoing discussion, with a wide range of factors potentially causing unsafe driving scenarios [2]. Next, we discuss the core components of autonomous systems, the main sources of their failures and how such failures are currently handled in the form of disengagements of the system.

2.1.1 General Pipeline

Liu et al. [32] divide an autonomous vehicle system into three major parts: The software algorithms responsible for localization, perception, planning, and control of the vehicle, the hardware platform where the algorithms are integrated into the physical system, and a backend where data is stored and new algorithms are developed. In this thesis, we focus on the algorithmic part of such a system.

With terabytes of driving data being recorded every day by car manufacturers [33], learning-based algorithms are a promising direction. The tasks of perception, planning, and control can be addressed using end-to-end learning [34]. By training a deep neural network with sensor data as input and the correct steering angle and acceleration as the target output, the task of driving can be approached with a single model. The resulting system is challenging to analyze due to its black box nature. A modular pipeline consisting of different, learning-enabled systems allows to trace failures more easily to their source. Kim et al. [35] were among the first to propose to integrate failure evasion on a system level to ensure graceful degradation of a vehicle, for example switching to other sensors or modules in case one sensor or module fails. McAllister et al. [36] proposed to quantify the uncertainty of each individual component in a system and propagate them to eventually detect a larger problem before it causes a critical failure. While both works highlight the importance of integrating uncertainty awareness into an autonomous system, implementing such failure-aware systems remains an open challenge.

The most critical components where failures can occur are the perception modules of an autonomous system. While planning and control is not a trivial task, fail-safe trajectories that are guaranteed to be safe can be generated in real-time [37]. This only holds for perfect perception of the environment, however. Next, we summarize the core approaches for learning-enabled perception algorithms.

2.1.2 Perception Modules

Deep-learning based perception methods have shown state-of-the-art performance in multiple areas over the last years. Here, we summarize the main concepts of the three fundamental perception tasks of image classification, object detection and semantic segmentation.

2.1.2.1 Image Classification

Image classification describes the task of assigning an image to one specific class, typically with manually assigned labels as the ground truth. The human visual nervous system has been an inspiration for achieving this task with computers as early as 1980 [38]. LeCun et al. [39] demonstrated the potential of convolutional layers, successfully using a Convolutional Neural Network (CNN) for single digit recognition. In 2012, Krizhevsky et al. proposed AlexNet [40], a deeper CNN which achieved state-of-the-art results on ImageNet [41], a data set with 1000 classes and over 14 million natural images. In the following years, architectures such as VGG [42] and ResNet [43] have further improved the performance. Since very deep networks such as ResNet are computationally complex, more lightweight architectures such as MobileNet [44] have been designed to allow real-time classification on mobile devices.

All those architectures work by using multiple sequential convolutional layers to extract features, followed by one or more fully connected layers for classification. A convolutional layer consists of two-dimensional kernels that are convolved with the spatial input. The numerical values of those filters are learned during training using backpropagation, allowing the network to extract the most useful features for classifying the image. Intermittent pooling layers reduce the spatial size of the resulting feature maps, while increasing rotational and positional invariance. An exemplary visualization of this structure is shown in Figure 2.1. A 4×4 kernel with three channels is applied to a simple input image of size 9×7, resulting in a $6 \times 4 \times 3$ feature map. After applying 2×2 pooling with stride 2, the feature map is flattened and fed into a fully connected layer. The first convolutional layers typically extract low-dimensional features like edges and corners. Those low-level features are the foundation for all subsequent semantic interpretation of the input. For classification, this is done by the final fully connected layer of the network consisting of N neurons, where N is the number of classes. The output of the fully connected layer is commonly referred to as the logit vector z. For $N = 2$ classes of "1" and "0", for example, the resulting 2×1 output vector is used to classify the input image as a "1" in this case.

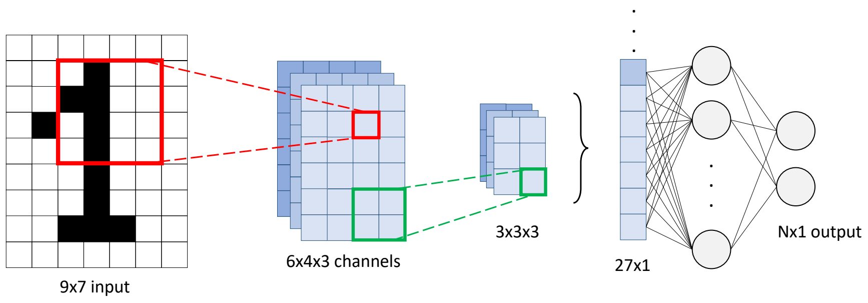


Figure 2.1 Visualization of the main components of a CNN. A kernel (red) slides over the input image to create a convolved feature map. A pooling layer (green) increases spatial invariance. The flattened features are classified by a fully connected layer as one of N classes.

Once such a convolutional architecture is designed, it can be trained with input images and the associated class labels. Training is typically done using the backpropagation algorithm [45]. To compare the output of a neural network to the ground truth during training, a loss function is needed. For classification, a common choice is the cross entropy loss. For this, a softmax operation is applied to the output logit vector z from Figure 2.1 to obtain an output score P_c for every potential class $c \in \{1, N\}$:

$$P_c = \frac{e^{z_c}}{\sum_{k=1}^{N} e^{z_k}} \tag{2.1}$$

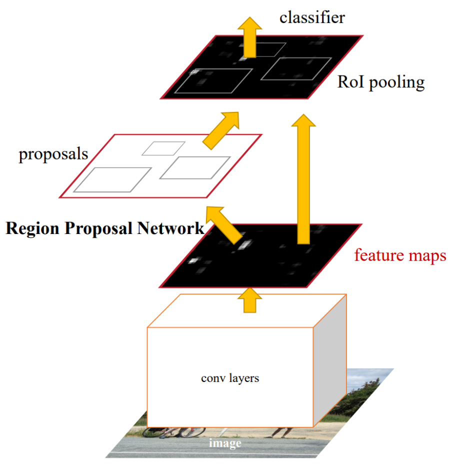
To calculate a loss between these class-wise scores P and the label l of the current image, the scalar class label l is turned into a one-hot encoded vector $L \in \mathbb{R}^{Nx1}$ with $L_l = 1$ and $L_i = 0$ for $i \neq l$. Then, the Cross Entropy (CE) is calculated as

$$CE = -\sum_{k=1}^{N} L_k \log(P_k) = -\log(P_l)). \tag{2.2}$$

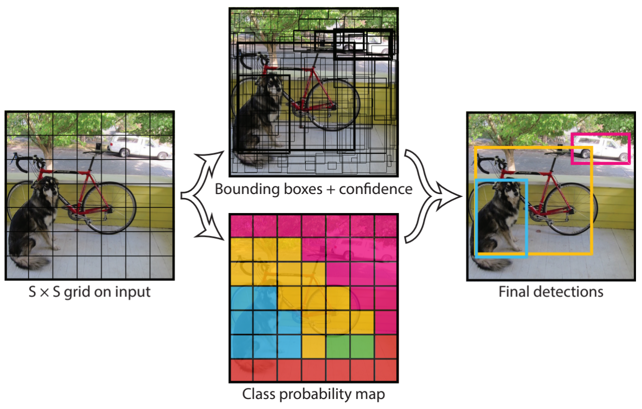
The model's CE for the current training image is used to calculate the gradients using backpropagation, which are then used to adjust the model parameters. After a sufficient amount of iterations, the network ideally converges and is capable of correctly classifying its input images. Its performance can be measured using various evaluation metrics, such as the straightforward accuracy or more complex methods such as precision-recall curves. A summary of common metrics can be found in [46]. Selecting the proper metric for the task at hand is an important part of the development of a perception model. In safety-critical tasks such as pedestrian classification, false negatives can be more important to avoid than false positives, for example.

2.1.2.2 Object Detection

Object detection is an essential part of the perception module of autonomous systems. It goes beyond image classification by classifying multiple regions for one image, while also predicting their locations. In order to localize any number of objects at varying sizes in an image, the CNN framework used for simply classifying an image needs to be extended. Girshick et al. [47] proposed R-CNN in 2014, a two-stage object detection framework that extends CNN classification with an initial region proposal step. In the first stage, 2000 region proposals are generated for a given input image using selective search. In the second stage, a CNN extracts features from those proposals which are then classified. Additionally, the second stage predicts the bounding box of each region that was classified as an object. Fast R-CNN [48] improved this concept by applying convolutional layers first and creating the region proposals from the extracted feature map, significantly reducing the amount of required convolutions. Finally, Faster R-CNN [49] replaced the selective search algorithm with a region proposal network that is trained together with the rest of the model. Its structure is shown in Figure 2.2a. More recent architectures have further improved performance [50], but still use the two stage concept to achieve state-of-the-art object detection.



(a) Faster R-CNN (source: [49]).



(b) Workflow of YOLO (source: [51]).

Figure 2.2 Overview of a two-stage (a) and a single-stage (b) object detection network.

The precision of two-stage detectors is high, but the cost of running two steps is significant. Redmon et al. [51] proposed a single-stage object detector to address this issue, which they named You Only Look Once (YOLO). YOLO performs object detection with a single model by splitting an image into a grid and then generating bounding boxes of variable sizes for each grid cell. By additionally assigning each grid cell a class probability, objects are detected as the bounding boxes with a high underlying class probability. This approach is significantly faster than splitting the detection up into a region proposal and a classification step. An overview of both the two-stage Faster R-CNN detector and the single-stage YOLO model is shown in Figure 2.2.

In the first version of YOLO, only one object could be detected per grid cell, which is problematic if multiple small objects are present. While subsequent improvements like YOLOv3 have addressed this issue [52], the performance of single-stage detectors is still generally lower than two-stage models. For autonomous systems, this trade-off between the desired high speed for a mobile system and the desired high precision for the safety-critical task of driving remains a challenge.

Regarding the evaluation of object detection models, classification metrics such as accuracy are not sufficient anymore. A common metric to evaluate the quality of the predicted bounding boxes is the mean Intersection over Union (IoU). For each predicted bounding box, the area of the intersection with the corresponding ground truth bounding box is computed and divided by the area of the union of the two bounding boxes, as visualized in Figure 2.3. Another popular metric is the mean Average Precision (mAP). It sets an IoU threshold to classify each detection as correct or incorrect and then computes the average precision of all detections. By taking the mean over all IoU thresholds, the mAP is obtained.

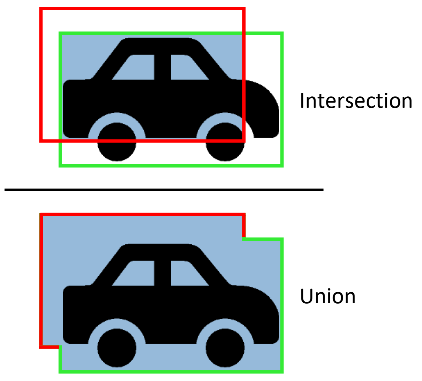


Figure 2.3 Visualization of the Intersection over Union (IoU) metric to evaluate the performance of object detection. The intersection between the predicted bounding box (red) and the ground truth bounding box (green) is divided by their union.

2.1.2.3 Robust Object Detection

When deploying object detectors to real roads, the conditions can be much harsher than in curated benchmark data sets. To ensure high performance, there are several approaches for making object detection more robust. One direction is to improve object detection in the presence of occlusions. While standard two-stage detectors such as R-CNN struggle with partially occluded objects, compositional nets [53] decompose object representations into individual parts and context of each object. An object where one part is occluded can thus still be detected by which parts is visible given the right context, such as a grey area around a car. Wang et al. [54] further improved this approach be explicitly segmenting the objects and their context in each region proposal, avoiding the context to be falsely detected as an object itself. Under heavy occlusions, they outperform the precision of Faster R-CNN by a factor of over 2. Wang et al. [55] addressed a different issue by considering the uncertainty of the ground truth labels of Light Detection and Ranging (LIDAR) data sets. They argued that human annotators draw deterministic bounding boxes even when the available LIDAR scans do not offer

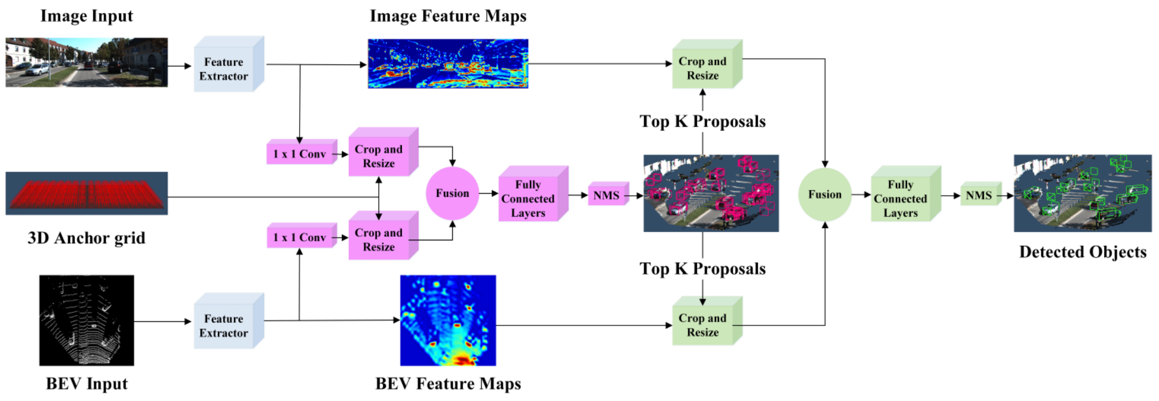


Figure 2.4 Overview of the AVOD architecture that is later used in this work. Features extracted from camera and LIDAR input are fused by either addition or concatenation (source: [21]).

precise information about the location of objects. They proposed a generative model that predicts bounding box uncertainty based on the shape and structure of the corresponding LIDAR point clouds. Training with the resulting probabilistic bounding boxes results in object detectors that are more robust to noisy LIDAR input. Besides occlusions or noisy labels, object detectors can also be susceptible to adversarial attacks. Adversarial attacks generate perturbations to the input image which make the attacked model misclassify the input. Cars equipped with state-of-the-art object detectors can misclassify road signs where inconspicuous adversarial patterns have been added, demonstrating the danger of such attacks [56]. Zhang et al. [57] addressed this by proposing adversarially robust detection, performing adversarial training based on both localization and classification loss for all detected objects.

2.1.2.4 Fusion-Based Object Detection

Another way of improving the performance and robustness of object detectors is to use more than one sensor modality. While cameras are comparatively cheap with rich sensor readings, they struggle in low light conditions or in adverse weather like rain or snow. Adding further sensor modalities to an object detection framework can mitigate those weaknesses. LIDAR (Light Detection and Ranging) sensors are more robust against adverse light or weather conditions. They scan the environment with laser beams that measure the distance to objects around the sensor, creating 3D point clouds that offer depth information, which regular cameras do not provide. Object detection can be performed using only LIDAR input, for example, by adjusting originally camera-based architectures such as YOLO [58]. However, fusing LIDAR with other sensors such as cameras generally improves performance and increases redundancy [59].

The most promising fusion approach for object detection is deep fusion [60], since state-of-the-art detectors typically rely on deep neural networks. Fusion can be performed by adding or concatenating features extracted from the various sensors used. Early fusion combines the sensors after the first convolutional layers, while late fusion only combines high-level features shortly before a prediction is made. Different studies have shown both early fusion [61] or late fusion [62] to perform better. For fusing 2D data from cameras and 3D point clouds from a LIDAR, a common approach is to use Bird's Eye View (BEV) representations of the LIDAR scans [21]. The 2D image features can be projected into BEV [63], or the 3D point clouds can be projected onto a plane [64]. Some state-of-the-art architectures such as Aggregate View Object Detection (AVOD) have more than one fusion step, first fusing low-level features to create region proposals and then fusing the region proposals from both low-level feature maps to classify the objects. As a representative of multimodal fusion that we also use in our work, we show the structure of AVOD in Figure 2.4.

While using multiple sensor modalities is inherently more robust than using a single sensor, the fusion strategy employed can be further optimized for robustness against corrupted sensor input.

Pfeuffer et al. [65] investigated the impact of adverse weather conditions on multimodal object detection and concluded that late fusion strategies perform the best. In early fusion networks, compromised features extracted from noisy or occluded sensor readings propagate throughout almost the entire architecture. A single corrupted sensor input is less likely to be harmful if the feature streams are kept separate for longer. Instead of finding the best fixed architecture, Mees et al. [66] proposed an adaptive multimodal fusion model for changing environments. They trained a separate model for each type of environment and combined them using late fusion. While this strategy improves performance in challenging lighting conditions, the model complexity increases significantly and training data from each expected adverse condition is required. Finally, Kim et al. [67] increased robustness against noise in single sensors by using a convolutional layer for fusion. By adding noisy data to the training set, the fusion layer learns to select the best subset of channels from the mix of unchanged and corrupted input.

2.1.2.5 Semantic Segmentation

While object detection only classifies some regions per image, semantic image segmentation classifies every pixel in an image. For autonomous vehicles, this allows for a much denser scene understanding compared to object detection, for example enabling free space computing to plan the car's trajectory. The main difference to classification or detection is that the output remains entirely spatial. To achieve a spatial semantic prediction for an image input, Long et al. [68] proposed fully convolutional networks. A sequence of convolutional layers with intermittent pooling layers results in spatial feature maps. Upsampling the final feature map to the original resolution results in a semantic prediction. Instead of simply upsampling the last convolutional layer, Badrinarayanan [69] proposed a dedicated decoder to gradually increase the resolution of the encoded feature map with multiple transposed convolutional layers. To avoid reducing resolution during the convolutional layers, atrous convolutions can be used [70]. Atrous convolutions increase the receptive field without reducing resolution, allowing to aggregate multi-scale contextual information. The DeepLab architecture proposed by Chen et al. [71] further introduced an Atrous Spatial Pooling Pyramid (ASPP) to semantic segmentation models, where feature maps are filtered at different sampling rates to capture information at multiple scales. Finally, the concepts of atrous convolutions, ASPPs and a decoder to refine the feature map output by the encoder were combined in the DeepLabV3+ architecture [72]. We use DeepLabV3+ as a state-of-the-art semantic segmentation architecture in this thesis. Its structure is shown in Figure 2.5. For evaluation, the same metrics as for object detection can be used, most commonly the mean IoU and mAP. The insights from semantic image segmentation can be applied to other sensor types as well. For LIDAR data, architectures such as RangeNet++ [73] achieve state-of-the-art segmentation performance by projecting the LIDAR point clouds into 2D and then using a fully convolutional network, similar to the networks used for image segmentation.

2.1.2.6 Postprocessing for Semantic Segmentation

Noisy input or model inaccuracies can lead to erroneous semantic predictions. While one solution for noisy input is to use denoising models on the input directly [74], another recent approach is to postprocess the semantic prediction to correct some of the mistakes. Larrazabal et al. [75] proposed to use a Denoising Autoencoder (DAE) to integrate anatomical priors in the task of medical lung segmentation. They add manual degradations like erosions, dilations, and occlusions to the ground truth semantic map and then train a DAE to reconstruct the original ground truth labels from the noisy input. The resulting model thus learns to create an anatomically reasonable semantic map even from a highly irregular initial prediction. The approach can also be extended to multi-class segmentation [76]. The concept of denoising a semantic prediction is a potential way of increasing the performance of an arbitrary segmentation model. Only the output of the model is required and the segmentation model itself does not need to be changed.

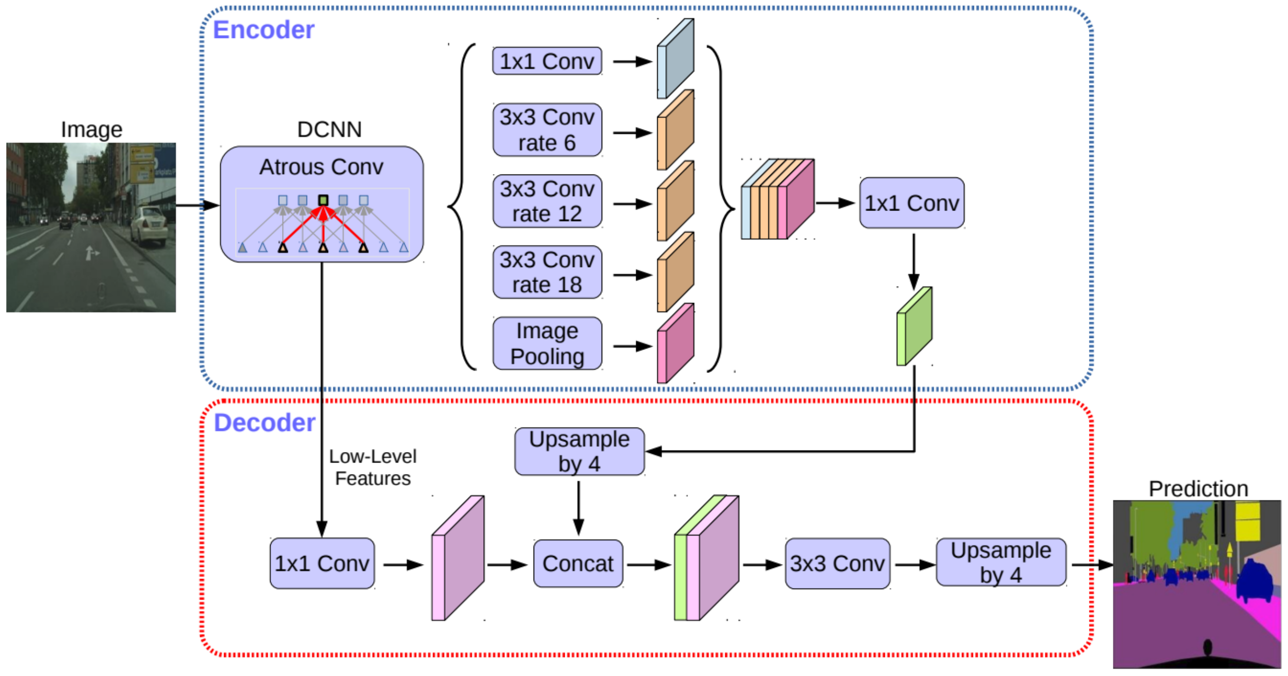


Figure 2.5 Overview of the DeepLabV3+ architecture for semantic image segmentation (source: [72]).

2.1.2.7 Future Semantic Segmentation

In dynamic tasks such as driving, being able to anticipate events before they occur is a valuable capability. Since semantic segmentation can be the foundation of tasks such as free space computing, knowing how the semantic segmentation of a scene will look in the future can improve safe trajectory planning. To extrapolate into the future, video data is required. Pfeuffer et al. [77] performed semantic video segmentation using convolutional Long Short-Term Memory (LSTM) cells [78] to extract spatio-temporal features, thus improving the frame-wise segmentation. To predict the segmentation of future frames, one approach is to generate the future image input using a Generative Adversarial Network (GAN) [79]. The generated future image can be segmented using a regular video segmentation model to obtain a predicted future segmentation. Instead of predicting future images and segmenting them, Luc et al. [80] showed that directly predicting future segmentation maps performs better. They use an autoregressive CNN to predict the next segmentation using the previous four segmentations. By using the future prediction as a new input to their model, they predict up to three frames into the future in an autoregressive fashion. This approach was further improved by Rochan et al. [81], who employed convolutional LSTMs in an encoder-decoder architecture to predict future segmentations. Their architecture is shown in Figure 2.6. They encode four sequential frames at four encoding depths and feed the encoded sequences into four convolutional LSTMs. The resulting spatio-temporal feature maps are then decoded into the predicted next semantic map. This demonstrates the potential of convolutional LSTMs, which can be used both for improving the segmentation of the current frame by using image sequences as input as well as predicting future segmentations by using sequences of semantic predictions as input.

2.1.2.8 Data Sets

All deep learning models discussed so far require large scale training sets to achieve useful performance. For image classification, the ImageNet data set [41] remains one of the most important data sets available. It consists of over 14 million images from 1000 different classes. While image classification alone is not sufficient for complex tasks such as driving, models trained on ImageNet are commonly used as backbone networks for both object detection and semantic segmentation. Low-

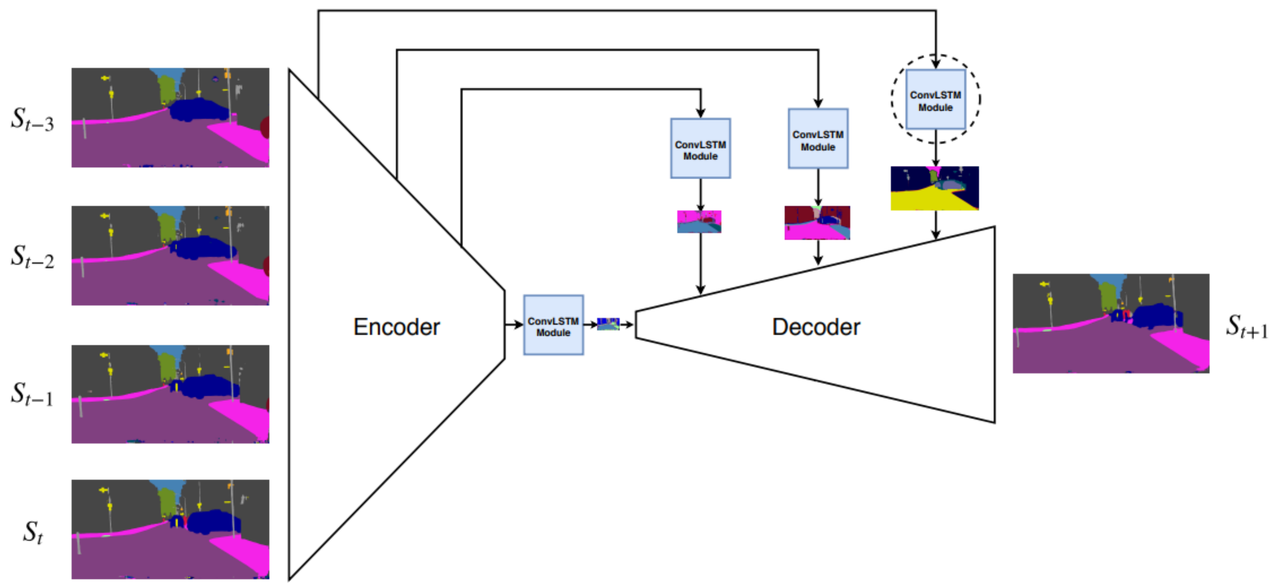


Figure 2.6 Overview of an encoder-decoder architecture with convolutional LSTMs that predicts future semantic maps based on a sequence of the previous four semantic predictions (source: [81]).

level features like edges and simple shapes are similar across many domains, making the first layers of classification architectures trained on ImageNet useful in most specific applications such as driving as well. For object detection, the KITTI data set [82] is one of the most popular benchmark data sets. It consists of over 15 000 images plus corresponding LIDAR scans, with bounding box annotations from seven object classes. It was collected on urban roads in Germany, making it useful for training object detectors intended for autonomous driving. For the LIDAR scans, the semanticKITTI data set also provides point-wise semantic labels [83].

For image segmentation, Cityscapes [84] is one of the most popular data sets. It contains 5000 images with pixel-wise semantic labels from 19 classes. Around ten sequential frames before and after each labeled image are available as well. All images were collected in multiple German cities, making it a realistic and challenging benchmark for autonomous driving. While Cityscapes remains popular, larger data sets have since been released. The BDD100k data set [85] contains 10 000 images with semantic labels collected on both highways and urban roads in the United States. More recently, the A2D2 data set [86] with over 41 000 images with semantic labels from 38 classes was released, including both highways and urban roads from Germany. For tasks that require sequential images with labels, there are only few data sets available. The CamVid data set [18] contains video sequences with frame-wise semantic labels at a frequency of 1 Hz, but is limited to around 700 frames in total. The Highway Driving data set [19] offers pixel-wise labels at 30 Hz, but is still limited in size at a total of 1200 frames. In contrast, driving simulators are an almost limitless source of labeled sequential data. The Car Learning to Act (CARLA) simulator [17] offers pixel-wise semantic labels for every frame. While the gap between simulation and real world data remains a challenge, multiple works have successfully deployed models trained on simulation data to the real world [88, 89, 87]. In this thesis, we introduce a new large-scale video data set with pixel-wise labels for every frame recorded in CARLA in Chapter 4, being around 10 times larger than the largest comparable publicly available data set.

Finally, for evaluating models designed for autonomous driving, data collected by self-driving vehicles is a valuable source of information. The Waymo Open data set [90] is currently the only public data set recorded with self-driving cars and contains 1150 sequences of 20 s of annotated camera images and LIDAR scans. However, it only contains data from successful driving without interventions, making it less suitable for this thesis' focus on detecting failures of autonomous vehicles.

2.1.3 Sources of Failure

Having introduced the main modules of an autonomous vehicle, we next summarize what the sources of failures in such a system are. In this thesis, we focus on failures that occur due to the internal logic or software of a vehicle. There are other straightforward sources of failure that we do not further discuss here, such as malfunctioning hardware. Such issues can be detected using watchdog systems [91], which are already commonly used to monitor the hardware side of autonomous systems. Concepts from predictive maintenance [92] also contribute to ensuring the physical functionality of all components. While such hardware-related failures need to be detected as well, we focus on failures that occur in a physically intact system. Such failures are generally more challenging to predict and resolve than a broken sensor or actuator.

The most critical type of software-related failure is when the system's behavior does not ensure safe driving anymore. This is caused by one or more individual components failing. Those components include the models for perception, planning, and control. We distinguish between system-level failures and component-level failures. If one or more individual components fail, for example by misclassifying an object or planning an unsafe trajectory, a system-level failure can occur in which a human operator has to take control of the vehicle again. Since component-level failures are a main reason for system-level failures, we focus on failures of individual models first.

There are three main reasons for individual components to fail: epistemic uncertainty, aleatoric uncertainty, and Out-of-Distribution (OOD) data as visualized in Figure 2.7. We briefly summarize all of them in the following.

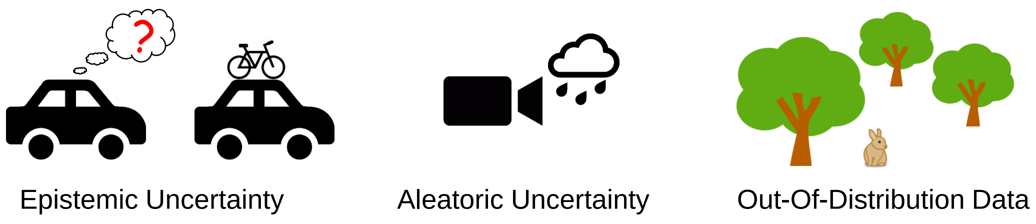


Figure 2.7 There are three main sources of failures in autonomous driving. Epistemic uncertainty stems from an insufficient model. Aleatoric uncertainty is inherent to the input. Finally, Out-of-Distribution (OOD) data differs significantly from the training set distribution.

2.1.3.1 Epistemic Uncertainty

Epistemic uncertainty is also referred to as model uncertainty [8]. It describes the uncertainty of a model's output that stems from an imperfect model. With more training data or a model with more capacity, such uncertainties could in theory be resolved. Figure 2.7 shows an example from the context of driving. Even with clear sensor input, a bicycle transported on top of a car can be a challenge for models that were trained with separate instances of cars and bicycles. Adding images of such combinations of cars and bicycles to the training set would reduce those uncertainties. Failures of a model that can be traced back to epistemic uncertainty could therefore be prevented by more carefully designing the model or by increasing the training set.

2.1.3.2 Aleatoric Uncertainty

In contrast, aleatoric uncertainty is inherent to the input a model receives. It is therefore also called input uncertainty [8]. Such uncertainties are caused by noisy, occluded, or ambiguous input. Figure 2.7 shows a typical example from driving. Rain drops on a camera lens can lead to compromised input that is impossible for any model to understand correctly. While training models with data from such adverse weather conditions has been shown to improve robustness [65], such augmentations of the training set only reduce the epistemic uncertainties of the scenario. Aleatoric uncertainty refers to the uncertainties than cannot be removed regardless of how the model is refined. While the argument can be made that with the right training data or a sufficiently large model, any uncertainty could be resolved [8], accepting that some uncertainties remain irreducible is more useful for practice. While epistemic uncertainties should be reduced as much as possible during model development, aleatoric uncertainties cannot be completely removed by improving the model. To anticipate this, separate mechanisms need to be employed for detecting them.

2.1.3.3 Out-Of-Distribution Data

Finally, the reason a model fails can be that it is deployed to an environment it was not designed to operate in. The input data from a novel environment that differs significantly from the training environment is referred to as Out-of-Distribution (OOD) data. If a model encounters data it was not trained for, its output will be inherently uncertain. An example is shown in Figure 2.7. If an autonomous car is intended to operate on highways and its training set does not include animals on the road, then encountering such objects will lead to OOD data. For high-dimensional image data, it is challenging to define the distribution that underlies a training set [93]. In the literature, test data from a data set that the model has not been trained with is sometimes referred to as OOD data [94]. This can be misleading, however, since two different data sets might still share common objects or structures. More recent works such as Yang et al. [9] offer more precise definitions of anomalies, outliers, novelties, and OOD data. In the context of driving, encountering object classes not present in the training set remains one of the most critical examples of OOD data.

2.1.4 Disengagements

If one or more component-level failures happen, the overall system eventually can fail as well. Compared to tasks such as object detection where a ground truth to compare to is available, it is harder to quantify when a system-level failure occurs. While the car crashing or violating traffic rules is definite proof that a system-level failure has occurred, the system was likely already not operating in a safe manner in the time before a collision. Therefore, the moment the car is not driving in a safe or controlled way anymore can be seen as the point of failure. To prevent a system-level failure from causing harm, autonomous systems are designed to return control to the human operator early enough to avoid dangerous situations. This is referred to as a disengagement of the autonomous system. Besides the system triggering disengagements automatically, the second source of disengagements is a manual take-over by a human driver who decides the situation is not safe anymore. A general overview of accidents and disengagements of autonomous test vehicles by commercial companies can be found in [4]. The consistent presence of accidents demonstrates that existing systems cannot avoid mistakes even with human safety drivers present. Next, we further discuss the role of humans in the disengagement process and why accidents can still occur.

2.1.4.1 Human Takeover

In case the autonomous system disengages, humans remain the main way of resolving the situation. There are two ways for a human to take over control again. First, the car can be controlled via Teleoperated Driving. Kang et al. [95] discuss the requirements and challenges of using human operators that are not physically present in the car. They mostly discuss the latency issues that arise from remotely sending control input, arguing that dedicated network and real-time streaming methods are required. Beyond low-latency video transmission, another open research question is how to achieve a sufficient situation awareness to safely steer the vehicle from afar. Georg et al. [96] designed an immersive interface to optimally represent all sensor data to the human, going beyond only showing camera streams on displays. Another option is to enhance displays by adding predictive information about the future motion of the vehicle, or by adding haptic feedback [97].

A second option is to still rely on a safety driver inside the vehicle. A takeover from within the car results in regular driving, removing the need for novel interfaces or low-latency transmissions. However, asking a human driver to take control of a vehicle that was previously driving autonomously is still a challenge. Merat et al. [7] investigated transitions from automatic to manual driving in a driving simulator. They found that it can take up to 40 seconds for a human driver to completely regain control of the car if the human had not been paying attention to the road beforehand. If the driver expected the disengagement, control was regained more quickly. The capability to predict disengagements early in advance therefore is a desirable property to increase overall safety of autonomous vehicles. In case a disengagement is predicted early, the system can then alert and prepare the driver.

2.1.4.2 Takeoverability

The concept of takeoverability describes how well prepared the driver is for an impending takeover maneuver. While an early prediction of a disengagement is useful, a disengagement will not increase safety if the driver is not ready for taking control yet. Studies show that the vigilance of human drivers in automated vehicles steadily decreases during longer drives [98]. To address this, Xia et al. [99] predict the driver's attention by learning from recorded gaze and object detection data. By comparing the predicted attention with the current gaze data, the situation awareness can be estimated [100]. Besides evaluating the current awareness of the driver, there are also approaches to increase the takeoverability in the seconds before a takeover request. Disengagements are typically caused by the car not being able to handle some part of the driving situation. Thus, the focus of the car can be an indicator of where the human driver should also focus on in order to resolve the challenging scenario. Bojarski et al. [101] used backpropagation to calculate which parts of an input image a CNN was most influenced by. This allows to visualize the most relevant areas of the input from the perspective of the car's perception module. Highlighting those regions can give the human operator a first understanding of why the system disengages and allows them to prepare for that specific situation. Similarly, class activation maps [102, 103] can be used to visualize which parts of an image led to the prediction of a classification model.

Finally, the driver can be monitored by the car to assess their takeoverability. The driver's physical state such as their heart rate or stress level can be monitored [104]. Additionally, the reaction of the driver to a takeover request gives insight into their ability to perform the takeover. Herzberger et al. [105] found that the lack of a noticeable orientation reaction such as scanning the traffic situation indicates a lack of takeoverability. Prompting such a reaction using visual or acoustic alerts can increase the readiness to take control of the vehicle again. This again highlights the need for an early prediction of impending disengagements.

2.2 Uncertainty Estimation

One of the most relevant research areas that enable failure prediction is uncertainty estimation. Quantifying the uncertainty that a system is exhibiting is the foundation both for component-level failure prediction and, by aggregating component-level uncertainties, for system-level failure prediction. If the uncertainty of a system is available, a straightforward thresholding approach is sufficient to predict a failure or success of the system.

Autonomous vehicles are an important application of uncertainty estimation methods. Arnez et al. [106] gave an overview of state-of-the-art uncertainty estimation for this area. Many machine learning models offer an inherent confidence score, such as the class-wise scores P introduced in Section 2.1.2. However, such scores do not reflect actual probability estimates [10]. Additionally, minimizing the cross entropy loss as outlined in Section 2.1.2 inherently encourages the model to assign the highest possible score P to its prediction, since a score of 1 minimizes the loss in case the prediction is correct. The field of uncertainty estimation is focused on generating uncertainty values that are more closely correlated to the actual success rate of the system. This is an important concept for failure prediction.

A high uncertainty estimate suggests that the system should not be trusted, even if the system's decision turns out to be correct. Accurately quantifying uncertainty allows a model to know when it does not know what to do. This knowledge can be used to safely trigger a takeover request before a dangerous situation can develop.

As already visualized in Figure 2.7, the literature distinguishes between epistemic and aleatoric uncertainties [8]. Epistemic uncertainty refers to model uncertainty caused by the imperfections of the given model. Any epistemic uncertainty could, in theory, be resolved with more training data or a more sophisticated model. In contrast, aleatoric uncertainties cannot be reduced with more data or a larger model. This includes any issues inherent to the model's input, such as sensor noise or occlusions. There are several studies discussing how to address the uncertainty of machine learning models. Ghahramani [107] introduced a general probabilistic machine learning framework to incorporate uncertainties. Wang et al. [13] further proposed a similar framework for Bayesian deep learning, incorporating Bayesian probability modeling into deep learning architectures. Kendall et al. [108] discussed the relevancy of different types of uncertainties in the context of Bayesian deep learning, arguing that Bayesian methods are an important step towards modeling epistemic uncertainties. While the distinction between epistemic and aleatoric is useful for describing which uncertainties can be resolved, failure prediction requires accurately quantified uncertainty regardless of which type the uncertainty is. A more relevant distinction is where the uncertainties can be detected in a model workflow. Specifically, we distinguish between methods that analyze the model input and methods that investigate the model output. Assigning an uncertainty to the model's output is the more common approach in the literature. However, a model can be confident in its output and still be incorrect. Analyzing the input allows estimating whether the model should be allowed to make a prediction in the first place. We therefore begin by discussing methods based on input analysis.

2.2.1 Input Analysis

Sensor inputs such as camera images or LIDAR point clouds are a key component of how an autonomous vehicle perceives its environment and on what it bases its decisions. By analyzing the input immediately after the sensor has recorded it, it is already possible to predict that a subsequent model or system will likely not make a correct prediction. By estimating the uncertainty of a model before even evaluating it, the required time for a potentially expensive model inference can be saved. It should be noted that OOD input that is new to a model by definition. For such input, a model is always uncertain. In this section, we focus on input that is part of the operative design domain, but nevertheless leads to failures. Unless a model performs at 100 % accuracy, such inputs inevitably exist. There are two main approaches for detecting that a given input will be problematic for a system to handle. First, a hardness predictor can be trained jointly with the model, learning which input caused incorrect outputs during the training process. Second, an input analysis model can be designed separately from the inspected system. For this approach, information about model failures is obtained from recording model predictions on a training set and then learning which inputs caused incorrect outputs. This concept of learning from recorded failures is referred to as introspection and will be a core foundation of several contributions of this thesis.

2.2.1.1 Hardness Prediction

First, we summarize two approaches that incorporate a hardness predictor into the training process of the inspected system. A hardness predictor is a model that predicts how difficult the current input is to classify correctly. Wang and Vasconcelos [109] proposed to train a hardness prediction model jointly with the model itself. The additional model assigns a hardness score to each training sample based on the accuracy of the current prediction of the inspected model. Throughout the training process, input samples with higher hardness scores are assigned higher weights when computing the training loss. Due to the higher loss, the model is adjusted more strongly to improve its predictions on those hard samples. Conversely, those samples will be assigned a lower hardness score during the next update of the hardness predictor. Once training is complete, only the most challenging inputs still receive high hardness scores, which correspond to high estimated model uncertainty. By evaluating the hardness predictor first, highly uncertain inputs can be rejected preemptively before running the inspected system.

Yoo et al. [110] proposed a similar concept. They trained a second model to predict the training loss a new input would generate for the inspected model. While this captures similar information to using the current training accuracy as done by Wang and Vasconcelos [109], it can assign different scores to two predictions that are both wrong. A higher training loss means that the prediction is not just wrong, but also further away from being correct than an incorrect prediction with a lower training loss. Additionally, their approach can be applied to multi-task networks as long as they are trained using a single loss. Training the loss predictor alongside the inspected model allows for efficient joint training. However, both approaches are only applicable to individual neural networks where the training process can be modified. In complex systems such as autonomous vehicles, not all components are necessarily neural networks. To keep all components as modular as possible, failure prediction models should ideally not require any changes to the training process of the system itself. A failure prediction approach that operates only on the system input while treating the system as a black box is introspection. We summarize this powerful concept as well as several recent works based on it next.

2.2.1.2 Introspection

Instead of using the training accuracy to predict the hardness of an input, a more general and flexible approach is to take a set of inputs and record the performance of the given system for each input. Those inputs can be either the training set or a separate set reserved for this purpose. By recording the input-performance pairs of the system as it will be deployed in practice, a failure prediction model can be trained to predict when the system will make mistakes. The concept of learning from previous failures to be able to predict future ones is referred to as *introspection* [111].

The task of an introspective failure prediction model is to map each input to the corresponding performance of the inspected system. One of the first works that applied this approach to vision-based systems was Zhang et al. [112]. They argued that when a model fails to generate the correct output, it can still be possible to correctly predict that the model will fail for the given input. Being able to predict which outputs are likely incorrect can help mitigate the overconfidence often exhibited by discriminative classifiers. They propose a framework based on an arbitrary vision-based model and an image data set. First, they generate training data by extracting handcrafted features from each image. By evaluating the inspected vision-based model for each training image, they assign binary training labels of *Success* or *Failure* to each image. Then, they train a Support Vector Machine (SVM) to predict the label based on the current image features. In their experiments, this approach allowed them to reliably predict failures of a variety of computer vision tasks based solely on the input, for example by learning that images with large brightness contrasts are more likely to be classified incorrectly.

Daftry et al. [111] continued this line of research, but advocated the use of deep learning instead of using hand-crafted features. They were the first to refer to the concept of learning from previous failures as *Introspection*. They generalize the concept from predicting failures of a classifier to predicting failures of an entire system, in their case an autonomously flying drone. They record flights of the drone through a forest and record the video data leading to crashes as well as successful flying sequences. Then, they use both a spatial and a temporal CNN to extract features from the recorded image sequences. Those features are used to train a SVM to again predict a binary label of *Success* or *Failure* for each sequence. They applied their resulting introspective failure prediction model to their drone and used it to reject inputs with a high predicted failure probability, instead of simply stopping the drone before resuming flight. This significantly increased the time the drone could fly autonomously. Since they showed that introspection allows to successfully predict failures of an autonomous system, we use ideas of their work in multiple of our contributions.

The approach from Daftry et al. [111] was later extended by Saxena et al. [113] by additionally training a model to select a suitable recovery maneuver once a failure is detected. This modification further increased the amount of time flown in autonomous mode. They also noted that the failure prediction scores could be used to automatically detect challenging scenarios in test flights, allowing to efficiently obtain relevant new training data.

Another application of introspection is to improve obstacle detection, proposed by Rabiee et al. [114]. They extend image-wise introspection to region-wise introspection. They predict which regions of an image are incorrectly classified regarding whether they contain an obstacle for a mobile rover. Finally, Yang et al. [115] applied introspection to object detection. They train an introspective model to predict where in an image the inspected object detector missed objects. Their model learned to detect false negatives on the KITTI data set at both a precision and a recall of over 81 %. Considering that incorrect obstacle detection and false negatives are both critical dangers to autonomous driving, these recent works further demonstrate the potential of introspection for failure prediction in autonomous systems.

2.2.2 Output Analysis

Analyzing the input of a system to estimate its uncertainty does not require running the entire system before making a failure prediction. The output of the system is only needed during the training of the input analysis models. While this approach can save time, it does not make use of all of the information available about the current situation. Taking the time to obtain the model output yields the actual behavior of the system for the current input. Since this behavior is what would eventually cause the failure, analyzing the system output is the second main way of achieving failure prediction. In the literature, estimating the uncertainty of a model's output is the most common way output-based failure prediction is approached. While it depends on the system's design and robustness what levels of uncertainty actually result in a failure, having an accurate estimate of the uncertainty is a useful first step. One way the uncertainty estimation can be performed is by modeling it as a probability. Thus, a large focus in the literature is on using Bayesian probability principles for estimating the probability that a model is correct. Since Bayesian approaches do come with some practical downsides, a second research area focused on non-Bayesian approaches for quantifying the uncertainty of a model has also been developed. In the following subsections, we summarize the most relevant works from both research directions.

2.2.2.1 Bayesian Methods

Many machine learning models already generate a probability-like confidence measure alongside their prediction. Commonly, the softmax score from Equation 2.1 is used as the model's confidence. However, the softmax score is not an actual probability and does not describe model uncertainty as discussed in works such as [116] or [10]. The softmax score can still be useful by modifying it to match the model uncertainty more closely. The goal is for the softmax probability of a prediction to correspond to the frequency that the prediction actually occurs. The similarity of the predicted probability and the observed frequency is referred to as the calibration of a model. Guo et al. [117] evaluated a range of calibration techniques and found that Platt temperature scaling is an efficient way of obtaining a calibrated output score P_{cal} of a model. Equation 2.1 only needs a minor modification of inserting a temperature scaling factor $T > 1$ into the exponential:

$$P_{cal} = \frac{e^{z}/T}{\sum_{k=1}^{N} e^{z_k/T}} \tag{2.3}$$

They showed that this scaling effectively addresses the overconfidence inherent to softmax scores. A large factor T reduces overly large softmax scores without affecting the accuracy, since the relative order of class scores stays the same. Kuleshov et al. [118] further proposed to train a dedicated regression model to calibrate the softmax score of a model, improving calibration at the cost of increased model complexity. While calibrating the model output is a fast way of obtaining a simple uncertainty estimate, the field of Bayesian deep learning offers a much wider range of approaches. There are several papers outlining the theoretical advantages of Bayesian deep learning compared to traditional networks [13, 108]. Traditional neural networks generate a point estimate output, whereas Bayesian neural networks are designed to generate an output distribution. In general, this can be approached by modeling both the output and the weights of the architecture as probability distributions. A straightforward way to implement this idea is *Bayes by Backprop*, proposed by Blundell et al. [14]. They assign each weight a Gaussian distribution and then learn the mean and standard deviation for each weight during training. By adding one additional parameter per weight, the resulting model then consists of many consecutive distributions instead of scalar point estimates. For inference, the authors proposed to sample multiple times from each distribution to obtain a set of different scalar models. By generating the output from the resulting set of models, an output distribution is generated. The variance of this output distribution can then be used as an estimate of the model's uncertainty.

A major downside of actually learning distributions instead of scalar weights is the significantly increased model complexity. The number of parameters is doubled and the optimization becomes more challenging due to the more complex loss function. In practice, deep neural networks are already highly complex and require significant resources just to perform their main task, such as classification or object detection. Dedicating the same amount of resources to the secondary task of assigning an accurate uncertainty estimation to the model output may not be feasible. Gal and Ghahramani [10] proposed a much less complex alternative to Bayes by Backprop that achieved comparable results. They termed their approach Monte Carlo (MC) dropout. Dropout is a generalization technique where some neurons are set to zero during training. This prevents the network from relying too much on specific neurons [119]. Gal and Ghahramani proposed to keep dropout active during testing. They showed that performing multiple forward passes through a neural network with active dropout layers is an approximate Bayesian inference of a Gaussian process. Effectively, each different dropout mask corresponds to sampling a new model from a distribution of models. The set of samples obtained from inferences with different dropout masks then corresponds to an output distribution similar to Bayes by Backprop. Then, the variance or entropy among the output samples can be used to obtain an uncertainty estimate. A visualization from [10] of how the output distribution of MC dropout samples corresponds to model uncertainty is shown in Figure 2.8. A model trained to classify an input image as a digit from "0" to "9" is presented with increasingly rotated versions of a "1". For the non-rotated version, most of the 100 forward passes result in the prediction of a "1". For the partially rotated inputs in the middle, the model equally predicts either "1", "5", or "7", demonstrating its uncertainty about the input. Any single scalar prediction would not have captured this uncertainty, with many samples even having a high softmax confidence score of 0.8 or higher.

MC dropout is applicable to any neural network containing dropout layers, while requiring no changes to the training procedure or the model architecture. The main limitation is the required sampling by running forward passes. Running 100 inferences for a single input can be prohibitively expensive for large networks. Nevertheless, MC dropout has been applied to a range of applications since its introduction. Kendall et al. [120] used MC dropout to estimate pixel-wise uncertainties in semantic image segmentation. This allows to visualize the predicted uncertainties and thus failures of a segmentation model. In the context of autonomous driving, this is a valuable property. The predicted uncertainty maps can be displayed to the human operator to alert them of which regions the model is struggling with. Fortunato et al. [121] proposed Bayesian Recurrent Neural Network (RNN)s by applying both *Bayes by Backprop* and MC dropout, demonstrating the flexibility of those concepts.

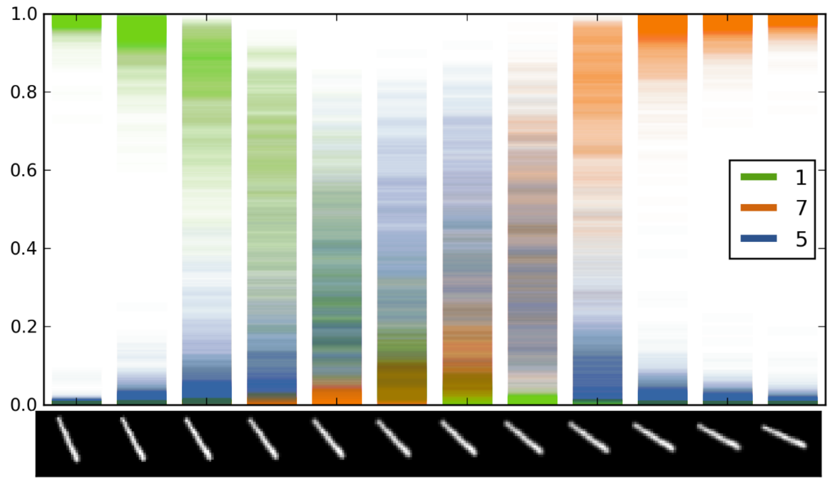


Figure 2.8 Visualization of the softmax output from 100 forward passes with active dropout from a model trained to classify digits. The bottom row shows the input of an increasingly rotated "1". The output samples are most scattered around a rotation of 45 degrees. This corresponds to the increased model uncertainty for an input that does not resemble any of the three classes (source: [10]).

An alternative to MC dropout is to use different stochastic mini batches in batch normalization layers to obtain different outputs from one model [122]. Similar to MC dropout, this approach can be applied to any trained neural network as long as batch normalization layers are present. Since the performance is comparable to MC dropout, this approach is a useful option in case a model without dropout, but with batch normalization layers is used.

Besides detecting incorrect predictions, uncertainty estimation can also be used to improve network performance. Kendall et al. [123] improved multi-task neural networks by modeling the output of each task as a Gaussian distribution and weighing the training losses based on the output variances. A single uncertainty-aware model trained to perform three tasks outperformed three individually optimized models in their experiments. In Chapter 5, several ways of using uncertainty estimation results to improve a system beyond only detecting its failures are developed.

Korattikara et al. [124] addressed the main issue of increased computational complexity caused by using MC dropout. Inspired by the concept of knowledge distillation as proposed by Hinton et al. [125], they proposed a trainer-student scheme where a student model learns to predict only the uncertainty of the teacher model. First, they applied MC dropout to a teacher network to obtain an output distribution from multiple samples. Then, they train an identical student model without dropout to predict that distribution from only the input. Obtaining the uncertainty estimation for the teacher model then requires only one additional forward pass of the student model during testing. Learning from the recorded uncertainties of a teacher model like this is similar to the concept of introspection introduced in Section 2.2.1. Their experimental results outperformed both MC dropout [10] and Bayes by Backprop [14]. While the testing complexity is reduced, the training complexity is significantly increased. A distribution of MC samples needs to be generated for each training sample, and a complete additional model needs to be trained.

Gurau et al. [126] used distillation of MC dropout samples to improve overall model performance. They trained a student network to perform the same task as the teacher model, but added the difference to 100 averaged MC dropout samples obtained from the teacher to the training loss of the student. The training loss for highly uncertain samples is thus increased. In both classification and object detection tasks, the student model with uncertainty-aware training outperformed the teacher model without additional testing complexity. The concept of training a model to predict the output of a second network with the same architecture is similar to the idea of Born-Again Networks as proposed by Furlanello et al. [127]. They also observed that student models identical to the teacher can outperform their teacher significantly.

The uncertainty obtained from MC dropout can also be used to guarantee a certain success rate by rejecting predictions with a high estimated uncertainty. Geifman et al. [11] specified a threshold and rejected all outputs of a classifier with an estimated uncertainty above this value. On the ImageNet data set [41], this approach allowed them to guarantee a 2 % top-5 error with a probability of 99.9 %, while still covering 60 % of the test set. While performance guarantees are important for autonomous driving, rejecting inputs is not always feasible in practice.

Output-based uncertainty estimation has been used in a range of tasks specifically related to autonomous driving as well. In [128], MC dropout is used to estimate the uncertainty of 3D vehicle detection based on LIDAR point clouds. The uncertainty for novel test data was higher than for data from the training distribution and the confidence score of the output was better calibrated. Meyer et al. [129] proposed probabilistic object detection by using an architecture that generates a probability distribution for each detection. Their approach offers a probabilistic uncertainty estimate for each detection at a competitive detection performance, but requires a dedicated architecture and cannot be applied to existing trained models. Finally, Michelmore et al. [16] applied MC dropout to an end-to-end autonomous driving model. They trained a neural network to predict the steering angle of a car in a simulator based on the camera images from the simulator. While MC dropout is computationally demanding, their system was capable of running in real time, generating 128 dropout samples per image at six frames per second. In their experiments, when the variance among the samples crossed an empirically determined threshold, a crash followed 73 % of the time within the next five seconds. This demonstrates the potential usefulness of uncertainty estimation for failure prediction in autonomous driving.

2.2.2.2 Non-Bayesian Methods

While Bayesian methods have shown great success in the field of uncertainty estimation, they come with several limitations. First, sample-based approaches such as MC dropout are expensive during testing, which is problematic for mobile systems such as cars. Postels et al. [130] give a general overview of deterministic instead of probabilistic uncertainty estimation methods, emphasizing the common advantage of reduced inference time. Second, many Bayesian methods are designed only for neural networks, often requiring to modify their architecture. Autonomous vehicles are complex systems that can consist of both conventional and learning-enabled components, which can be expensive to modify if they are already trained. In the following, several non-Bayesian uncertainty estimation and failure prediction methods that address one or more of those limitations are summarized.

Mohseni et al. [131] use a teacher-student scheme to predict the failures of a given model. They train the student model with the saliency maps from the teacher model as input and the errors of the teacher as the target output. While learning to predict failures from recorded failure instances is the same concept as used in introspection, their approach requires to obtain the saliency map of the output of the inspected model first. While requiring the model output is slower than predicting failures on the input directly, this method is still faster than requiring multiple samples such as MC dropout. The authors evaluated their approach on an end-to-end steering angle prediction network, correctly detecting 68 % of safety-critical deviations.

For neural networks, fail cases can be determined before deployment using the concept of neuron coverage proposed by Pei et al. [132]. They argue that the idea of code coverage when testing software can also be applied to neural networks. For neural networks, neuron coverage means generating inputs that make every neuron fire. Instead of determining if a random input leads to an uncertain model, they propose to generate inputs intended to lead to an uncertainty model to then retrain the model with those inputs. They proposed a gradient-based optimization algorithm that efficiently finds inputs that both trigger as many neurons as possible and lead to incorrect behavior of the system. The additional constraint of maximizing neuron coverage is the main difference to the established concept of generating adversarial inputs. The resulting inputs outperform adversarial training by 3 % when used to retrain the model.

While the work by Pei et al. can make a model more robust against uncertain inputs, it cannot be used at test time to assess the current model output. Corbière et al. [133] proposed a non-Bayesian way of estimating the output uncertainty by adding a confidence prediction network to the end of the inspected network itself. Their confidence network uses the high-level features extracted by the inspected model as input and receives the softmax score of the correct class as a target label. This way, the confidence network learns to predict the true class confidence even when the model assigns a higher score to a different class and makes a mistake. Their work outperformed Bayesian methods such as MC dropout in both classification and semantic segmentation. Yoo et al. [110] use a similar approach of adding another network at the end of the inspected architecture. Instead of predicting the confidence, they predict the training loss of the current sample. This loss predictor can be used both for active learning to predict the hardest samples and during testing to predict which inputs will likely be misclassified.

Another approach for evaluating the uncertainty of model output is the trust score proposed by Jiang et al. [134]. The trust score is a distance-based metric calculated in the high-level feature space of a model. It is derived from the ratio of the distance of the input to the predicted class and the distance to the nearest not-predicted class in feature space. For low-dimensional input, this nearest-neighbor-based approach outperforms other uncertainty estimation measures. For high-dimensional inputs such as images, the trust score does not even outperform the softmax score. This makes the trust score less suitable for computer vision tasks commonly performed in autonomous driving.

Most Bayesian uncertainty estimation methods discussed in the previous section relied on the variance or disagreement between samples generated from one model. The concept of disagreement between multiple outputs can be used for non-Bayesian methods as well. Fridman et al. [15] proposed to use two different models trained for the same task and monitor when their outputs differed. They apply this idea to autonomous driving by running both a Tesla autopilot system and an end-to-end steering angle prediction network in parallel. Using an empirically selected disagreement threshold between the two steering angle systems, they predicted when the Tesla autopilot would disengage five seconds in advance at an accuracy of 90 %. While those results are promising, their approach requires designing and training two structurally different models that both perform well, with no theoretical guarantees that there will be noticeable differences between the two models at test time.

The concept of deriving uncertainty from disagreement was again used by Ramanagopal et al. [135]. They analyzed the output of a detection model by looking for inconsistencies between the object detector and a separate object tracking module. They also proposed using the disagreement between the detections from the two cameras of a stereo vision system. Their approach effectively detected false detections at little extra cost, assuming that systems such as a stereo camera and object tracking are already present for increased redundancy.

A straightforward implementation of using model disagreement specifically designed as an alternative to Bayesian methods is the idea of Deep Ensembles proposed by Lakshminarayanan [12]. They train five instances of the same neural network architecture with different initializations. The resulting ensemble of models generates a distribution of outputs for one input. The variance of those outputs can be used as an uncertainty measure, similar to MC dropout. Their approach significantly outperforms MC dropout in their experiments. While MC dropout requires the inspected model to contain one or more dropout layers, the Deep Ensemble approach has no requirements on the underlying model. It also requires much less time during testing than sampling-based methods such as MC dropout, since only five models need to be evaluated instead of running up to 100 forward passes. The additional cost is mostly shifted to the training phase, where the commonly expensive training of a deep neural network needs to be performed five times as often.

Instead of training the same architecture multiple times, Geifman et al. [136] proposed storing snapshots of the model at evenly spaced time points during training to effectively create an ensemble. While the snapshots stored during the early phases of training are not capable of classifying all samples yet, they assign simple inputs a more realistic confidence score. Later snapshots tend to be overconfident on such samples. By averaging the confidence score of model snapshots both from early and late phases of training, an improved uncertainty of the output of the trained final model is obtained. While this approach outperforms Deep Ensembles and MC dropout in several experiments, it cannot be applied to a model retrospectively since access to the entire training phase is required.

2.2.3 Out-Of-Distribution Detection

The third main source of failures in autonomous driving is novel data that is too different from the training distribution, typically referred to as Out-of-Distribution (OOD) data. If a model has never seen a class before, it will fail regardless of how well it has been trained or calibrated. Next, the most relevant works focused on OOD detection in general are outlined. Then, an overview of recent work using OOD detection specifically for autonomous driving is given.

2.2.3.1 General OOD Detection

Yang et al. [9] gave a comprehensive overview of the field of OOD detection. They distinguish between several terms often used synonymously in the literature, most notably anomaly, novelty, and outlier detection. A visualization of these general types of OOD detection tasks is shown in Figure 2.9. Sensory anomalies refer to an unexpected sensor style such as a drawing instead of a picture, while semantic anomalies are unexpected semantic inputs such as a picture of a cat after training with pictures of dogs. Open set recognition is the ability to classify multiple classes such as different animals, while being able to detect if an input does not belong to any of the known classes. Outlier data refers to data that is available to the model, but that strongly differs from the rest of the observations. All of those OOD concepts theoretically pose risks to autonomous driving and can therefore be considered when discussing failure prediction.

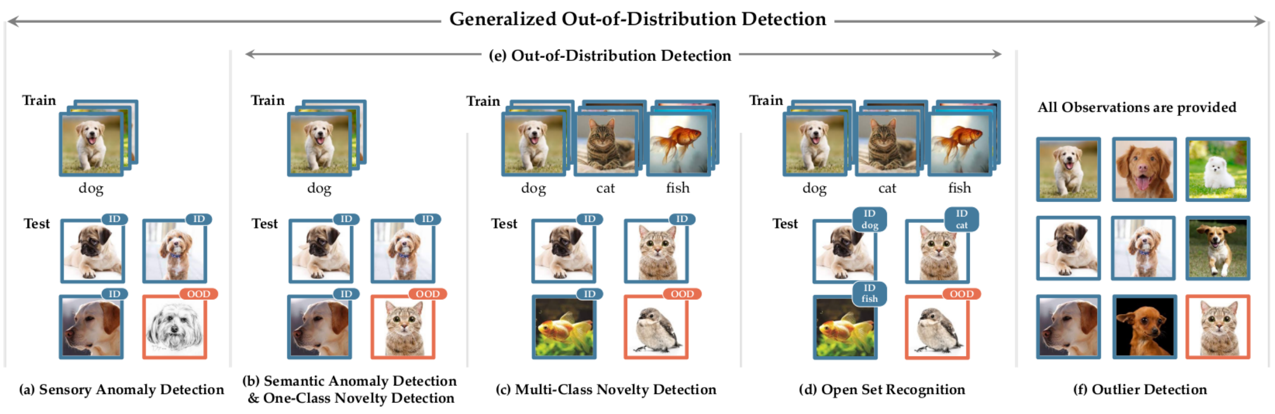

Generalized Out-of-Distribution Detection
(e) Out-of-Distribution Detection
Train
Test
dog
cat
fish
ID
OOD
ID dog
ID cat
ID fish
All Observations are provided
(a) Sensory Anomaly Detection
(b) Semantic Anomaly Detection & One-Class Novelty Detection
(c) Multi-Class Novelty Detection
(d) Open Set Recognition
(f) Outlier Detection

Figure 2.9 Overview of the key concepts of the field of OOD detection. Anomaly, novelty, outlier and OOD detection are often used synonymously in the literature (source: [9]).

Chalapathy et al. [137] gave an overview of anomaly detection specifically for the field of deep learning. A similar work was conducted by Shafaei et al. [138], who evaluated a range of state-of-the-art OOD detection methods. They found that for image-based models evaluated on common benchmark data sets, none of the methods they investigated were capable of accurately detecting anomalies. Nalisnick et al. [139] made similar observations about density-based OOD methods, concluding that they are not reliable for high-dimensional data such as images. They observed that such methods could even assign a higher likelihood to OOD data than to in-distribution data. Choi et al. [93] addressed this by using a deep ensemble [12] to estimate the uncertainty of the density-based OOD detector. Estimating the uncertainty of the OOD detector itself allowed for more accurate detection of OOD inputs.

Another promising direction for OOD detection is to use autoencoders, which can learn an embedding for a given data set in an unsupervised manner. Marchi et al. [140] used an autoencoder combined with an RNN structure to detect novel inputs in acoustic data. For new input data, they use the difference between the original input and the input reconstructed by the autoencoder. They find that the reconstruction error is generally higher for novel data, outperforming traditional statistical methods for novelty detection. An extension of this idea is to use a Variational Autoencoder (VAE) for OOD detection [141]. VAEs learn a continuous latent distribution from which the reconstruction is generated. Since the embedding of each new input is a distribution, density-based OOD detection in the latent space is possible. However, those approaches still struggle to achieve useful performances on large and complex natural image data sets.

Finally, instead of attempting OOD detection based solely on the input data, the output of the model that the data is given to can also be used. Hendrycks et al. [94] proposed a straightforward baseline for detecting OOD input by simply using the softmax score of a classification network as an OOD score. While the softmax score is not a proper probability and does not correspond to model uncertainty as discussed before [10], Hendrycks et al. empirically showed that the softmax score nevertheless allows to accurately detect if a classification network has been fed data from a different data set. They evaluated and verified this approach on speech, text, and image classification data sets. Liang et al. [142] soon improved the work of Hendrycks et al. [94]. They achieved state-of-the-art results for OOD detection by both applying calibration via temperature scaling [117] and by adding small perturbations to the input in the direction of the loss gradient. The argued that a classification model is more sensitive to data it has been trained on, and therefore such a perturbation will have a greater effect when added to in-distribution data. The resulting calibrated softmax score of the perturbed input image is significantly larger for in-distribution data than for OOD data. While recent survey papers such as [138] find that the approach by Liang et al. is still the state of the art for this task, it is only applicable to image classification networks. Additionally, the OOD data used in the works discussed in this section is typically drastically different to the training distribution, for example when training a model on images of numbers and then showing it natural images of animals. In autonomous driving, the changes in the distribution can be more subtle and the systems are more complex than classification networks. The next subsubsection discusses OOD detection approaches specifically designed for the domain of autonomous vehicles next.

2.2.3.2 OOD Detection for Autonomous Driving

Filos et al. [143] analyzed how distribution shifts affect trajectory planning for autonomous vehicles. They proposed using model ensembles to make the trajectory planner more robust by averaging multiple trajectories as well as for estimating the current model uncertainty in the same vein as Deep Ensembles [12]. They evaluate their approach both in the CARLA simulator and on a real driving data set, where their approach detects 97.5 % of trajectories that would have led to a crash. Their findings suggest that uncertainty estimation methods are a powerful tool for failure prediction both for in-distribution and OOD scenarios.

A successful application of unsupervised OOD detection using an autoencoder was proposed by Richter and Roy [144]. They trained a fully connected autoencoder with image data from the buildings in which an autonomous RC car was trained to drive. Then, they use the reconstruction error of their autoencoder to detect if the car is driving in the hallways it was trained in or if it is driving in a new building. They correctly detected every novel hallway this way and avoided collisions by switching to a slower and safer driving strategy. By recording the novel environments and retraining with the recorded data, they iteratively adapted both the autoencoder and the driving system. The car then learned to classify previously novel environments as known and could increase its speed without additional collisions. While their results are promising, the evaluation is limited to a restricted set of indoor scenes. The insides of the new buildings often looked drastically different to the building the car was trained in, for example by having a differently colored floor. Real-world road environments tend to look more similar to each other, with novelties often being temporal and local, such as when an unknown object enters the road. Bolte et al. [145] focused specifically on the detection of such local corner cases in driving environments. They leverage specific knowledge about dangerous anomalies in traffic, namely that an anomaly needs to move in an unpredictable way, needs to be a relevant class such as a pedestrian, and needs to be in a relevant location such as the side of the road. To localize an object, they divide the image into square patches and use semantic segmentation to detect relevant objects such as people. Then, they use an autoencoder to predict the current image from the sequence of previous images. If the reconstruction differs significantly from the actual current image, they assess the patch to be unpredictable and classify it as a corner case. In a qualitative evaluation, they showed that this approach is capable of detecting critical video sequences such as pedestrians moving erratically at the side of the road in an unsupervised way. While such scenarios do not necessarily pose a problem to a well-designed perception module, their approach is useful at automatically detecting relevant scenarios among unlabeled training data sets.

Another OOD detection approach capable of detecting changes in specific environmental factors is to use a β-VAE [146]. The training data is first partitioned into subsets. Each subset contains variations of only one critical factor, such as weather condition or traffic density. Then, a β-VAE is trained for each of those factors and the latent variable that is most sensitive to the respective factor is determined. During testing, the divergence of that variable's distribution to a normal distribution is used as an OOD score. This approach allows both the detection of a distribution shift as well as determining which factor was responsible for the shift. The ReSonAte (Runtime Safety Evaluation in Autonomous Systems) framework [147] extended such a detection of environment shifts into a general risk score calculation for a given scene.

Amini et al. [148] combine the task of OOD detection, uncertainty estimation and end-to-end steering angle prediction into a single network. They trained a convolutional VAE on driving images and used one of the latent variables as the steering angle output, which they supervised with the ground-truth steering angles available for each training image. The rest of the latent variables are used to reconstruct the input image. The reconstruction error serves as a novelty score. The variance of the latent variables, which take the shape of Gaussian distributions in a VAE, is used as an uncertainty estimate of the model. Their model detects 97 % of test images taken at day times not present in the training set and detects all test images taken with a malfunctioning camera. While the resulting model is highly efficient since a single forward pass yields the control output, uncertainty score and novelty score, it is limited to systems that rely on such end-to-end control. Commercial autonomous systems are typically significantly more complex and rely on multiple modules instead of a single control network. The idea of using one encoding for multiple purposes is still promising, however. Bevandic et al. [149] followed a similar idea and combined OOD detection with semantic segmentation. After a shared encoder, one decoder generates a semantic prediction while a second decoder performs a pixel-wise outlier prediction. The outlier predictor is trained with in-distribution images where outlier objects not present in the original training set have been manually added. Those outlier objects are labeled as an outlier class. While this is an effective approach for segmenting outlier objects, it requires a supervised data set of outliers. Since one of the key ideas of OOD detection is to detect objects that were not available during training, such a supervised approach is hard to generalize. The contributions of Chapter 4 use a similar approach to generate both a semantic segmentation prediction and the prediction's failure probability from a single encoding. However, they are focused on failures made only on in-distribution images.

Instead of using the encoding of a semantic segmentation model to create a pixel-wise outlier prediction, Xia et al. [150, 151] analyze the output of a semantic segmentation model to find semantic anomalies. The idea is that semantic anomalies in the input will often lead to spurious and ill-shaped semantic predictions in the output. They synthesized the input image from the semantic prediction using a GAN. Then, they trained a comparison module to detect the differences between the original input and the synthesized input that relate to anomalies. Their approach achieved state-of-the-art results for anomaly segmentation on the StreetHazards data set [94], a driving data set with 250 types of anomaly objects present only in the test set.

Finally, besides semantic anomalies, sensor anomalies can also pose a danger to autonomous systems. Jatzkowski et al. [152] address the issue of unexpected sensor input by proposing to use deep learning to detect overexposure of camera images. They train a classification CNN with a manually curated data set of both overexposed and normal driving images, achieving a test accuracy of 97 %. While training a dedicated network for each type of expected anomaly would get excessively complex, their approach is a viable option for the most critical types of anomalies that need to be detected to ensure safety.

2.3 Future Prediction

In dynamic tasks such as driving, detecting an ongoing failure might not be sufficient for avoiding dangerous situations. Predicting future problems before they actually occur would be an important improvement. While predicting future events is always challenging, it is especially complex in driving, where critical events are inherently rare [153]. We summarize several ideas for anticipating future events that are related to the future failure prediction methods proposed in Chapter 3 and Chapter 4.

First, the prediction of future driving behavior is considered. Knowing how the current driving situation will unfold can be used for anticipating problematic driving situations. Another relevant field is accident prediction. While failures of the ego vehicles are the focus of this thesis and not accidents caused by other traffic participants, the tasks are similar regarding the prediction of future events. Finally, the few works available in the literature that directly predict future failures in autonomous driving are discussed. The task of predicting failures of autonomous vehicles seconds in advance is also where we make several contributions in Chapter 3.

2.3.1 Future Driving Prediction

To predict future driving behavior from a sequence of previous observations, several deep learning concepts are available. 3D convolutions [154] and LSTMs [78] both allow to extract spatio-temporal features from input sequences which can be used for classification or regression, while GANs [155] allow to synthesize future data directly. While both motion models [156] and conventional methods such as Hidden Markov Models [157] have been used to predict future driving behavior, Kuefler et al. [158] showed that GANs are more robust for this task and generate realistic behavior even for long time horizons. They used both the state of the ego vehicle as well as information about neighboring vehicles as inputs, generating the predicted future state of the ego vehicle as their output.

For extrapolating the trajectory of the vehicle into the future, LSTMs [159] have shown state-of-the-art performance. Jain et al. [160] additionally proposed to use a video stream of the driver's face to better predict future maneuvers. They used a fusion-based LSTM to accurately predict the maneuvers of the driver 3.5 s in advance, using the outward camera images, vehicle state, and driver face images as input. While knowing what action the car will perform in the next seconds is valuable information, for failure prediction it is also important to know if that action will lead to a dangerous situation. Approaches dedicated to predicting accidents in a given traffic scenario are therefore discussed next.

2.3.2 Accident Prediction

A straightforward approach to accident prediction is to learn from video sequences where accidents have been manually labeled. With such videos, a neural network can be trained to predict the crash probability of each vehicle in an image. Tian et al. [161] used this approach and trained a variant of YOLO [51] where each detected car bounding box is assigned an accident probability. They created a data set from news reports, documentaries, and similar websites to obtain video recordings of car accidents to use as training data. While they assess the crash probability only based on the current visual context, Chan et al. [162] extend an object detector with an RNN to extract motion features, which allows to extrapolate into the future where the objects are headed. Training with dashcam videos from the internet, their approach is capable of predicting accidents two seconds before they happen at an mAP of 74 %. Suzuki et al. [163] further improved early accident prediction by using an adaptive loss that penalizes the model differently after each epoch depending on how well it can predict accidents early on. Weights that enable early accident anticipation are encouraged this way. Secondly, they manually added annotations of near-accidents to their crash video data set, obtaining more training data of how dangerous situations can develop. Their modifications achieved accident prediction over two seconds earlier than previous work, at an mAP improvement of over 6 %. Huang et al. [164] continued this approach of near-accident detection using a compact spatio-temporal network that predicts accidents at a speed of up to 30 frames per second. While all of those papers predict accidents that occur outside of the ego vehicle, they demonstrate that spatio-temporal neural networks are capable of extracting patterns in real time that allow predicting critical traffic situations seconds in advance. The contributions of this thesis to the field of early disengagement prediction in Chapter 3 are partly motivated by these results.

2.3.3 Future Failure Prediction

Finally, there are several works about future failure prediction that are explicitly applicable to predicting failures of autonomous vehicles. Hallac et al. [165] proposed an autoencoder model that learns to embed all sensor and state information of a car measured over one second into a compact vector representation. They train their autoencoder on a large driving data set and show that risky situations, such as hard braking or fast turning maneuvers, have distinct embeddings in latent feature space. By monitoring the distance of the embedding of the current scene to the clusters associated with such dangerous situations, they were capable of predicting strong braking three seconds in advance. While this approach can give a human driver time to react to an autonomous car about to suddenly brake, it is not capable of detecting dangerous situations that the car misses and would not brake for in the first place. Michelmore et al. [16] proposed to use MC dropout [10] as a way of predicting such dangerous situations in autonomous driving. They trained an end-to-end steering angle prediction network, applied MC dropout and used the variance among the resulting MC samples to assess the uncertainty of the system. When applied to a driving simulator, their approach allowed to predict crashes in the simulator three seconds in advance at a true positive rate of 73 %.

There are also several works that use the non-Bayesian uncertainty estimation concept of model disagreement outlined in Section 2.2.2.2. They assess when an autonomous car is making a mistake based on the disagreement of the car with reference data. Huang et al. [166] proposed a method for predicting when a trajectory planner is deviating from the trajectory a human driver drove on test data. They trained a deep neural network to predict a car's trajectory and additionally implemented a physics-based trajectory predictor. Then, they trained an error predictor to predict the deviation of both models from human reference data. Their approach is capable of predicting situations in which both trajectory models make large errors two seconds in advance 66 % of the time. Hecker et al. [167] follow a similar approach of predicting the disagreement between two models. They first trained an end-to-end neural network to predict speed and steering angle given image sequences. Then, they recorded disagreements between human driving data and the trained model and used this disagreement data as training data for a second neural network. The second network is designed to predict disagreements between the end-to-end model and the human reference data based solely on sensor inputs. While learning from recorded failures is similar to introspection as introduced in Section 2.2.1, Hecker et al. extended the concept by predicting future failures instead of classifying the current scenario. By training with sequences of sensor data leading up to the disagreement, they accurately predicted large steering angle disagreements two seconds in advance.

Finally, the work that is most similar to the novel disengagement prediction proposed in Chapter 3 is the work by Fridman et al. [15]. They predicted the disengagements of a Tesla autopilot based on the disagreement of the Tesla with an end-to-end steering angle prediction network designed and trained by them. By tuning a threshold on the validation set, they were able to predict disengagements five seconds in advance at an accuracy of 90 %. Their work is an important reference approach for the system-level failure prediction methods that are proposed in Chapter 3 of this thesis.

2.4 Chapter Summary

In this chapter, we introduced the theoretical background and most relevant related work on which the approaches proposed in this thesis are built. First, we outlined the general pipeline of an autonomous system and discussed perception systems as one of the main components where failures can be detected. After establishing what is considered to be a "failure" in this work and what the theoretical sources of failures are, we summarized the most relevant state-of-the-art failure prediction methods from the literature. The two main research fields are uncertainty estimation and Out-of-Distribution (OOD) detection. Finally, approaches for predicting future critical events were discussed, since the goal of this thesis is to develop a failure prediction framework capable of predicting problematic situations as early in advance as possible. For the proposed early failure prediction methods, we mostly rely on concepts from the field of uncertainty estimation and the prediction of future events. Especially the concept of introspection is used prominently in both the system- and component-level failure prediction methods. In the next chapter, we introduce a system-level failure prediction approach as the first major contribution of this thesis.

3 System-Level Failure Prediction

In this chapter, we present the proposed framework for system-level failure prediction. The framework treats the car as a black box and applies the concept of introspection to predict disengagements of the autonomous system. Four implementations of this general idea are developed: a state-based, an image-based, a trajectory-based, and a fusion-based approach. Before discussing these specific implementations, we introduce the general idea of introspective disengagement prediction in more detail first.

Some of the concepts and contributions of this chapter have been published in [23, 25, 26].

3.1 Introspective Black Box Disengagement Prediction

This section defines system-level failures more precisely in the context of autonomous driving. Then, we introduce the concept of introspection and the introspective failure prediction framework on which the subsequent failure prediction methods are built is presented.

3.1.1 Disengagements as System-Level Failures

A system-level failure means that a system is not performing its task correctly. In autonomous driving, the result of a system-level failure is a state in which the autonomous vehicle cannot drive safely anymore. The most extreme version of a system-level failure is therefore a crash caused by the vehicle. However, crashes caused by autonomous vehicles as they are currently being tested on public roads are still very rare events [4]. This can be largely explained by the fact that human safety drivers are typically present in the car during test drives. During such supervised test drives of autonomous systems, it can therefore be assumed that unsafe situations are detected before they have significant consequences. This process of returning control to the human driver to avoid a dangerous situation is referred to as a disengagement of the autonomous system. Not every situation in which a disengagement happens would have necessarily led to a crash if no intervention had occurred. In the absence of actual crashes, situations in which the car disengages are as close to safety-critical failures of autonomous system as possible. In this dissertation, disengagements are therefore treated as system-level failures. There are two main ways the autonomous functions of a car can disengage. Automatic disengagements are triggered by safety systems implemented in the car, while manual disengagements are triggered by the human driver.

3.1.1.1 Automatic Disengagements

Assisted or automated driving functions typically already have an automatic way of switching off in case any safety requirement is violated. Such requirements can include a minimum distance to other vehicles, a minimum confidence in detected objects or a minimum quality of sensory input. Commercially available Level 2 functions such as lane-keeping assistants or cruise control will alert the human driver in case their functionality cannot be ensured anymore. Since the human driver is always expected to be ready to drive again, disengagements are usually not critical events. In Level 3 and beyond, the human is not expected to be fully aware of the road at all times. It can take up to half a minute to regain control [7]. Predicting disengagements as many seconds in advance as possible can therefore increase safety by allowing the human driver some time to become aware of the situation before having to take control.

3.1.1.2 Manual Disengagements

A more safety-critical system-level failure is a scenario where the car does not react automatically to an unsafe situation and the human safety driver has to intervene. For the purpose of designing a failure prediction framework, such failures are highly relevant situations to predict. Manual disengagements describe scenes in which the car would have continued its functions despite a human expert judging the behavior to not be safe anymore. Such scenes are therefore failures that the system itself would have missed. During deployment of automated driving functions where no test driver is present anymore, such situations have to be detected to ensure safety.

During the test drives of an automated driving function, the availability of the safety driver is a valuable source for obtaining failure data. A human intervention can be considered to be a manual labeling of the current scene as a failure. Human labeling of data is usually expensive and time-consuming. Since human safety drivers need to be present during test drives anyway, the effective labeling of such scenes as failures comes at no additional cost.

In summary, system-level failures as considered in this dissertation refer to either automatic or manual disengagements. Recordings of such failures can be obtained as a side product of inherently necessary test drives. After months of test drives, considerable amounts of recorded system-level failures can become available this way. Next, the question of how to predict those failures is addressed. The concept of introspection allows creating such a failure prediction framework by learning from recorded failures. It is discussed in the context of autonomous driving next.

3.1.2 Introspective Failure Prediction

Introspection describes the concept of learning from previous failures of a given system. Instead of trying to correctly react to every scenario, the idea is to accept that a system will encounter situations it is incapable of handling. An introspective system should "know when it does not know" [111]. By recording as many failures of a system as possible before deployment, it is possible to learn typical failure patterns of a system. While the system will still not necessarily know what to do if such a pattern is encountered, it can at least be aware that it has entered a problematic scenario and enter a safe fallback mode to minimize danger.

For this purpose, an introspective failure prediction model is required that is trained to assess if the current situation is a failure. Training a separate model for this task has the benefit of being less susceptible to overconfidence. By learning from recorded failures, the introspective model knows that the inspected system made an incorrect choice regardless of how confident the inspected system was. For introspection, it is only necessary to know if the inspected system has failed. This makes introspection complementary to model-based uncertainty estimation method. Any critical uncertainties that model-based methods did not manage to detect will result in recorded failures. Introspection can then be used to learn to detect those situations that model-based uncertainty estimators overlooked. No knowledge about the interior workings of the system is required. Introspection allows to treat the inspected system as a black box and can operate using only observations of the system. From the recorded failures of the inspected system, any observed data can be used as input data for the introspective model. By recording both successful behavior and failures of the inspected system, the introspective model can learn a mapping of the observed data to the one of the two system states of *Success* and *Failure*. For this mapping, deep neural networks can be used. Since only the two states of *Success* and *Failure* are needed, the most straightforward approach to implementing an introspective failure prediction model is as a binary classification network.

A summary of the resulting framework that can be applied to autonomous driving is shown in Figure 3.1. During training, driving data is recorded. This can include the state of the car, sensory input such as camera images, or the output such as the planned trajectory. Since only input and output of the car are used, the car itself is treated as a black box. No assumptions are made about how the autonomous driving functions are implemented. During test drives, both successful driving and driving ending in a disengagement are recorded. After a sufficient amount of system-level failures are recorded, a neural network classifier can be trained to classify the current driving data as either *Failure* or *Success*. The softmax score corresponding to the class *Failure* can be used as the current failure probability. This way, the introspective model learns to detect patterns in the recorded driving data that indicate a failure. During inference, the classifier is fed the current driving data and classifies the current scenario. By detecting patterns that have led to failures in the past, the introspective model can ideally detect new failures as well. For the model to be able to generalize from recorded failures to new failures, a sufficiently large amount of failures is needed. Since car manufacturers are already completing millions of miles during autonomous test drives [4], such data can be available at little extra cost.

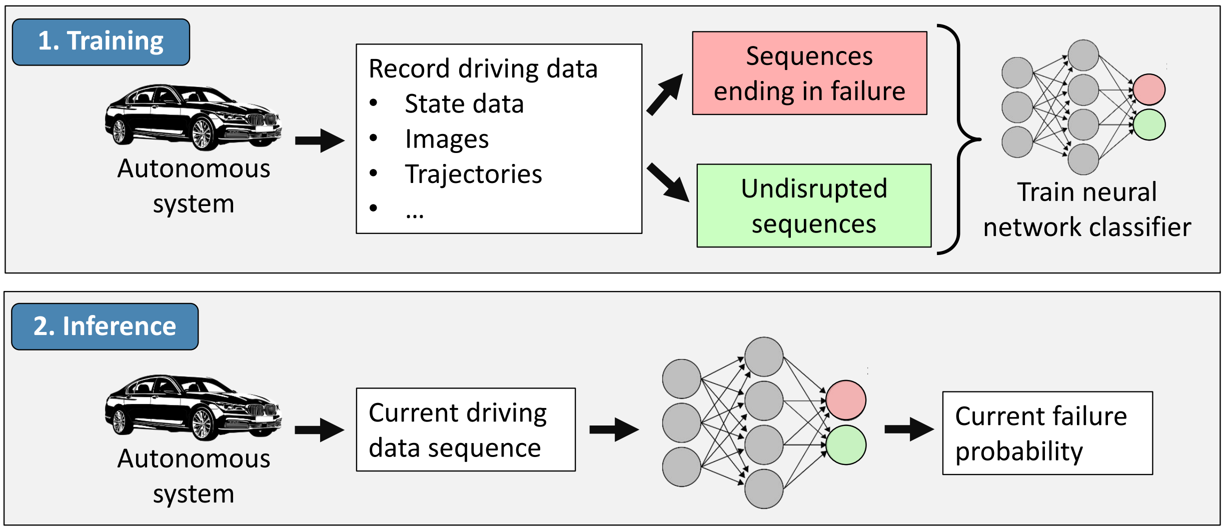


Figure 3.1 An overview of the general concept of black box introspective disengagement prediction. During training, a classifier is trained with sequences of recorded driving data labeled as either *Success* or *Failure*. During testing, the failure probability is constantly updated with the latest driving data as input.

The proposed introspective failure prediction for autonomous vehicles is based on data that is effectively labeled for free during test drives, makes no assumptions on the underlying system and can be used in addition to existing safety measures while not being susceptible to overconfidence. If the introspective model is trained only with input data of the inspected model, a failure can be predicted even before the potentially complex and time-consuming system processes the input. Motivated by those appealing theoretical properties of introspection, several specific implementations with a focus on early temporal failure prediction are presented next.

3.2 State-Based Black Box Failure Prediction

In this section, we present an introspective black box failure prediction approach for autonomous driving based on the state data of the car. The state of the car, i.e., its speed, angle, acceleration, and angular velocity, is a readily available source of information about the system. Regardless of the employed sensors or software, the physical state captures the actual behavior of the system. The low dimensionality of state data is another useful property. While new complex traffic scenes can look significantly different to each other in camera images, for instance, the state of the car is more limited in its variations. Overly complex situations are likely to lead to more abrupt steering or braking maneuvers, regardless of the specific problem in the environment. Such variations of the car state can capture complex changes in the environment in a very compact form. Based on this argument, we design a model that takes sequences of state data as input and outputs a failure probability. First, the generation of a suitable data set as the basis of subsequent machine learning methods is discussed.

3.2.1 Data Set Generation

For introspection, the availability of failure data is essential. Since disengagements correspond to an autonomous driving system not being able to fulfill its intended function, car or software developers working on autonomous driving usually do not make such data public. This dissertation is part of a project of Technical University of Munich together with the BMW Group. Recordings from test drives by the BMW Group were therefore made available for the purpose of this research. They contain the collected data from six months of test drives on urban and highway roads nearby Munich, Germany. Those roads include intersections, construction sites, and varying environments such as buildings and vegetation. The drives include all daytime settings as well as diverse weather conditions including snow, rain, fog and sun. The recorded drives are therefore a realistic and challenging representation of urban driving.

All test drives were performed by BMW development vehicles driving in their current prototype autonomous mode. A human safety driver was at the wheel at all times to ensure continuous safety. If the human driver considered a situation to be too dangerous or complex, a manual disengagement was triggered. In parallel, the car's safety system triggered an automatic disengagement if any safety requirement was violated, returning control to the human driver. Every disengagement was recorded as well as all state and sensor data of the car. System-level failures are inherently rare events. Driving is typically uneventful, with complex or overly challenging traffic scenarios being the exception. Even in such extensive driving records, the number of disengagements is therefore limited. From terabytes of recordings, a total of over 2500 unique disengagements were extracted. This number demonstrates the need for a fleet of development vehicles performing months of test drives to obtain a sufficient number of failure cases. Effectively, the resulting failure data set is created from month-long human supervision and partially manual labeling of critical situations. Considering that any autonomous driving system requires intensive test drives before deployment, such system-specific failure data can be obtained without significant additional effort.

Having obtained a large number of drives ending in a disengagement, the next step is to create an actual data set out of the raw recordings. Since the goal is to predict disengagements as early in advance as possible to give the human driver time to prepare for the takeover, temporal sequences of data are needed. We therefore use the 10 seconds of driving before each disengagement as the failure data to learn from. To obtain a balanced data set, we also randomly sample an equal number of 10 second sequences from successful driving recordings. For early failure prediction, it is also possible to consider the time frame earlier than 10 seconds before a disengagement. However, a vehicle driving at 50 km/h already moves almost 140 m in 10 seconds, for example. In such a distance, the environment can change drastically and the obstacles that caused the failure could still be far away. We therefore limit the time window to the 10 seconds before each failure during training, which is still early enough to give the human operator a warning multiple seconds in advance.

2549 disengagement sequences of state data are extracted from the raw BMW driving records and the same number of undisrupted driving sequences is sampled. The resulting data set is summarized in Table 4.2.

Number of disengagements	2549
Number of randomly sampled success sequences	2549
Sequence length	10 seconds
Sensor frequency	10 Hz
Selected state variables	Speed v Frontal acceleration a_x Lateral acceleration a_y Steering angle θ Angular speed ω

Table 3.1 Summary of the failure data set extracted from the raw driving recordings provided by the research vehicles of the BMW Group.

The entire data set consists of over 14 hours of driving data. From each 10 second sequence, we record a series of state data at a frequency of 10 Hz. This frequency is also commonly chosen in related work [167] and allows the failure prediction model a reasonable 100 ms to generate its prediction. The recorded state data consists of the speed v, the steering angle θ, the frontal acceleration a_x, the lateral acceleration a_y and the angular speed ω. At a frequency of 10 Hz, each 10 second sequence consists of a total of 100 recorded car states. Each sequence therefore consists of 100 state feature vectors F_i, given as

$$F_i = [v_i \ \theta_i \ a_{x,i} \ a_{y,i} \ \omega_i]^\top, \qquad i \in \{1, 100\}. \tag{3.1}$$

While sequential data is necessary to allow the introspective failure prediction model to learn temporal patterns, the 10 s sequences need to be further processed before being used as training data. Using the entire sequences as training input would make the model learn patterns that span across 10 seconds. In practice, changes in the environment that lead to a system-level failure often take place much faster than this. Since the model should be able to detect fast changes that indicate an impending failure as well, the sequences are split up into shorter samples.

Each sequence consisting of 100 feature vectors is split into shorter samples of length $L < 100$ that overlap by all vectors except the last one in each sample. Figure 3.2 visualizes the process of obtaining $100 - L + 1$ samples of length L out of one sequence consisting of 100 feature vectors F_i.

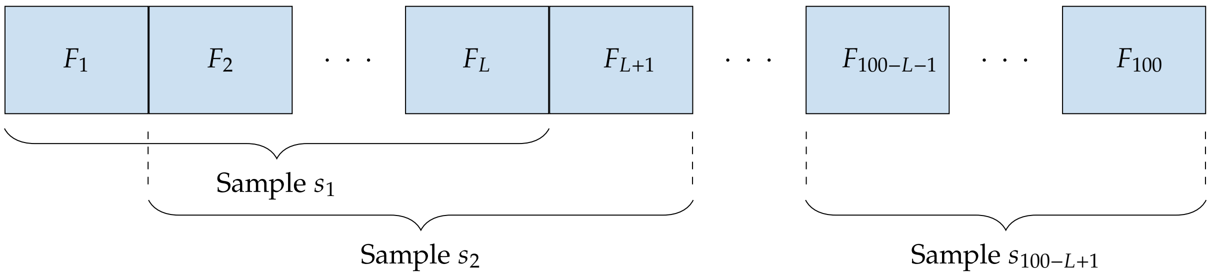


Figure 3.2 Visualization of how shorter samples s are generated from a sequence of 100 feature vectors F_i that have been extracted from a driving sequence with a duration of 10 seconds.

While the resulting samples s_i, $i \in \{1, 100 - L + 1\}$, are largely correlated to each other, the overlap ensures that every pattern that leads up to the disengagement at the end of a failure sequence is captured among the samples. The sample length L is a parameter that can be tuned based on the temporal patterns that are expected to be relevant. Values ranging from $L = 10$ to $L = 50$ will be evaluated in our experiments, corresponding to a temporal sample length of 1 s and 5 s, respectively. Once a length is selected, the corresponding samples are created from the sequences. Each sample is labeled as either *Failure* or *Success*, depending on which type of sequence it was created from. The set of all samples together with the binary labels constitute the overall data set that can then be used to train a machine learning classification model. Given a total of N_F recorded disengagements, the N_F failure sequences are divided into $100 - L + 1$ failure samples $s_{n_F,i}$ each. After randomly sampling N_F success sequences from the recordings without disengagements, $100 - L + 1$ success samples $s_{n_S,j}$ are generated from each of the N_F success sequences. The overall state-based data set D_{state} can then be described as

$$D_{\text{state}} = \{s_{n_F,i}, s_{n_S,j}\}, \ i, j \in \{1, 100 - L + 1\}, \ n_{F,S} \in \{1, N_F\}. \tag{3.2}$$

Next, the state-based data set D_{state} can be used to train and test an introspective failure prediction model. For this, the data set is split into 80 % for training, 10 % for validation and 10 % for testing. With the data fully processed, a classifier to predict the label of each sample can be designed.

3.2.2 LSTM-Based Classification Approach

The introspective failure prediction model needs to be able to classify a sample consisting of L sequential feature vectors as one of two states, *Failure* or *Success*. To enable early temporal prediction, the model should learn to detect temporal patterns in the samples. Since feature vectors are sampled at 10 Hz, the resulting 10 predictions per second can be prone to occasional outliers and need to be post-processed accordingly. Both the design of the classification model and the post-processing are explained in detail next.

3.2.2.1 Model Design

Introspective failure prediction can be framed as a binary classification task, where a sequence of state vectors is mapped to either *Failure* or *Success*. To perform this task, a classifier C_{state} is required. The score of the *Failure* class can be considered as the predicted failure probability. Given an input sample s_t, the model C_{state} therefore assigns a failure probability $P_{\text{state},t}$ to that sample as follows:

$$P_{\text{state},t} = C_{\text{state}}(s_t),\ P_{\text{state},t} \in [0,1],\ s_t \in \mathbb{R}^{5\times L} \tag{3.3}$$

To achieve early failure prediction, C_{state} needs to detect temporal pattern in the input samples. Recurrent Neural Networks (RNNs) are neural networks designed to extract temporal patterns from sequential data. RNNs with Long Short-Term Memory (LSTM) units are a state-of-the-art approach for learning longer temporal patterns from such data and are one of the most popular choices in comparable works [167, 15]. For the task of classifying temporal sequences as *Failure* or *Success*, we therefore also base the classification architecture of C_{state} on LSTMs. The architecture that performed best in preliminary experiments on the validation set is shown in Figure 3.3.

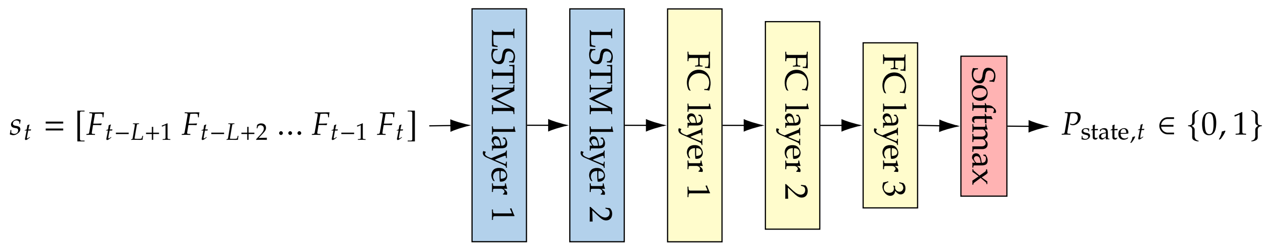

$s_t = [F_{t-L+1}\ F_{t-L+2} \dots F_{t-1}\ F_t]$
LSTM layer 1
LSTM layer 2
FC layer 1
FC layer 2
FC layer 3
Softmax
$P_{\text{state},t} \in \{0,1\}$

Figure 3.3 Visualization of the architecture of the state-based classifier C_{state} that assigns an input sample s_t consisting of L sequential feature vectors F_i a failure probability $P_{\text{state},t}$. Two bi-directional LSTM layers are followed by three FC layers. The final softmax layer generates the score for the class *Failure*, which is used as the failure probability $P_{\text{state},t}$ (Adopted from [25] © 2020 IEEE).

The core of the architecture are the first two bi-directional LSTM layers. Both have 40 nodes. After the LSTM layers have extracted temporal features from the input sequence, several FC layers perform the classification of those features. The three FC layers have 40, 20, and 2 neurons each. The last layer is responsible for the actual classification as one of the two classes. The number of nodes per layer as well as the number of layers were determined empirically. Adding more layers or increasing the size of the individual layers did not improve performance. Reducing the number of nodes in the second FC layer by the factor of two reduces computational complexity without affecting accuracy. After the FC layers, a softmax layer turns the output of the third FC layer into a score between 0 and 1 for both classes. The score of the *Failure* class is used as the failure probability $P_{\text{state},t}$ of the current input sample s_t, which consists of the current feature vector F_t as well as the $L-1$ previous feature vectors. Before training, the value of the speed v is normalized to the range $[0,1]$ since no reverse driving is recorded. The frontal and lateral acceleration a_x and a_y, the angle θ, and the angular velocity ω are normalized to take values in the range of $[-1,1]$. At an input rate of 10 Hz, the predicted failure probability is updated 10 times per second.

The hyperparameters of the training process were selected empirically. The Adam optimizer [168] is used to minimize the cross entropy loss with a learning rate of 0.001. The network is trained for 10 epochs, after which the model had converged and the validation loss did not further decrease. The architecture and the training procedure are purposefully chosen to be straightforward and low in complexity. This allows to evaluate the proposed framework as independently of the specific architecture as possible. More elaborate and complex models could still be designed to potentially optimize performance. Here, the focus is on implementing a successful proof of concept for using deep learning to predict failures early in advance.

3.2.2.2 Output Filtering

The input of the classifier C_{state} consists of physical state measurements which inevitably contain noise, for example from uneven roads briefly affecting the steering angle. Since the model generates 10 new failure predictions per second, it is susceptible to react to short-lived changes in the input caused by such noise. The raw failure predictions generated by C_{state} can therefore also be noisy and contain outliers. To remove those outliers, a post-processing step is applied. The output sequence is filtered with a low pass. A moving average filter is selected due to its low computational complexity and its property of smoothing noisy sequences [169]. This way, the output of the classifier only changes if the predictions consistently change over multiple inputs in a row. The final failure probability $P_{\text{state},t}$ filtered over a horizon H of previous predictions is then obtained as

$$P_{\text{state},t} = \frac{1}{\min(t,H)} \sum_{k=1}^{\min(t,H)} P_{\text{state},t+1-k}. \tag{3.4}$$

The horizon H is set equal to the sample length L. The model first uses the previous L state vectors to make the current prediction and then computes the average over the previous L predictions to obtain the final failure probability. At the very beginning of a sequence, the time t of the sequence is still lower than the horizon H, in which case the average is computed with fewer than H previous predictions until $t = H$ is reached. For the prediction to change from *Success* to *Failure*, the failure probability needs to be larger than 0.5 on average for L inputs. The lower L is selected, the quicker the system is capable of changing its prediction, at the cost of being more susceptible to outliers.

3.2.3 Results

In this section, results of the evaluation of the state-based failure prediction approach are presented. To the best of our knowledge, no black box failure prediction approach based on low-dimensional data exists in the literature. Since no comparable reference approaches are available, the proposed model is analyzed on its own in multiple ways. First, the state data is visualized to obtain a more intuitive understanding of what the proposed model can realistically learn. Then, the failure prediction performance is evaluated and the impact of the sample length L is analyzed. An ablation study is performed to show the importance of each state variable selected for the input. Finally, the failure cases of the proposed model are discussed to gain more insights into the limitations of this concept.

3.2.3.1 Data Visualization

The normalized state data used to train the introspective failure prediction model is visualized in Figure 3.4. For each time step t of both success and failure sequences, the average value of each state variable is calculated and plotted. Regarding the speed v, it can be seen that the car drives significantly faster on average during successful sequences. Averaged over all recordings, there are few variations in its speed. In contrast, the speed in failure sequences decreases noticeably during the last three seconds before a disengagement. Similar observations can be made when plotting the frontal acceleration a_x. During successful driving, braking and accelerating are almost even. Roughly six seconds before a disengagement, the car starts to increasingly decelerate, reaching the strongest braking at the time of the disengagement itself. The lateral acceleration a_y shows that, on average, the car moves more to the left during failure sequences and more to the right during success sequences. This is in line with the fact that left turns at intersections on German roads are significantly more complex than right turns. The angle θ offers similar insights, with failure sequences containing the strongest left turns. Finally, the angular speed ω shows much noisier values during disengagement sequences, supporting the intuition that the car starts driving more erratically when encountering a challenging situation.

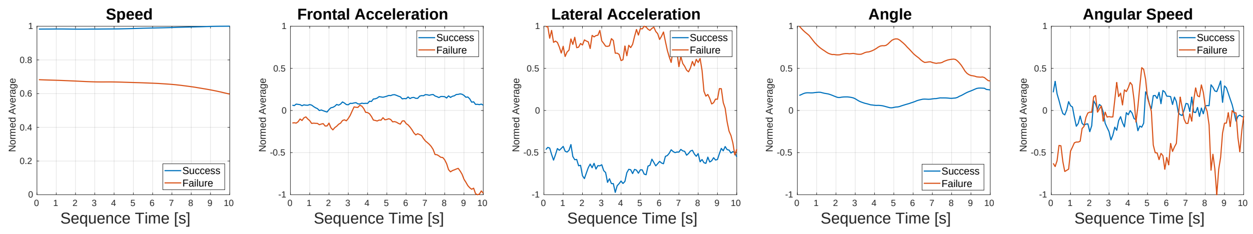


Figure 3.4 Visualization of each state variable over time. The average value of each state when approaching a disengagement develops significantly differently than during undisrupted driving.

The visualization of the state data shows that there are clear differences between disengagement and success sequences. This indicates that the application of a discriminative classifier is a promising direction. Next, the failure prediction performance of the classifier trained with the state data visualized in Figure 3.4 is presented.

3.2.3.2 Failure Prediction Performance

The failure prediction performance of the proposed state-based classifier is evaluated on 255 test sequences ending in a disengagement and 255 test sequences of successful driving. First, the impact of the sample length L and of the output filtering is investigated. Five different sample lengths of 1 s to 5 s are tested, corresponding to 10 to 50 feature vectors per sample. After training the classifier C_{state} once for each sample length, the average error over time both for the unfiltered predictions and the predictions after the moving average filter is computed. In Figure 3.5, the error plus the standard deviation over time for each of the five sample lengths L is shown for both failure and success sequences.

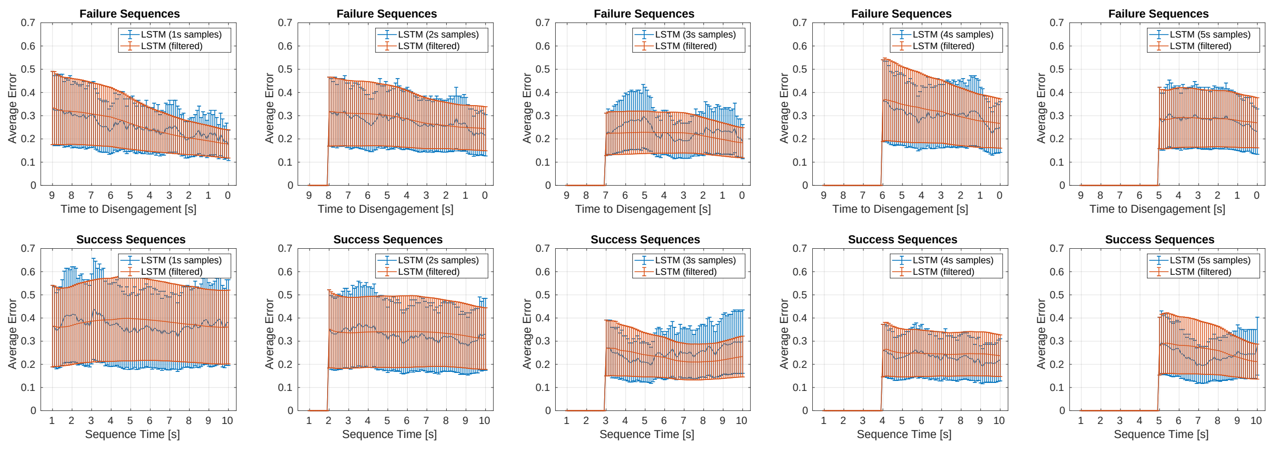


Figure 3.5 Comparison of the failure prediction performance of the state-based LSTM classifier over time for both failure and success sequences when trained with samples of length 1 s to 5 s. Adding the moving average output filtering improves the accuracy and reduces the variance.

The error from using the unfiltered output of the state-based model is shown in blue, the filtered version is shown in orange. The moving average filter noticeably smoothes the error. For each sequence length L, the error decreases when a disengagement is approached.

Next, the performance is quantified by calculating the overall accuracy averaged over all time steps and over both failure and success sequences. As a second metric, a Receiver Operating Characteristic (ROC) curve analysis is performed, using the predicted failure probability $P_{\text{state},t}$ as the score for each sample. For a scalar comparison, the Area Under Curve (AUC) is computed, once for the unfiltered and once for the filtered predictions. A comparison of the five sample lengths L and the effect of the output filtering is shown in Table 3.2.

Sample Length	**Accuracy**	**Accuracy (filtered)**	**AUC**	**AUC (filtered)**
1 s	68.9 % (±0.28)	69.5 % (±0.12)	74.5 %	75.7 %
2 s	70.0 % (±0.23)	70.1 % (±0.10)	75.7 %	76.6 %
3 s	75.9 % (±0.23)	78.2 % **(±0.08)**	82.6 %	84.6 %
4 s	73.4 % (±0.21)	73.3 % (±0.09)	78.0 %	79.2 %
5 s	73.9 % (±0.19)	74.2 % (±0.06)	79.7 %	80.1 %

Table 3.2 Comparison of the accuracy and Area Under Curve (AUC) for samples of length 1 s to 5 s, with and without output filtering. The best performance is achieved by using 3 s samples together with output filtering.

Output filtering increases the accuracy and AUC for all sample lengths and significantly lowers the standard deviation. The average accuracy as well as the AUC is highest when samples of length $L = 30$, meaning a duration of 3 s, are used. An accuracy of 78.2 % at an AUC of 82.6 % can be achieved this way. In Figure 3.6, the entire ROC curves for all five sample lengths are shown. The sample length of 3 s performs the best across the entire curve. For all further experiments, this sample length is used.

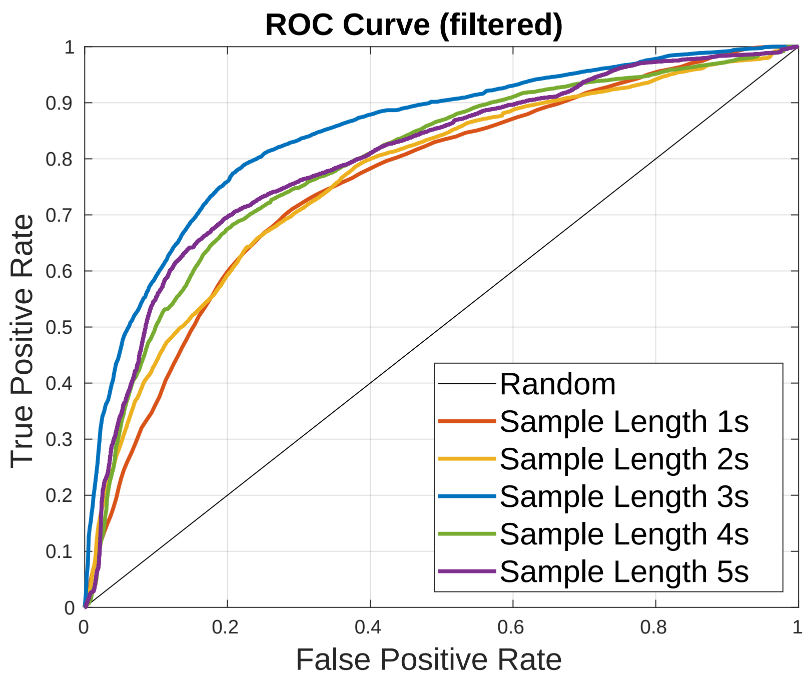


Figure 3.6 Receiver Operating Characteristic (ROC) curves of the state-based classifier trained with samples of length 1 s to 5 s.

The ROC curve in Figure 3.6 allows to adjust the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) by selecting different classification thresholds. In a safety-critical task such as driving, false negatives are more dangerous than false positives. A higher TPR is desirable. For using the state-based LSTM to make a discrete prediction, a threshold is required to decide at what predicted failure probability $P_{\text{state},t}$ the predicted class is set to *Failure*. To make the model more sensitive to failures, a slightly reduced threshold of 0.45 is selected. At the overall accuracy of 78.2 %, this leads to a TPR of 80.0 %.

Finally, the failure prediction performance over time is evaluated. The average accuracy for each time step of both failure and success test sequences is plotted in Figure 3.7. Since a sample length of 3 s is used, the first prediction is made after the first three seconds of the sequence have been observed. The earliest failure prediction is thus made seven seconds in advance of a disengagement. Figure 3.7 shows that the performance increases during the last three seconds of a failure sequence. This indicates that the car state starts changing more significantly in that time, making it easier for the model to discriminate between successful driving and disengagement sequences.

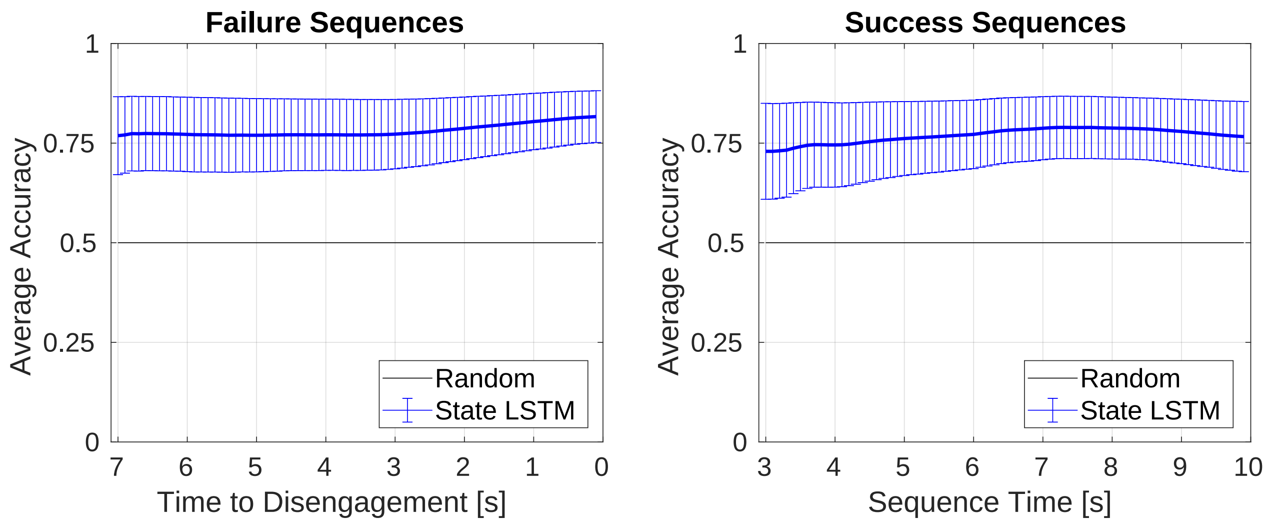


Figure 3.7 Average failure prediction performance by the state-based classifier over time for both failure sequences and success sequences. The average accuracy noticeably increases during the last three seconds before a disengagement.

3.2.3.3 Ablation Study

Using sequences of the selected state vectors allows predicting failures seven seconds in advance at an accuracy of over 75 %. To verify that each state variable used in the training is necessary, an ablation study is performed next. The five state variables of speed v, angle θ, frontal acceleration a_x, lateral acceleration a_y and angular speed ω are removed from the feature vectors one at a time. Then, the LSTM-based classifier is retrained using the same approach as before, but with each 3 s sample consisting of sequences of only four state variables. For each state, the average accuracy loss compared to using all five states is given in Table 3.3.

The accuracy of the model decreases by at least 2.7 % when removing one state, demonstrating that all five states are needed to achieve the best results. Removing the angular speed ω leads to the largest accuracy drop of 4.3 %. While the visualization of ω in Figure 3.4 appeared noisier than the other state variables, this results suggests that the fast variations in the angular velocity are an important part of the patterns that the LSTM-based classifier learns to detect.

Removed State	Accuracy Loss
Speed v	4.1 %
Frontal acceleration a_x	2.7 %
Lateral acceleration a_y	3.7 %
Angle θ	4.1 %
Angular speed ω	4.3 %

Table 3.3 Ablation study for the five state variables used as input for the state-based classifier. One state is removed at a time, the model is retrained and the accuracy loss caused by removing each state is calculated. All five selected state variables lead to a significant drop in performance when removed.

3.2.3.4 Failure Cases

Finally, the limitations of the proposed model are investigated by analyzing the failure cases of the system. While a TPR of 80 % is a promising first result, the LSTM-based classifier still misses 20 % of all disengagements. To obtain some intuition about these failure cases, the speed of the car for both correct and incorrect predictions as well as the steering angle is plotted for both success and failure sequences in Figure 3.8.

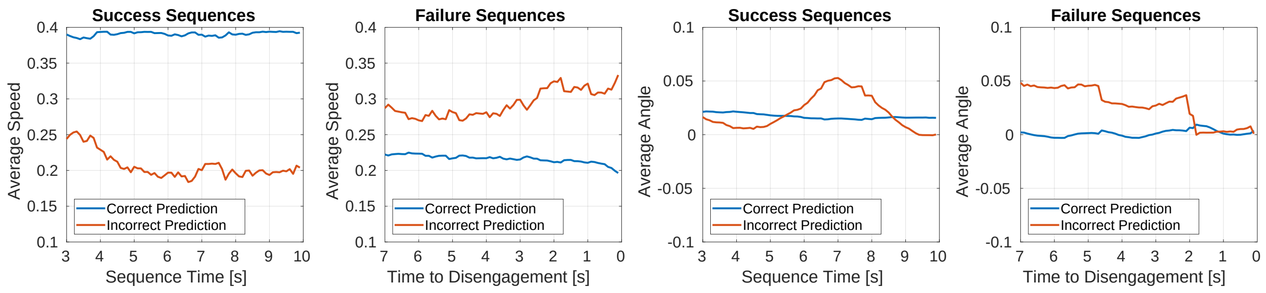


Figure 3.8 Visualization of the failure cases of the state-based LSTM classifier. The average angle and speed of incorrect predictions is compared to the respective angle and speed of correct predictions for both success and failure sequences.

In success sequences, the average speed for correct predictions is almost twice as large as for incorrect predictions. Conversely, the speed of incorrect predictions in failure sequences is over 50 % larger than for correct predictions. Considering that the average speed during failure sequences is lower than during success sequences as shown in Figure 3.4, these observations suggest that the model learned to rely on the fact that lower speeds indicate failures. Failure samples with an unusually high speed as well as success samples with an unusually low speed are therefore the most common error sources of the state-based LSTM classifier. While the speed of the car is an important input variable as shown in Table 3.3, these findings indicate that additional information about the scene is required.

When plotting the average steering angle of both correct and incorrect predictions on the right of Figure 3.8, it can be seen that the average angle of correct predictions is almost zero. This holds both for success and for failure sequences. Interestingly, the car mostly performs left turns during any kind of incorrect prediction. This indicates that left turns are among the most challenging events to classify correctly by the failure prediction model. During left turns, the state of the car might not be informative enough anymore to detect a failure. Similar to sequences with unusual speeds, a different source of information is required to correctly assess those situations. In the next section, images are therefore explored as another, much denser input modality to capture the environment.

3.3 Image-Based Introspective Failure Prediction

Images are a rich source of information about the environment of the vehicle. Modern self-driving systems are equipped with a multitude of cameras to monitor the surroundings. Images are therefore readily available as another potential input for a failure prediction model. Considering the limitations of the state-based approach outlined in the previous section, an image-based introspective failure prediction concept is designed next. Same as with the state-based approach, the first step is to generate a suitable data set and the second step is to design the classification model.

3.3.1 Data Set Generation

Since the same introspective concept as before is used, the same approach of extracting disengagement sequences from months of recordings by the BMW Group can be employed. Now, the RGB images recorded by the central front camera of the vehicles are used as data. In contrast to state data, video sequences of RGB images take up considerable storage and are much slower to process. Since a failure prediction rate of 10 Hz is the objective, a method to efficiently store the temporal information of the video sequences is required.

In the literature, a range of approaches for learning from video sequences in the context of autonomous driving is available [167, 111, 16, 15]. Fridman et al. [15] trained an end-to-end steering angle prediction network with video sequences. Instead of training directly on the raw images, the authors efficiently captured temporal patterns in images by creating difference images of gray-scale versions of the input images. To obtain the temporal development of the differences between two subsequent images, they create three difference images at three different time points and merge these three difference images into one three-channel dynamic difference image. This way, the information about the changes between up to six images is stored in one three-channel image. In this work, a similar approach is used.

Same as before, sequences of 10 seconds are extracted from the raw driving records. Half the sequences end in disengagements, the other half is sampled from successful driving to obtain a balanced data set. The 10 second sequences are again split up into shorter samples. With the state-based approach, a sample length of 3 s led to the best results. The same sample length is used for the image-based approach. At a sensor frequency of 10 Hz, 30 images would be available per failure prediction. Instead of using the images directly, they are processed as follows.

For each new image I_t recorded by the camera, three past images from 1 s, 2 s, and 3 s before are considered as well. Thus, only the images I_t, $I_{t-1\,\mathrm{s}}$, $I_{t-2\,\mathrm{s}}$, and $I_{t-3\,\mathrm{s}}$ are used from each sample. While they are recorded in color, they are converted to grayscale to further reduce dimensionality of the input data. Then, the temporal difference image $I_{d,t}$ of each consecutive pair of images is calculated as

$$I_{d,t} = I_t - I_{t-1\,\mathrm{s}}. \tag{3.5}$$

By calculating three difference images $I_{d,t}$ at time intervals of 1 s, the spatio-temporal changes across the three seconds of each sample are obtained. Combining the grayscale difference images results in the three-channel dynamic difference image $I_{\mathrm{dd},t}$ at time t, defined as

$$I_{\mathrm{dd},t} = \{I_t - I_{t-1\,\mathrm{s}}, I_{t-1\,\mathrm{s}} - I_{t-2\,\mathrm{s}}, I_{t-2\,\mathrm{s}} - I_{t-3\,\mathrm{s}}\}. \tag{3.6}$$

Besides creating a compact representation of three seconds of images, this processing step also effectively normalizes the images. In driving, quickly moving objects such as pedestrians or cars on the road are especially relevant. Difference images can therefore be more useful than regular RGB images since such objects are highlighted. Less relevant areas such as the unchanging road or horizon are mostly set to zero. The intensity changes between the three difference images capture changing speeds of the objects. A visualization of how a dynamic difference image is obtained from a sequence of four images, captured across three seconds of driving, is shown in Figure 3.9.

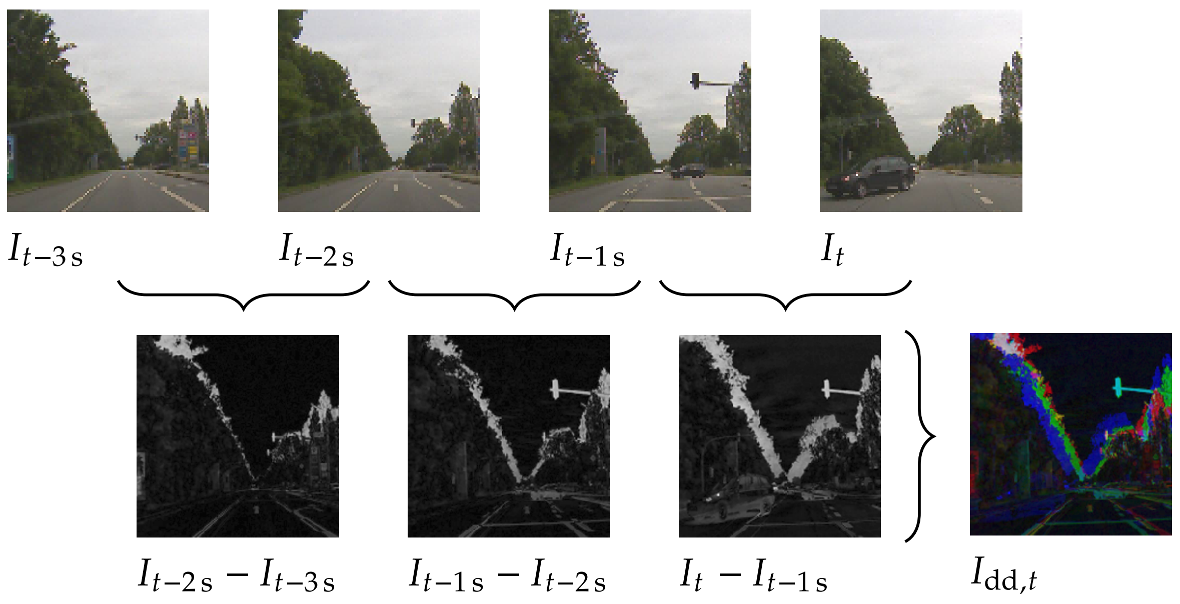

$I_{t-3\,\mathrm{s}}$
$I_{t-2\,\mathrm{s}}$
$I_{t-1\,\mathrm{s}}$
I_t
$I_{t-2\,\mathrm{s}} - I_{t-3\,\mathrm{s}}$
$I_{t-1\,\mathrm{s}} - I_{t-2\,\mathrm{s}}$
$I_t - I_{t-1\,\mathrm{s}}$
$I_{\mathrm{dd},t}$

Figure 3.9 Visualization of the creation of a dynamic difference image $I_{\mathrm{dd},t}$ from a sequence of four images I_t, $I_{t-1\,\mathrm{s}}$, $I_{t-2\,\mathrm{s}}$, and $I_{t-3\,\mathrm{s}}$. The image I_t is taken at the moment of a disengagement. The relevant features such as the car cutting in from the right and the approaching intersection, indicated by the traffic light, are highlighted in the difference images.

The current image I_t in Figure 3.9 was taken at the moment of the disengagement from this particular failure sequence. At the intersection, a car cuts into the path of the ego vehicle from the right at the last second. When looking at the difference images leading up to this critical maneuver of the car coming from the right, it can be seen that the most relevant features are captured and highlighted. The quickly approaching intersection can be detected in the first difference image via the mostly white traffic lights over the road at the horizon. In the second difference image, the car cutting in from the right becomes visible as a light gray area on the road. In the third and last difference image, the car blocking the road is clearly visible. The fact that the intersection has been entered is captured by the traffic light still being part of the difference image, this time caused by its disappearance from the last frame. Combining the three images into the dynamic difference image $I_{\mathrm{dd},t}$ yields a compact representation where the different time steps are coded by the respective color channel in which they are stored.

The processing into dynamic difference images is applied to every 3 s sample. The same $N_F = 2549$ disengagement sequences as before are obtained. An equal number $N_S = N_F$ of success sequences is randomly sampled. To allow a direct comparison, the same success sequences are used that were sampled to train the state-based classifier. Using the sample generation process shown in Figure 3.2, each failure sequence is split into 70 dynamic difference images $I_{\mathrm{dd},n_F,i}$ labeled as *Failure* and each success sequence is split into 70 dynamic difference images $I_{\mathrm{dd},n_s,j}$ labeled as *Success*. The image-based failure data set D_{img} can then be described as

$$D_{\mathrm{img}} = \{I_{\mathrm{dd},n_F,i}, I_{\mathrm{dd},n_S,j}\},\ i,j \in \{1,70\},\ n_{F,S} \in \{1, N_F\}. \tag{3.7}$$

As a final processing step, each image is cropped into a square and then scaled to a resolution of 224×224. This is done since image classification networks pretrained on ImageNet [41] typically are trained on this input size. The data set D_{img} is split into the same 80 % for training, 10 % for validation and 10 % for testing as the state-based data set D_{state}. Next, the model for classifying each image-based sample as *Failure* or *Success* is designed.

3.3.2 CNN-Based Classification Approach

Given the image-based failure data set D_{img}, a classifier C_{img} is required that can extract features from the processed images and classify them as either *Failure* or *Success*. This is again a binary classification task, with the score of the *Failure* class being used as the failure probability $P_{\text{img},t}$. The failure probability $P_{\text{img},t}$ is assigned to a given dynamic difference image $I_{\text{dd},t}$ by the classifier C_{img}:

$$P_{\text{img},t} = C_{\text{img}}(I_{\text{dd},t}),\ P_{\text{img},t} \in [0,1],\ I_{\text{dd},t} \in \mathbb{R}^{224\times224\times3} \tag{3.8}$$

Due to the preprocessing of the video data, the information from each 3 s sample is stored as a single three-channel image. Since the dynamic difference images already contain temporal features, the classifier C_{img} only needs to extract spatial features to then perform the classification. For classifying images, Convolutional Neural Networks (CNNs) are a powerful feature extraction method. CNN-based architectures for image classification have received significant attention over the last decade, with a wide selection of state-of-the-art image classification models being available [40, 43]. By posing image-based introspective failure prediction as a binary image classification task, these state-of-the-art architectures can be exploited to perform the failure prediction task. The driving scenes from the image-based data set D_{img} contain natural objects in natural environments. This allows to make direct use of architectures pretrained on ImageNet [41], which contains over a million natural images from 1000 classes. Since low-level features such as edges, corners, and simple shapes are present in most natural images, the early layers of CNNs trained on ImageNet are useful general feature extractors. For using such models on new data sets, it can be sufficient to finetune only the later layers to the desired domain. Since disengagements are rare events and it is time-consuming to record large amounts of them, failure data sets are inherently limited in size. Being able to use pretrained models is therefore especially relevant for the task of system-level failure prediction.

There are several options regarding which specific classification architecture to use. Daftry et al. [111], who first proposed the concept of introspection to predict crashes of autonomous drones, used the *AlexNet* architecture [40] to classify input images as failures. The classification performance of *AlexNet* has since been improved by a range of newer architectures, such as residual networks [43]. In this thesis, *AlexNet* as well as *ResNet18* and *ResNet50* are used and compared. As a final architecture option, *MobileNetV2* [44] is selected due to its focus on computational efficiency. In the case of restricted hardware capabilities of a mobile system such as an autonomous vehicle, a compact architecture can be a crucial property.

All four architectures are pretrained on the ImageNet data set. Then, they are finetuned with the dynamic difference images from the training set of the image-based failure data set D_{img}. The last layer is changed from predicting one of 1000 classes to a binary classification layer. Due to the relatively small size of D_{img} and the high dimensionality of image data, overfitting is a potential issue. For all models, only the last layers are trained to avoid overfitting to the training set. The number of trainable layers is selected empirically using the validation set. If more failure data becomes available, those numbers could be further increased. The architectures along with the numbers of trainable layers that are used in this dissertation are summarized in Table 3.4.

Architecture	Layers	Trainable Layers
MobileNetV2	53	20
AlexNet	8	5
ResNet18	18	10
ResNet50	50	10

Table 3.4 Summary of the classification architectures used for the image-based classifier C_{img}. For each architecture, only the final layers are trained to avoid overfitting.

The optimal number of trainable layers varies significantly among the models due to their different structures, ranging from more than half the layers for *AlexNet* to just 20 % of the layers for *ResNet50*. This can be explained by *AlexNet* containing three FC layers at the end, whereas *ResNet50* contains only a single FC layer. For finetuning a relevant amount of convolutional layers, more than half the layers of *AlexNet* need to be made trainable. This means that significantly more parameters, both in absolute and relative terms, need to be retrained when using *AlexNet* to implement the classifier C_{img} compared to using *ResNet50*.

For training the architectures, the Adam optimizer [168] is used. The binary cross entropy loss is chosen as a loss function and the learning rate is empirically set to 0.0005. For all models, the decrease of the validation loss is monitored and used as a stopping criterion in case it starts increasing again. The resulting training durations range from 10 epochs for *MobileNetV2* to 20 epochs for *ResNet50*. As a standard technique to further avoid overfitting, data augmentation is applied to all training images. Random rotations of up to 10 degrees, random translations of up to 10 pixels and random cropping by up to 30 % of the image width followed by rescaling to the original resolution are used during training to make the models more robust.

Finally, the predicted failure probabilities $P_{\text{img},t}$ are filtered with a moving average filter. The output filtering is applied for the same reasons presented in the previous section when designing the state-based failure prediction approach. C_{img} predicts new failure probabilities at 10 Hz and uses information from the previous three seconds, same as the state-based classifier C_{state}. Therefore, the same output filter with a horizon $H = 30$ is applied. The filtered failure prediction $P_{\text{img},t}$ from C_{img} given a dynamic difference image $I_{\text{dd},t}$ is therefore calculated as

$$P_{\text{img},t} = \frac{1}{\min(t, H)} \sum_{k=1}^{\min(t,H)} P_{\text{img},t+1-k}. \tag{3.9}$$

Same as before, the number of available observations in the 10 s sequences is below H while $t < H$. During the first three seconds of each sequence, the average is therefore computed using only the last t predictions.

3.3.3 Results

In this section, the results of the experiments designed to evaluate the proposed image-based failure prediction approach are presented. First, two state-of-the-art reference approaches are introduced to compare to the proposed introspective method. Then, the attention of the investigated architectures is visualized using their Class Activation Maps (CAMs). The failure prediction performance is evaluated using an ROC analysis and by analyzing the prediction accuracy over time. Finally, failure cases of the image-based classifier are discussed.

3.3.3.1 Reference Approaches

For the state-based approach, no comparable reference methods were available in the literature. While image sequences have been used for failure prediction in the literature before [15, 16, 167, 111], to the best of our knowledge, there is no existing work that explicitly predicts the disengagements of a fully autonomous car outside of simulation. Therefore, the proposed image-based method is compared to two works that use substitutes of real self-driving vehicles. Fridman et al. [15] used video sequences to predict disengagements of a Level 2 car and Michelmore et al. [16] used video data to predict crashes of an autonomous car in a simulation. Both reference approaches are explained in more detail next to allow a better understanding of the comparison to the proposed approach.

The method proposed by Fridman et al. [15] will be referred to as *Arguing Machines*. They recorded disengagements of a Tesla driving in autopilot on highways. Additionally, they recorded successful driving data of the Tesla to train a state-of-the-art end-to-end steering angle prediction network [34]. The normalized absolute difference between the steering angle generated by the Tesla and the steering angle predicted by their end-to-end network is used as a failure probability. To implement their approach, both the data set D_{img} and D_{state} are needed. The success sequences from D_{img} are used as training data for the same end-to-end steering angle prediction network. As target labels, the corresponding steering angles θ from D_{state} are used. Fridman et al. undersampled driving data where the angle is close to zero to avoid the steering angle predictor from overfitting to that value. The same approach is applied to the driving data used in this work. The end-to-end network trained using the corresponding data from D_{img} and D_{state} predicts the steering angle of the validation set at a Mean Average Error (MAE) of 1.2. The model trained by Fridman et al. achieved a comparable MAE of 1.1, having been trained on a significantly larger training set.

The second reference approach follows the work of Michelmore et al. [16], who trained an end-to-end steering angle predictor in a driving simulator using the same architecture as Fridman et al. [15]. Then, they estimated the uncertainty of their end-to-end network and use that estimate as a failure probability, allowing them to predict crashes five seconds in advance. They estimated the uncertainty by computing the variance of multiple Monte Carlo (MC) dropout samples [10], thus their approach is referred to as *Predictive Variance* in the following. In this work, Deep Ensembles [12] are used instead of MC dropout since they have shown superior performance for uncertainty estimation. They also allow estimating uncertainty in real-time, since only five forward passes are required as opposed to 10 or more for MC dropout. Using the training and validation sets of D_{img} and D_{state}, the end-to-end network is trained five times with different initializations to obtain the Deep Ensemble. All five models achieve an MAE of around 1.2. For each new input image, five steering angles are predicted and the variance among them is used as the failure probability.

3.3.3.2 Feature Visualization

First, the spatial focus of the trained image classification networks is visualized to allow some insights into what they learned. Class Activation Maps (CAMs) [102] are a straightforward method to visualize which regions of an input image had the largest influence on the prediction of each model. They are calculated by multiplying the last spatial feature map in each network architecture with the weights of the connection between the last convolutional layer and the neuron of the predicted class in the last FC layer. This way, each high-level spatial feature is assigned a weight that describes how much that feature activated the neuron of the predicted class. The resulting spatial weight map can then be visualized over the input image as a heatmap to show which regions led to the model making its current decision.

In case there is more than one FC layer after the last convolutional layer, gradient-weighted Class Activation Maps (grad-CAMs) [103] can be used. They are obtained by calculating the gradient through the last FC layers and weighting the last spatial feature map with the gradients from the neuron of the predicted class. Since *AlexNet* contains three FC layers at the end, grad-CAMs are used for this architecture.

The CAMs allow to visualize which regions of an image led the introspective failure prediction models to predict a failure or a success. Since machine learning models are largely black boxes, such a visualization is a useful step for obtaining some intuitive understanding of how the models work. In Figure 3.10, two example images from success sequences and two examples from failure sequences are shown. All four architectures predicted the corresponding classes correctly. The dynamic difference image that the models received as input is shown as well. Finally, the CAM of each architecture is shown as a heatmap on top of the input image.

The focus of *AlexNet* is much more scattered than the other architectures, most notably for the images from success sequences. For the failure images, areas like the crane, traffic light and cars in front of the vehicle are highlighted, but seemingly spurious areas also have an impact on the classification. This can be explained by grad-CAMs being affected by the gradient passing through several layers before the weights of the spatial feature map are computed. While the predictions of this architecture are correct for these examples, this lack of intuitive interpretability is a downside. It is challenging to evaluate if sensible regions have been used for the failure predictions or not.

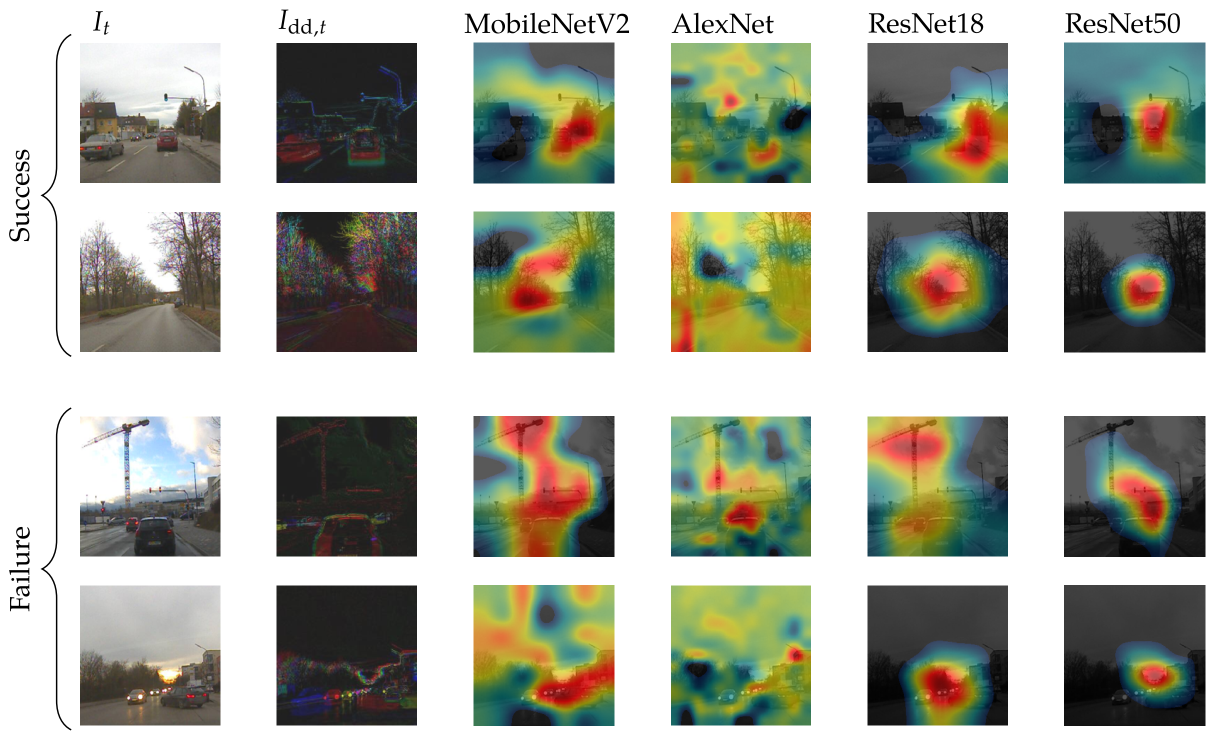


Figure 3.10 Visualization of the focus of each investigated architecture by plotting the Class Activation Maps (CAMs) as heat maps over sample test images from both failure and success sequences. The *ResNet50* model has learned to detect the most distinct and focused regions to make its predictions (adopted from [25] © 2020 IEEE).

While the CAMs of *MobileNetV2* are also not precisely focused on distinct regions, they do show that critical objects have been correctly detected. In the top row, both the traffic light and the car that is still far away from the ego vehicle are used to classify this scene as *Success*. The empty road as well as the sky devoid of any traffic lights also explains the correct prediction of the second row. Regarding the correctly predicted failures in the third and fourth row, most of the cars in front of the vehicle are used for this classification as well as the crane indicating a construction site.

The observations for the two ResNet variants are largely similar to the ones made for *MobileNetV2*. The main difference is that the focus of *ResNet18* is more spatially restricted. The CAMs of *ResNet50* are even more locally constrained, with one distinct region being used to classify each image. In the success images, the car in front and the empty road are used by both ResNet models to assess the scene. For the bottom row, both models focus on the complex traffic scenario with a car cutting in from the right. For the third row, the *ResNet18* model puts more emphasis on the crane. Only the *ResNet50* model focuses entirely on the traffic light and car in front of the vehicle.

Having gained first insights into how the models operate, the performance of the four models is evaluated quantitatively next.

3.3.3.3 Failure Prediction Performance

The failure prediction performance is evaluated on the test set consisting of 510 sequences. First, the four implementations of the proposed image-based approach are compared to the two reference approaches *Arguing Machines* [15] and *Predictive Variance* [16] in an ROC curve analysis. The state-based model C_{state} from the previous section, referred to as *State LSTM* for short, is also included in the analysis. This allows to evaluate the improvements achieved by using images instead of state data sequences. Figure 3.11 shows the ROC curves and the corresponding AUC of all compared models.

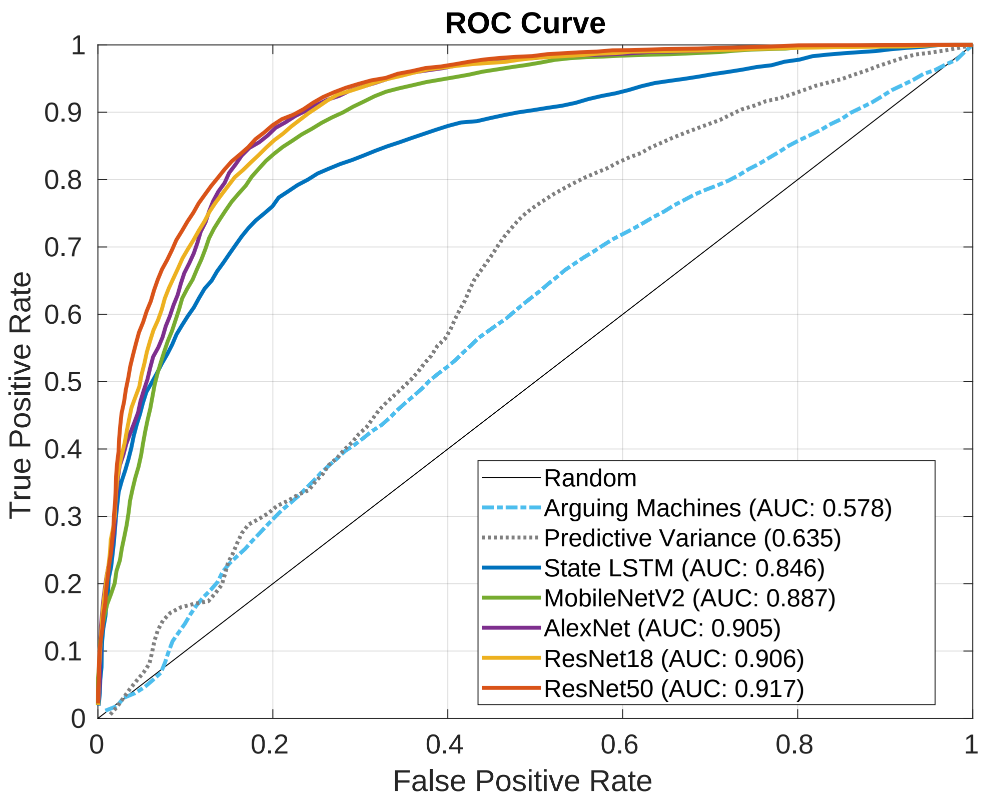


Figure 3.11 Comparison of the ROC curves of the two reference approaches of *Arguing Machines* [15], *Predictive Variance* [16], the state-based approach *State LSTM* from Section 3.2, and the four architectures used to implement the proposed image-based approach. Both the state-based and the image-based introspective models significantly outperform the state of the art by at least 33 %.

The proposed image-based failure prediction concept significantly outperforms both of the two state-of-the-art reference approaches. The lowest performing architectures, *MobileNetV2*, still has an AUC that is 53 % larger than the AUC of *Arguing Machines* and 39 % larger than the AUC of *Predictive Variance*. The large differences can be explained by the significant conceptual differences between the proposed introspective approach and the state of the art. *Arguing Machines* relies on differences between two models without any guarantee or indication that the differences will be informative about failures. *Predictive Variance* assumes a correlation between model uncertainty and failures, which performs better than *Arguing Machines* but still lacks the explicit failure information that the introspective models were trained with. It should be noted that the two reference approaches do not require failure data and are therefore more straightforward to apply to any given system. The large performance differences to the system-specific failure prediction approach demonstrate the benefit of the additional work of obtaining recorded failures of the given driving system.

The state-based approach from the previous section also outperforms the image-based reference approaches by at least 33 %. This result also indicates that exploiting explicit failure data instead of using generic uncertainty estimation techniques is a useful approach for failure prediction.

Among the proposed concepts, the image-based models outperform the state-based approach by 4 % to 8 %. The significantly higher-dimensional input of images therefore allows the model to better distinguish between failures and successes than when given only the low-dimensional state data. The performance of the four image-based classification architectures is similar, with the most complex *ResNet50* model outperforming the more compact *MobileNetV2* architecture by 3 %. At AUCs of 0.905 and 0.906, *AlexNet* and *ResNet18* exhibit almost the same performances.

In addition to the AUC values, Table 3.5 also summarizes the average accuracy and standard deviation, the number of parameters, and the average inference time per prediction for all compared methods.

Model	Accuracy	AUC	Parameters	Inference Time
Arguing Machines	56.6 % (±0.15)	57.8 %	0.37×10^6	18.2 ms
Predictive Variance	63.1 % (±0.09)	63.5 %	1.84×10^6	81.4 ms
State LSTM	78.2 % (±0.08)	84.6 %	0.02×10^6	3.8 ms
MobileNetV2	81.0 % (±0.07)	88.7 %	3.5×10^6	59.9 ms
AlexNet	83.5 % (±0.06)	90.5 %	61.0×10^6	56.2 ms
ResNet18	82.3 % (±0.06)	90.6 %	11.7×10^6	53.9 ms
ResNet50	83.6 % (±0.05)	91.7 %	25.6×10^6	59.5 ms

Table 3.5 Comparison of the average accuracy and standard deviation, the AUC, the number of parameters, and the average inference time per prediction of all investigated failure prediction methods.

Regarding the average accuracy, the state-based approach outperforms the state of the art by at least 23 %, while the image-based models outperform the reference approaches by up to 32 %. The standard deviation averaged over all time points of the test sequences is highest for *Arguing Machines* at 0.15, while the best-performing *ResNet50* has the lowest spread at 0.05. For all models, a higher accuracy also means a lower standard deviation and a higher AUC.

The main limitation of the proposed image-based approach is the computational complexity. All architectures are more than three times as slow as *Arguing Machines* and around 15 times slower than the *State LSTM*. The Deep-Ensemble-based approach *Predictive Variance* is the most expensive approach, since it requires five forward passes to obtain an output distribution to calculate a variance with. All compared methods require less than 100 ms and are therefore able to generate failure predictions in real time at 10 Hz.

Finally, the number of parameters is another relevant metric, describing how many parameters need to be updated during training and how much disk space is required for storing and loading the model. The *State LSTM* has the fewest parameters, demonstrating the efficiency of operating on low-dimensional state data only. At around 3.5 million parameters, *MobileNetV2* is the most compact image-based introspective model. At around 61 million parameters, *AlexNet* is the largest model since it contains three FC layers at the end, which can require significantly more parameters to store than convolutional layers. *ResNet50* outperforms *AlexNet* both in accuracy and in AUC, despite having only around 25 million parameters. In summary, the *ResNet50*-based implementation of the image-based approach shows the most useful trade-off between complexity and accuracy. For all further experiments, we will therefore only use the *ResNet50* model as the implementation of C_{img}.

Besides the overall performance, the prediction capabilities over time are of interest as well. In Figure 3.12, we therefore show the average accuracy of each time point in both failure and success sequences for the best-performing reference approach of *Predictive Variance*, the state-based model *State LSTM*, and the best-performing image-based model *ResNet50*. The reference approach has an accuracy of 50 % seven seconds before a disengagement, indicating that this early in advance, its predictions are effectively random. While the performance of *Predictive Variance* increases when the disengagement approaches, both the *State LSTM* and the *ResNet50* significantly outperform it across the entire sequence duration. Among the introspective models, the image-based approach consistently outperforms the state-based approach, most notably when the disengagement is still more than three seconds away. Seven seconds in advance, the *ResNet50* still correctly predicts over 82 % of all disengagements.

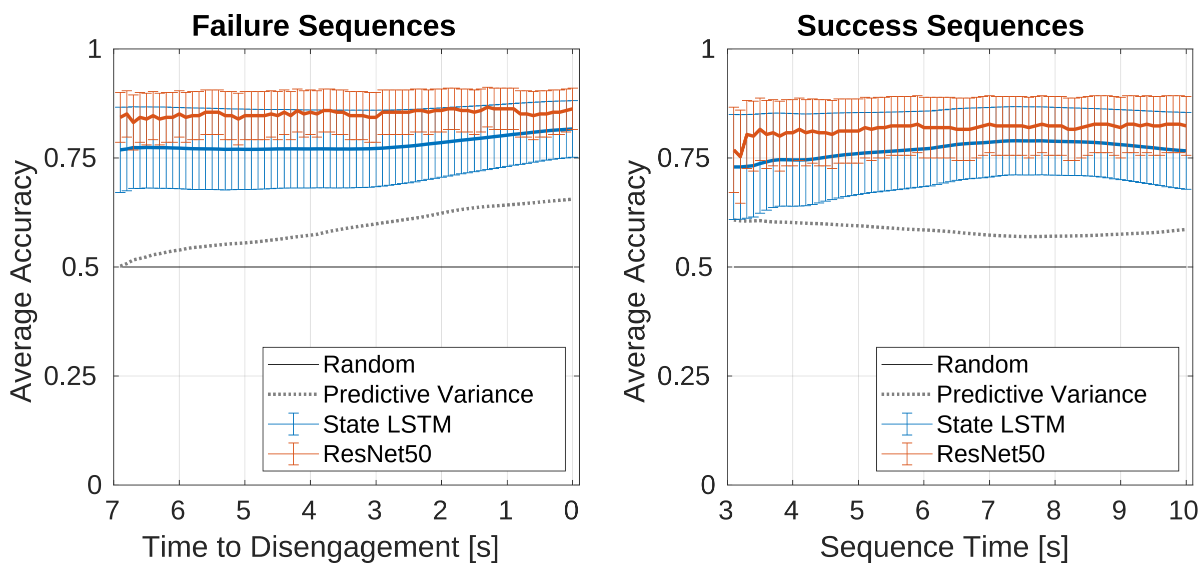


Figure 3.12 Average accuracy over time of the best-performing reference approach *Predictive Variance* compared to the proposed state-based and image-based classifiers.

3.3.3.4 Failure Cases

While using image sequences as input achieved a better failure prediction performance than the state-based approach, the *ResNet50* model still has an average error rate of 16.4 %. To obtain an intuitive understanding of what those failure cases have in common, the average speed and angle of all failure cases from both success and failure sequences are shown in Figure 3.13.

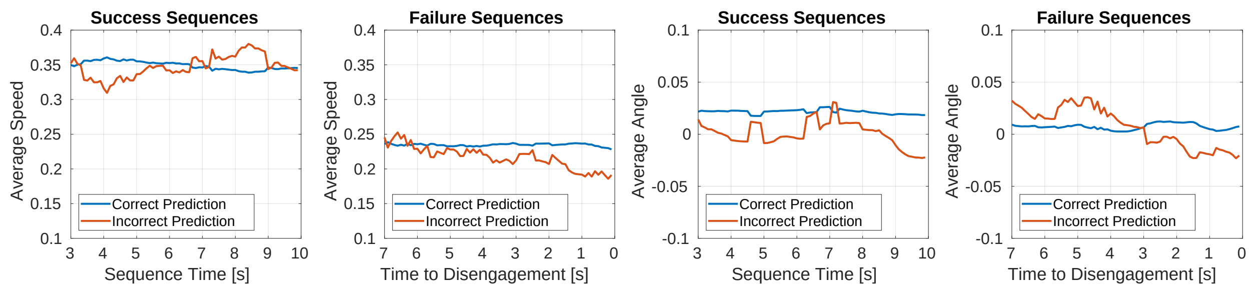


Figure 3.13 Visualization of the average speed and the average angle of failure cases of the *ResNet50*.

The average speed during correct predictions is almost constant in both success and failure sequences, with the speed during correctly predicted failures being much lower than during correctly predicted successes. A possible explanation is that the *ResNet50* has learned the most common degree of differences between subsequent images for both success and failure sequences. If a difference image shows smaller or larger differences due to a lower or higher speed, the model is more likely to make an incorrect prediction.

Regarding the angle, correctly predicted failures exhibit an average angle of almost zero. The average angle of incorrectly classified successes also shows an average angle of close to zero. Meanwhile, larger angle variations that lead to horizontal changes in the difference image tend to be classified as *Success*. This observation is in contrast to the failure cases of the state-based approach, where the model predicted the class *Success* mostly for sequences with a small angle. Such differences between the state-based and the image-based approach suggest that combining different input modalities could be a promising direction for mitigating the failure cases of the individual approaches.

3.4 Trajectory-Based Introspective Failure Prediction

In this section, we propose a third introspective failure prediction method for system-level failures. So far, the sensory input and the state of the car have been used as training data. In this section, the planned trajectories generated by the system itself are used to infer impending failures. This approach still treats the car as a black box and only requires its output.

Next, we discuss how the trajectories as planned by the car can be used for failure prediction and how to generate a suitable data set for failure prediction. Then, an LSTM-based model similar to the state-based approach is developed, which is then trained with the generated data set. Finally, the failure prediction performance of the trajectory-based approach is evaluated thoroughly.

3.4.1 Concept

As discussed in Section 3.1, any recorded driving data from disengagements can be used as input to train an introspective model. The state-based approach from Section 3.2 used state sequences, which correspond to the driven trajectory of the car. Now, we use the planned trajectory calculated by the vehicle at each time step instead. While the state data of the car reflects its current behavior, the planned trajectories give insights into the planned future behavior of the car. For the task of early failure prediction, the planned future behavior by the vehicle can therefore be a valuable additional source of information. Changes in the planned trajectory can take seconds until they manifest themselves in a changed physical state of the system. An obstacle or challenging situation that is still seconds away can already lead to changes in the planned trajectories, however. A simple toy example of such a scenario is shown in Figure 3.14.

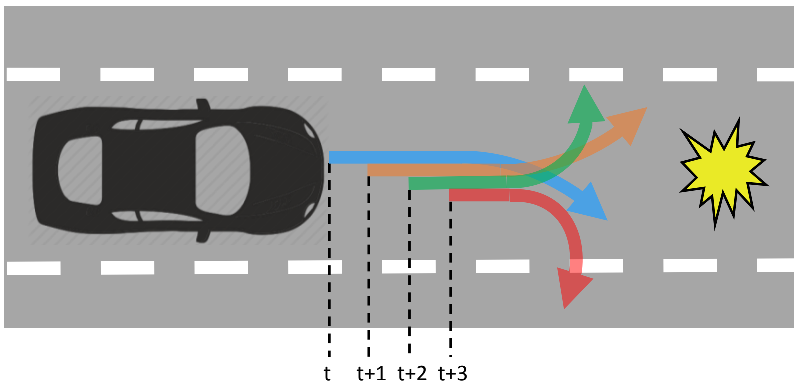


Figure 3.14 Exemplary visualization of the potential of using planned trajectories to predict failures early on. Between the time t and $t + 3$, the car drives a straight line, but the planned trajectories already contain large disagreements regarding the future behavior.

Figure 3.14 shows a car driving towards an obstacle that the car is not sure how to handle. When faced with such a challenging scenario, a system can exhibit its uncertainty in the planned trajectory while the actual trajectory is not yet affected. In Figure 3.14, the car plans four different evasion maneuvers at the time steps t to $t + 3$. During the same time period, its actual trajectory is still a straight line. The disagreement in the sequence of planned trajectories can therefore be a useful early indicator of impending challenges. Previous works have used the disagreement between different models [135, 15] or between multiple predictions sampled from the same model [10] for uncertainty estimation before. Here, we propose to use the disagreement of the model with itself over time. While an impending challenging situation is not guaranteed to lead to a system-level failure, our hypothesis is that the disagreement over time in a trajectory sequence is correlated to such failures. Next, the generation of a suitable data set consisting of such trajectory sequences is presented.

3.4.2 Data Set Generation

In the driving data provided for this work, each planned trajectory consists of the next 30 points to be driven in the next three seconds. The points are time-fixed, meaning each point is planned to be reached 100 ms after the previous one. Trajectories are planned at 10 Hz, which is equal to the sampling frequency used in the previous sections for state and image data. Same as before, sequences of a duration of 10 s are used. From the provided driving records, all disengagement sequences are extracted, resulting in N_F failure sequences. The same number $N_S = N_F$ of successful sequences is sampled to obtain a balanced data set. At a frequency of 10 Hz, each trajectory sequence S_n therefore contains 100 planned trajectories. Each planned trajectory T_t^n of the sequence S_n consists of 30 points p_i. The trajectories T_t^n and the trajectory sequences S_n can be described as

$$T_t^n = [p_1, p_2, ..., p_{30}], \quad p_i \in \mathbb{R}^2, \quad t \in [1, 100], \tag{3.10}$$
$$S_n = \{T_1^n, T_2^n, ..., T_{100}^n\}. \tag{3.11}$$

Same as in the previous sections, the 10 second sequences S_n are split up into shorter overlapping samples s^{traj} consisting of L consecutive planned trajectories. We again use a sample length $L = 30$, creating 71 samples from each sequence. All N_F trajectory sequences ending in disengagements and all N_S trajectory sequences from undisrupted automated driving are then combined into the trajectory-based data set D_{traj}. Each trajectory from a sequence ending in a disengagement is labeled as *Failure* and each trajectory from a successful sequence is labeled as *Success*. The resulting data set D_{traj} is thus given as

$$D_{\text{traj}} = \{s_{n_F,i}^{\text{traj}}, s_{n_S,j}^{\text{traj}}\},\ i,j \in \{1, 100 - L + 1\},\ n_{F,S} \in \{1, N_F\}. \tag{3.12}$$

The driving data used to generate D_{traj} is not completely identical to the driving data used for the state-based and the image-based approach due to the planned trajectories not being available in all initial recordings. To obtain a data set of comparable size to D_{state} and D_{img}, an additional month of driving records is therefore used for the trajectory-based approach. This way, a similar amount of 2624 unique disengagements can be extracted from the raw data. In total, the data set D_{traj} contains over 370 000 labeled samples of a duration of three seconds. The contents of the data set D_{traj} are summarized in Table 3.6.

N_F	2624
Sampled N_S	2624
Sequence length	10 s
Planning frequency	10 Hz
Trajectory content	Next 30 points

Table 3.6 Contents of the trajectory-based data set D_{traj}.

Since the trajectories are recorded in world coordinates, they need to be normalized before being used to train machine learning models. We normalize the orientation of the 30 trajectories contained in each sample by using the driving direction of the car in the last trajectory as a reference line. The angle between the reference line and the x axis in Cartesian coordinates is calculated and each trajectory is rotated by that angle. This way, the direction of the last trajectory in each sample is always the same and the differences of the rest of the trajectory sequence are always in reference to the same line. This normalization step prevents a model from overfitting to whether the car was driving in a specific global direction such as North or West. As before, the data set D_{traj} is split into three subsets. 80 % are used for training, 10 % are used or validation and 10 % are reserved for testing.

3.4.3 Model Design

Next, a classification model C_{traj} is designed to predict failures based on the sequences of planned trajectories. Given an input sample s_t^{traj} consisting of L consecutive trajectories containing 30 planned points $p_i \in \mathbb{R}^2,\ i \in \{1, 30\}$, the model C_{traj} needs to assigns a failure probability $P_{\text{traj},t}$ to that sample as follows:

$$P_{\text{traj},t} = C_{\text{traj}}(s_t^{\text{traj}}),\ P_{\text{traj},t} \in [0,1],\ s_t^{\text{traj}} \in \mathbb{R}^{2\times30\times L} \tag{3.13}$$

Each trajectory consists of 60 values and is therefore more complex than the state vectors consisting of five values each. Compared to images, both state and trajectory data can be considered to be low-dimensional input. The task of learning to extract temporal patterns from sequences of low-dimensional data is similar. For the classifier C_{traj}, we therefore design an architecture that is similar to the LSTM-based classifier C_{state}. Preliminary results on the validation set showed that a deeper architecture with more neurons compared to C_{state} achieves the best classification performance. The architecture of C_{traj} therefore consists of two LSTM layers with 100 neurons each, followed by four FC layers. The first two FC layers also have 100 neurons each, followed by one layer with 50 and a final layer with 2 nodes. This triangle structure of reducing the size of subsequent FC layers allows to reduce computational complexity without affecting the accuracy of the model. At the end, a softmax layer assigns the two classes *Failure* and *Success* their respective class probabilities. We use the probability of the class *Failure* as the final output P_{traj}.

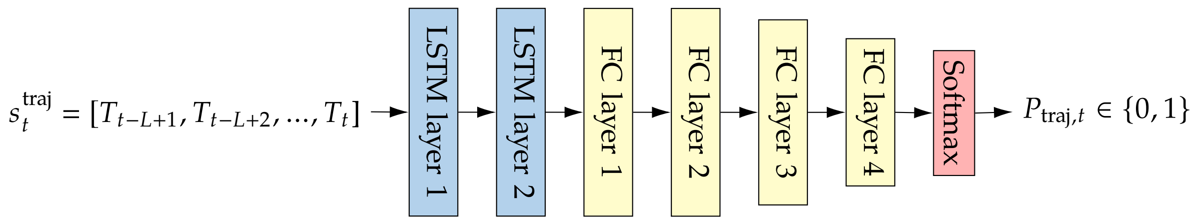

$s_t^{\text{traj}} = [T_{t-L+1}, T_{t-L+2}, \ldots, T_t]$
LSTM layer 1
LSTM layer 2
FC layer 1
FC layer 2
FC layer 3
FC layer 4
Softmax
$P_{\text{traj},t} \in \{0,1\}$

Figure 3.15 Overview of the architecture of the trajectory-based classifier C_{traj} that assigns a failure probability $P_{\text{traj},t}$ to an input sample s_t^{traj} consisting of a sequence of planned trajectories T_t. Two LSTM layers are followed by four FC layers and a final softmax layer to obtain the output score between 0 and 1.

The architecture in Figure 3.15 is trained with the training and validation set obtained from D_{traj}. Similar training parameters as used for the state-based approach are selected for the training process. Adam [168] is used as the optimizer to minimize the binary cross-entropy loss function. A learning rate of 0.0001 is used and training is stopped either after 20 epochs or if the validation loss has not decreased for more than five epochs.

At a prediction frequency of 10 Hz, the failure predictions can contain spurious outliers that need to be removed in a post-processing step. Same as with the state-based and image-based approaches, we apply a moving average filter with horizon $H = L$ to ensure that a failure is only predicted if the predicted failure probability P_{traj} is above 0.5 for multiple consecutive predictions. As before, the moving average is computed over less than H predictions at the beginning of each sequence when $t < H$ holds. The final, filtered predicted failure probability P_{traj} generated by the classifier C_{traj} is then given as

$$P_{\text{traj},t} = \frac{1}{\min(t,H)} \sum_{k=1}^{\min(t,H)} P_{\text{traj},t+1-k}. \tag{3.14}$$

3.4.4 Results

Next, the trajectory-based introspective failure prediction approach is evaluated. To obtain an intuitive understanding of what the model has learned, we first visualize the training data. Then, we introduce two baseline approaches trained with hand-crafted features to evaluate if the proposed deep-learning-based model is the most suitable choice for this problem. We also compare the trajectory-based approach to the state-based approach from Section 3.2 to evaluate if the use of planned trajectory data offers improvements over using the less complex state data sequences as input. All models are compared using an ROC analysis and by evaluating the average accuracy over time. Finally, failure cases of the trajectory-based model are discussed.

3.4.4.1 Data Visualization

We visualize two exemplary trajectory sequences from both the *Success* and the *Failure* class in Figure 3.16. For improved visibility, only every tenth of the 100 trajectories from each sequence is plotted. All plotted trajectories are therefore planned one second apart. The starting point of each trajectory is marked as an "X". The color indicates the time within the sequence. Dark blue corresponds to the earliest sequence time, with the disengagement still being nine seconds away in case of the two failure examples. The moment of the disengagement is shown in red. The trajectories are shown in world coordinates to make the shape of the road they were planned for visible. The only modification is that the start of each sequence was shifted to the origin. For training the trajectory-based classifier, the trajectories are normalized by rotating the trajectories as discussed in the previous section.

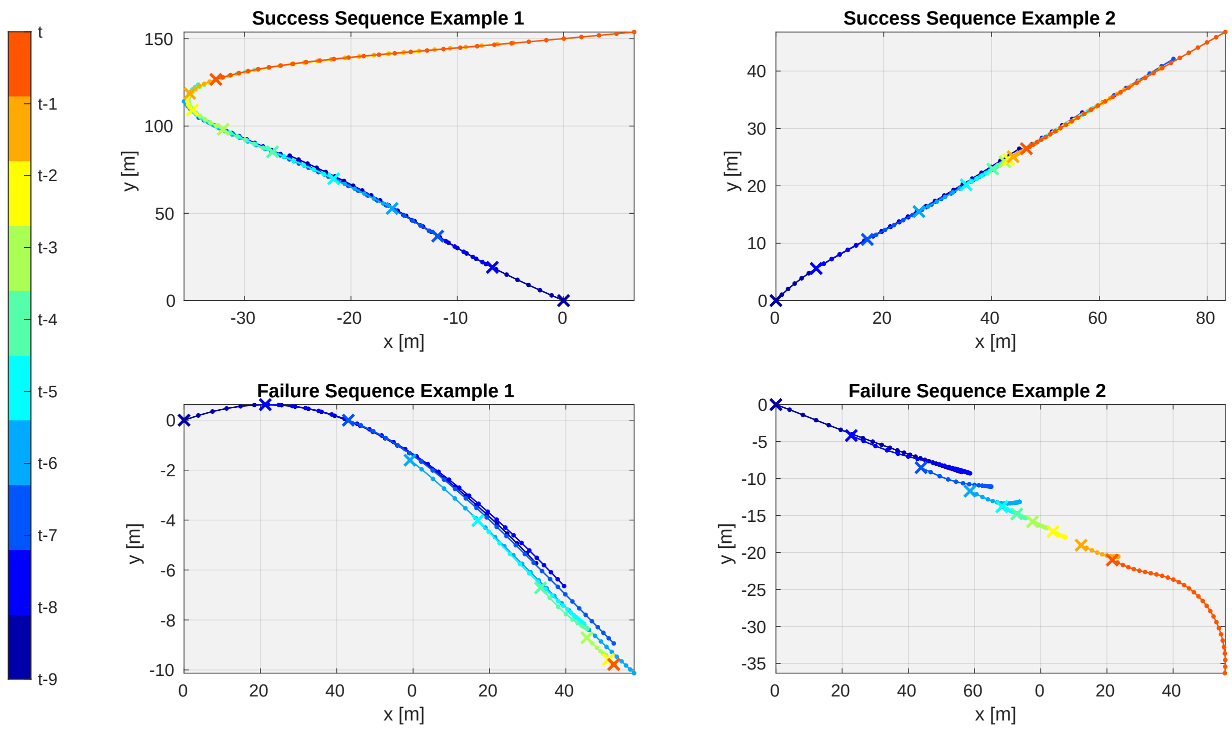


Figure 3.16 Visualization of two trajectory sequences from successful driving (left) and two trajectory sequences ending in disengagements (right). The start of each trajectory is marked by an "X" and the color corresponds to the time point within each sequence. The planned trajectories in the two success examples largely overlap. In the failure sequences, significant disagreements over time are visible.

In the success sequences, the planned trajectories largely overlap. In the left example, there are some minor variations during the left turn in the corner, but no significant disagreements are visible. The changes that lead to the car driving through the corner are gradual and the car tends to drive exactly its planned route, evident by the starting point of the next trajectory being mostly on top of the previous planned trajectory. In the example on the right in Figure 3.16, the car is decelerating during the sequence, but does so without noticeably changing its planned route. The vehicle's reaction to the traffic is again consistent over time.

In the first failure sequence example on the left of Figure 3.16, the first three trajectories still largely overlap. Then, noticeable disagreements over time become visible. At $t-7$, the car had intended to drive on the left side, but subsequent planned trajectories indicate that the car remains on the right side until the moment of the disengagement. The planned trajectories until that point keep proposing sharper turns to still perform the lane switch, increasing the disagreement between the trajectories over time. At the end, the planned trajectory is to halt the car, at which point the human safety driver resumed control. In the second failure example in Figure 3.16, a similar behavior can be seen with an attempted lane switch to the left. After repeatedly planning such a maneuver, the system ultimately stops attempting the switch and the human again needs to take over.

These examples were chosen to highlight the potential differences between success and failure sequences of planned trajectories. In undisrupted driving, objects can also suddenly appear or challenging driving situations can arise that require constant adjustments of the planned trajectories. While the resulting trajectory sequences can also disagree over time, our hypothesis remains that the disagreement before a disengagement is more pronounced and corresponding failure patterns can be learned using machine learning. Those disagreements are expressed as variations in the curvature and length of the trajectories. To further investigate this hypothesis, we next compare those characteristics of failure and success sequences explicitly.

To calculate the degree of curvature of a planned trajectory consisting of 30 points $p_i,\ i \in \{1, 30\}$, we use the following approach. For each point p_i, the line between p_i and p_{i+1} and the line spanned by p_{i+1} and p_{i+2} is obtained. Then, the angle $\theta_{traj,i}$ is calculated as the angle between those two lines. For 30 points, this yields 28 angles. The process of calculating the angles of a given trajectory sequence is shown in Figure 3.17.

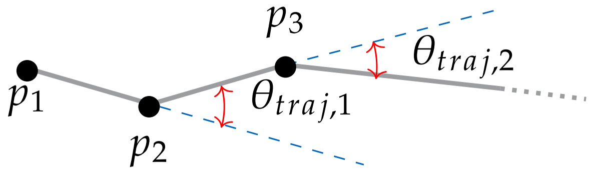


Figure 3.17 The degree of curvature is calculated as the sum of angles $\theta_{traj,1}, \theta_{traj,2}, ..., \theta_{traj,28}$ between 30 points $p_1, p_2, ..., p_{30}$ of a planned trajectory T (adopted from [26] © 2021 IEEE).

For brevity, we refer to the resulting degree of curvature only as "curvature" in the following. For each angle, the absolute value is used since only the degree of variation is of interest, not its direction. For a trajectory $T_t^n = [p_1, p_2, ..., p_{30}]$ from a sequence S_n, the curvature is then given as

$$Curvature_t^n = \frac{1}{28} \sum_{i=1}^{28} |\theta_{traj,i}|, \qquad \theta_{traj,i} = \measuredangle(\overline{p_i p_{i+1}}, \overline{p_{i+1} p_{i+2}}). \tag{3.15}$$

Additionally, the length of each trajectory is calculated as a second characteristic to compare failure and success sequences. The length of a trajectory T_t^n is computed as the sum of distances between each tuple of points p_i and p_{i+1}:

$$Length_t^n = \sum_{i=1}^{29} |p_i - p_{i+1}| \tag{3.16}$$

We calculate the average curvature and length for each time step t over all failure sequences and over all success sequences. The resulting values over time are plotted in Figure 3.18. The curvature is given in degree, while the trajectory lengths are normalized with the largest trajectory length in the training set.

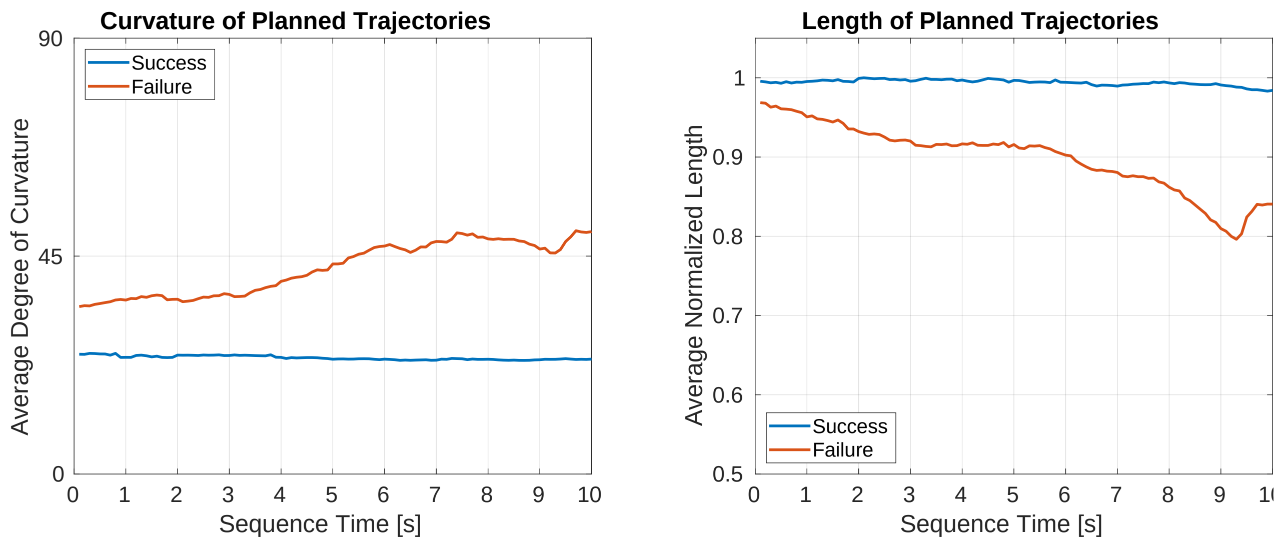


Figure 3.18 Average curvature and length of the planned trajectories in both failure and success sequences (adopted from [26] © 2021 IEEE).

In success sequences, the average curvature is almost constant at around 25°. The curvature of the trajectories in failure sequence is significantly larger than the curvature in success sequences. The curvature also noticeably increases when approaching a disengagement. In the four seconds before a failure, the average curvature is above 45°. This indicates that disengagements are typically preceeded by more drastic intended maneuvers, regardless of whether the car was capable of actually driving those planned maneuvers or not.

When visualizing the average normalized trajectory length, the success sequences again exhibit mostly constant values. The average length in meters of success trajectories is 34.2 m, while trajectories from failure sequences have an average length of 30.9 m. The average length of failure trajectories decreases visibly when approaching a disengagement. Both the decrease in average length and the increase in average curvature of failure trajectories begin already ten seconds before a disengagement. This suggests that planned trajectories are a promising source of information about impending failures even many seconds in advance.

3.4.4.2 Baseline Approaches

Next, three baseline approaches to compare the trajectory-based method are summarized. To the best of our knowledge, no comparable failure prediction methods using low-dimensional data such as planned trajectories exist in the literature. As a baseline, we therefore design two simpler versions of the proposed model.

The investigation of the trajectory curvature and length in the previous paragraph showed a significant difference between failure and success sequences when comparing these two characteristics. It is therefore a reasonable assumption that those two characteristics alone are sufficient to distinguish if a given trajectory belongs to a failure or a success sequence. As a first baseline, we therefore train a Support Vector Machine (SVM) with the curvature and length of each trajectory as input and the corresponding *Failure* or *Success* label as the target. No temporal information is considered for the SVM baseline and no deep learning architecture is employed. This approach is referred to as *Curve+Length SVM* in the following.

As a second baseline, we still use the manually crafted input features of curvature and length, but now use an LSTM-based architecture to evaluate the benefit of introducing temporal information to the model. The same architecture as for the proposed approach is used, shown in Figure 3.15. We refer to the second baseline as *Curve+Length LSTM*.

The proposed approach does not use manual features, but learns to extract the features from the trajectory sequences itself. We refer to it as *Trajectory LSTM*. As a third baseline, we compare to the *State LSTM* from Section 3.2. This comparison allows to evaluate the difference between using the actual state data and the planned trajectory of the vehicle. Since the state-based approach significantly outperformed state-of-the-art failure prediction methods of *Arguing Machines* [15] and *Predictive Variance* [16] in Section 3.3, the comparison to the state-based approach can be considered a comparison to the state of the art in disengagement prediction based on low-dimensional data.

3.4.4.3 Failure Prediction Performance

Next, we evaluate the failure prediction performance of the proposed trajectory-based approach and the three baseline methods. First, an ROC analysis is performed. The ROC curve of the proposed approach and the three baseline methods as well as their respective AUC values are shown in Figure 3.19.

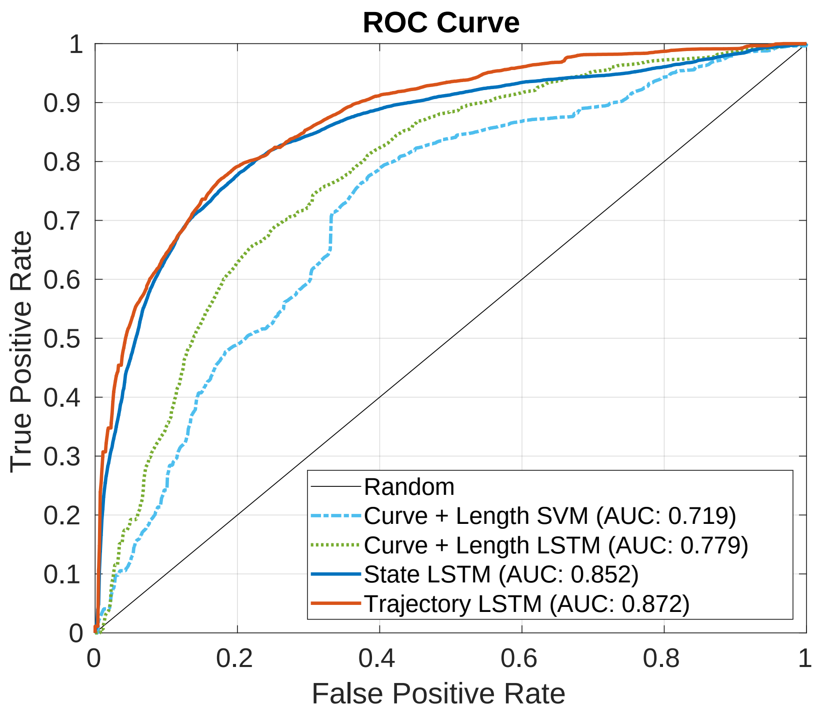


Figure 3.19 Comparison of the ROC curves of the three baseline approaches and the proposed *Trajectory LSTM* model.

The *Curve+Length SVM* approach performs the worst, achieving an AUC of 0.72. Using temporal sequences of manual features improves the failure prediction performance significantly, allowing the *Curve+Length LSTM* model to obtain an AUC of 0.78. Using deep learning to learn patterns from the trajectories directly instead of first extracting manual features such as length or curvature again leads to another significant improvement. The proposed *Trajectory LSTM* approach results in an AUC of 0.87, outperforming the baseline methods that use manual features by at least 11 %. These results indicate that the manual features of curvature and length do allow predicting failures, but are not as informative for distinguishing them as well as an entirely learning-based approach.

Finally, the state-based approach *State LSTM* obtains an AUC of 0.85, outperforming the other baseline approaches but being outperformed by the proposed trajectory-based approach by 2 %. We further compare the proposed trajectory-based model to the state-of-the-art state-based approach in Table 3.7. The accuracy of the *Trajectory LSTM* is higher by around 1 %, while having a lower standard deviation. While the *Trajectory LSTM* has ten times as many parameters and is over 50 % slower than the *State LSTM*, its inference time of below 6 ms is still very fast. The required prediction frequency of 10 Hz can be easily achieved by both models.

Model	Accuracy	AUC	Parameters	Inference Time
State LSTM	78.8 % (±0.09)	85.2 %	0.02×10^6	3.8 ms
Trajectory LSTM	79.6 % (±0.08)	87.2 %	0.2×10^6	5.9 ms

Table 3.7 Comparison of the accuracy, AUC, number of parameters and average inference time per prediction for both the state-based and the proposed trajectory-based approach.

Next, the failure prediction performance over time is analyzed. The average accuracy of the trajectory-based approach and the state-based approach for each time step within both failure and success test sequences are shown in Figure 3.20. For reference, we also include the best-performing baseline approach based on manual features, *Curve+Length LSTM*.

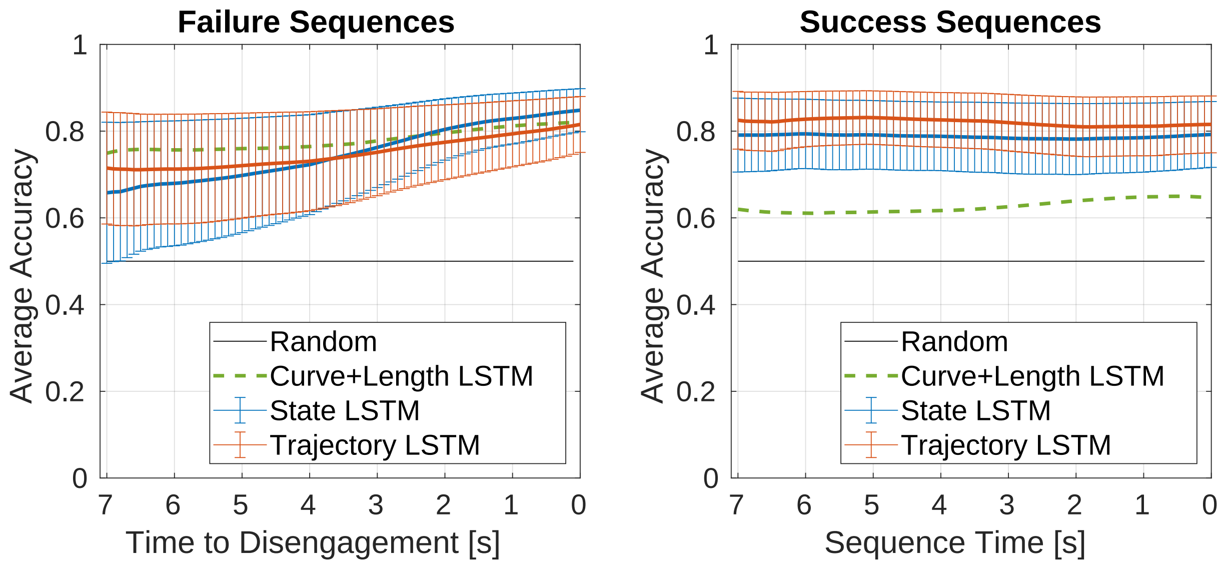


Figure 3.20 Average accuracy over time for both failure and success sequences of the proposed *Trajectory LSTM* approach compared to the best-performing baseline approaches.

For the success sequences, the trajectory-based approach consistently outperforms the state-based approach as well as the model based on manual features. The accuracy of all compared approaches is mostly constant. If no disrupting events that cause disengagements occur, the planned trajectories and the actually driven trajectories are largely identical. The state-based and the trajectory-based approach are therefore expected to perform similarly. The fact that the trajectory-based approach performs better could be explained by the fact that the planned trajectories are less prone to noise. A planned trajectory describes the best-case scenario as generated by the vehicle. In contrast, the actual car state can exhibit minor variations due to physical perturbations such as uneven roads which can cause false positives.

For the failure sequences, the accuracy of all approaches increases when approaching the time of the disengagement. The *Trajectory LSTM* outperforms the *State LSTM* during the early stages of the failure sequences, when the disengagement is still at least three seconds away. During the final three seconds before the failure, the state-based approach slightly outperforms the trajectory-based method. At this time point, the disagreements detectable in the planned trajectories four to seven seconds in advance are materializing in the physical state of the car. This result demonstrates the different strengths of using state or trajectory data. Planned trajectories are the most useful when attempting very early failure prediction, since they contain information about the planned future behavior of the vehicle. In the seconds right before the disengagement, the state of the car becomes more informative, since it shows the actual response of the system to the challenging situation regardless of the planned ideal maneuver.

3.4.4.4 Failure Cases

Finally, we discuss the failure cases of the trajectory-based failure prediction approach. To allow some intuitive explanations of where the proposed method fails, the average speed and average angle of the failure cases in both success and failure sequences are shown in Figure 3.21.

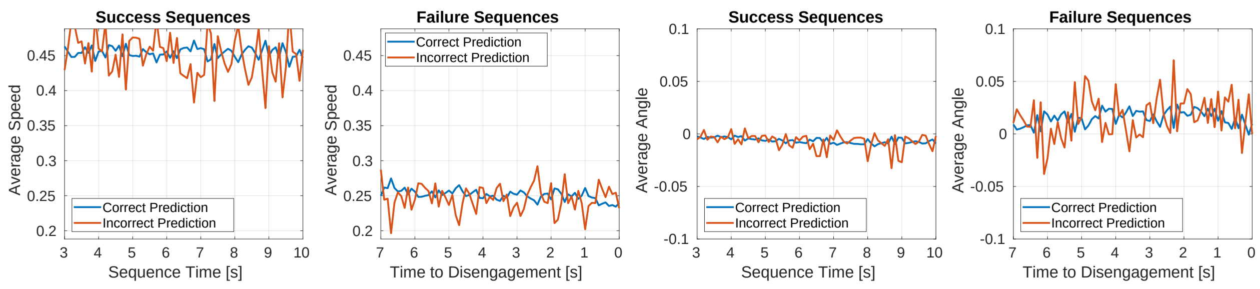


Figure 3.21 Visualization of the average speed and average angle of failure cases as well as correct predictions of the proposed trajectory-based failure prediction approach.

The mean of the average speed of incorrect predictions is almost identical to the mean of the average speed of correct predictions. This applies both to success and failure sequences. The same can be observed when plotting the average angle. This is in contrast to the state-based and image-based approaches, where the average speed or angle of the respective fail urecases differed significantly from correct predictions.

The most noticeable difference between correct and incorrect predictions is that the average speed and angle of incorrect predictions vary visibly around the mean, while they are more constant for correct predictions. It is challenging to infer how the corresponding planned trajectories were interpreted falsely by the proposed model. The noisy speed and angle plots of incorrect predictions suggest that failure cases of the proposed approach are not visibly correlated to the physical state of the car. Considering that the failure cases of the state-based and image-based approach were more closely connected to the speed and angle, this observation indicates that the third approach of using trajectories as input again has different weaknesses. All three failure prediction approaches have shown different failure case behavior, which motivates a combination of the different methods to alleviate the individual weaknesses and combine their strengths. Therefore, we introduce a fusion-based introspective failure prediction approach next.

3.5 Introspective Failure Prediction Using Late Fusion

As our final contribution to the topic of system-level failure prediction, we propose an introspective failure prediction approach based on fusing the separate approaches introduced in the previous three sections. Since the evaluations of the state-based, image-based, and trajectory-based approaches have all shown different strengths and weaknesses, the goal is to combine the benefits of all three sources of information into one model. Next, we generally discuss whether early or late fusion is more appropriate for this use case. Then, we present our late multimodal fusion approach for failure prediction. Finally, we evaluate the fusion-based approach and compare its performance to the individual methods from the previous sections.

3.5.1 Early Fusion vs. Late Fusion

Fusion of different data types and inputs can be performed either early or late in a neural network architecture. While some works suggest that the level at which fusion occurs does not have a significant impact on performance [62], there are several factors to consider when choosing the fusion level. Here, the data input types are state vectors, RGB images and trajectory vectors. For fusing these data types, both early and late fusion can be considered. We discuss both options in the context of system-level failure prediction next.

Regardless of performance, a key advantage of early fusion is that the overall computational complexity is lower. As soon as sensory inputs or low-level features extracted from them have been fused, only one joint architecture needs to be evaluated when making an inference. In driving, inference speed is an important aspect. However, the previous sections have shown that the state-based and the trajectory-based approach require only milliseconds for an inference. Running those models sequentially is still fast enough to achieve real-time performance.

Another potential advantage of early fusion is that the failure prediction model can learn patterns across multiple input modalities. The more layers have access to all input types, the more complex such cross-modal patterns can be. In late fusion, each input type is used separately to make a prediction. A late fusion model therefore cannot make use of patterns that are extracted from more than one modality.

While early fusion is a possibility, late fusion has several properties that are highly beneficial for the proposed disengagement prediction framework. First, the property of early fusion models to be able to learn cross-modal patterns can also be detrimental. If one modality is more informative than the others, the model can learn to focus only on the best input type. In the previous sections, the image-based classifier achieved the best accuracy by a wide margin. Early fusion of the image input with the low-dimensional data can lead to the network ignoring the low-dimensional data since during training, an almost perfect accuracy can be achieved using the image data alone. Additionally, even when cross-modal patterns are learned, some works suggest that learning co-dependencies between weights in early layers can reduce the performance of a model [119]. If the input types are kept separate for as long as possible, each part of the model needs to learn patterns that are unique to its data type. The fact that the failure cases of each model investigated so far showed noticeably different behaviors suggests that each model has learned distinct patterns for its respective failure prediction. To be able to detect as many different failures as possible, the capacity to detect a wide range of different patterns is desirable.

Regarding the implementation of the fusion network, the modalities of high-dimensional RGB images and low-dimensional state and trajectory data require significantly different architectures to extract patterns. For images, convolutional architectures are required, whereas LSTM-based models are well suited for the low-dimensional inputs. Since the best-performing image-based architecture, *ResNet50*, consists of convolutional layers until the last layer, early fusion with a non-spatial input such as state vectors is inherently challenging.

Another benefit of late fusion is that the resulting failure prediction framework remains modular and easily extendable. In this work, three information sources are used to predict failures. Other sources could be used as well, such as Light Detection and Ranging (LIDAR) input or internal representations of the system such as the output of individual perception models. Early fusion would require redesigning and retraining the entire framework whenever a modality is changed, whereas late fusion allows for fast testing of new input types without requiring to change previously trained models.

A related useful property of keeping the framework modular is that it allows more intuitive insights into what caused the predicted system-level failure. If all input types are fused early, it is challenging to infer what made the introspective model generate its prediction. If individual failure probabilities are derived from each modality and then fused, it is straightforward to see which input type had the largest impact on the current prediction. Knowing whether the planned future behavior or the current visual environment caused a disengagement could be valuable information both for the safety driver at the wheel and for a remote operator who needs to quickly assess what caused the failure.

For these reasons, we use a late fusion approach to combine the different input types investigated in the previous sections. Next, the resulting late multimodal fusion approach is presented in detail.

3.5.2 Late Multimodal Fusion Approach

In the previous sections, we have introduced three separate failure prediction models: a state-based classifier C_{state}, an image-based classifier C_{img}, and a trajectory-based classifier C_{traj}. The experimental results in Section 3.2, Section 3.3, and Section 3.4 have shown that all three models outperform the state of the art of system-level failure prediction [15, 16] significantly. In this section, the three models are combined in a late fusion approach to further improve failure prediction performance. An overview of the proposed late multimodal fusion framework is shown in Figure 3.22.

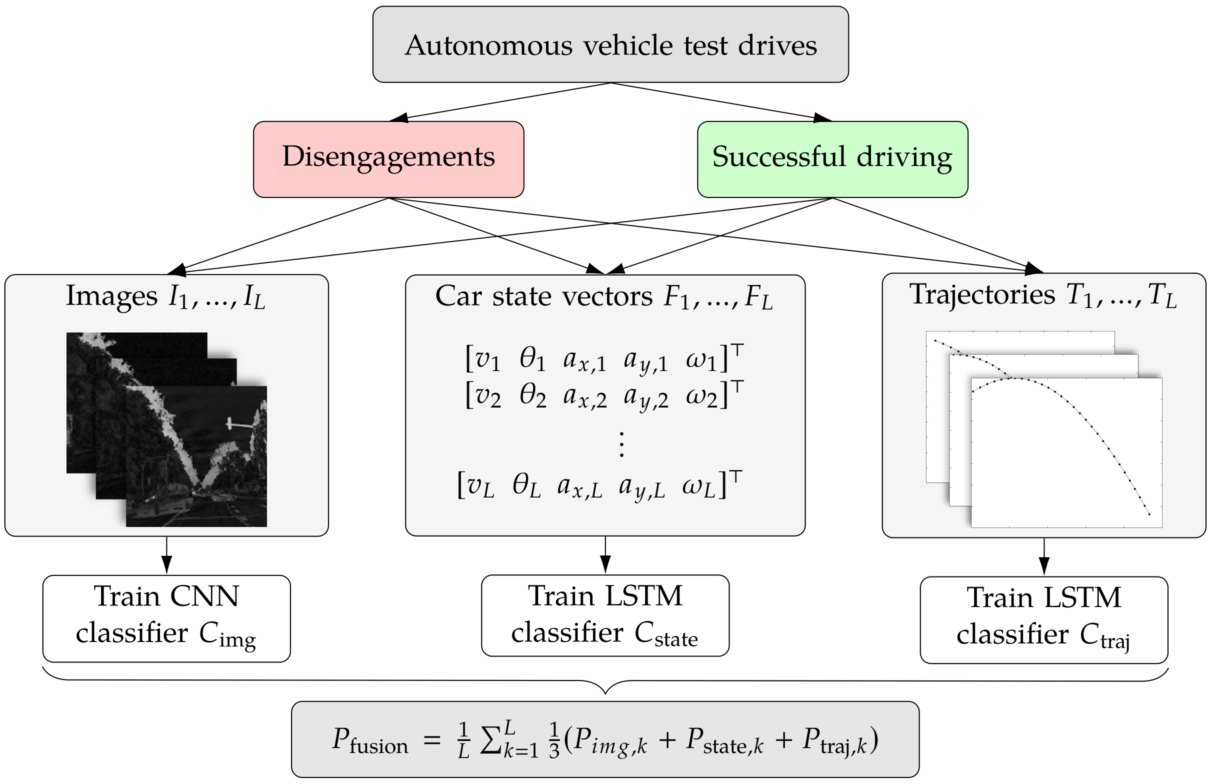

Autonomous vehicle test drives
Disengagements
Successful driving
Images $I_1, ..., I_L$
Car state vectors $F_1, ..., F_L$
$[v_1 \ \theta_1 \ a_{x,1} \ a_{y,1} \ \omega_1]^\top$
$[v_2 \ \theta_2 \ a_{x,2} \ a_{y,2} \ \omega_2]^\top$
$\vdots$
$[v_L \ \theta_L \ a_{x,L} \ a_{y,L} \ \omega_L]^\top$
Trajectories $T_1, ..., T_L$
Train CNN classifier C_{img}
Train LSTM classifier C_{state}
Train LSTM classifier C_{traj}
$P_{\text{fusion}} = \frac{1}{L} \sum_{k=1}^{L} \frac{1}{3}(P_{img,k} + P_{\text{state},k} + P_{\text{traj},k})$

Figure 3.22 Overview of the proposed late multimodal fusion framework. The state-based classifier C_{state}, the image-based classifier C_{img}, and the trajectory-based classifier C_{state} are used to obtain the individual failure probabilites P_{img}, P_{state}, and P_{traj}, which are then averaged to obtain the fused failure probability P_{fusion}.

The three classifiers are trained as before with all available disengagements and an equal amount of sampled successful driving sequences. The input to the fusion-based model then consists of the current dynamic difference image $I_{\mathrm{dd},t}$ consisting of three difference images created from four of the last L images, the last L state vectors F, and the last L planned trajectories. We again use $L = 30$.

Since late fusion is already costly due to requiring the inference of three separate models, the fusion step itself to combine the three classifiers is kept simple. From each classifier, the respective failure probability for the current input is obtained. Then, the resulting three failure probabilities P_{img}, P_{state}, and P_{traj} are averaged with equal weights. Preliminary experiments with using an SVM for determining the weights of the individual probabilities did not significantly increase the accuracy. The fused failure probability P_{fusion} is then calculated as

$$P_{\mathrm{fusion}} = \sum_{k=1}^{L} \frac{1}{3}(P_{img,k} + P_{state,k} + P_{traj,k}). \tag{3.17}$$

Simply averaging the individual failure probabilities does not introduce any further parameters that need to be tuned. If additional modalities are used to train additional classifiers, they can be easily added to the framework by adding their predicted failure probability to the average in Equation 3.17. The individual failure probabilities could be weighted according to the performance of the individual models, but preliminary results on the validation set showed that this does not improve accuracy. Giving each model the same influence on the final prediction combines the strengths of all models. It allows the models with overall lower performance such as the state-based classifier to mitigate the errors of the better performing models such as the image-based classifier. Weaker models improving the performance of a stronger model by averaging their predictions is an effect also observed in ensemble-based models [12].

Finally, the fused failure probabilities are smoothed using a moving average filter with horizon $H = L$ as before. A failure is then only predicted if more than one model consistently predicts a high failure probability over multiple time steps. The final filtered fused prediction P_{fusion} is therefore calculated as

$$P_{\mathrm{fusion},t} = \frac{1}{\min(t,H)} \sum_{k=1}^{\min(t,H)} P_{\mathrm{fusion},t+1-k}. \tag{3.18}$$

3.5.3 Results

Next, the proposed multimodal fusion approach is evaluated. All classifiers were trained using the disengagement and success data from Table 3.6. The failure prediction performance is analyzed using both an ROC curve analysis and by plotting the average accuracy over time. The computational complexity is also discussed, since the increased inference time is one of the main theoretical downsides of late fusion. To analyze if all three classifiers are needed for the best fusion performance, all combinations of the three individual models are evaluated this way. Finally, the failure cases of the fusion-based model are discussed.

3.5.3.1 Failure Prediction Performance

First, all models are compared using their ROC curves. Besides the individual models *State LSTM*, *Trajectory LSTM*, and *ResNet50*, all possible fusion combinations are considered. The combination of all three classifiers is referred to as *State+Trajectory+Image*. Additionally, the three combinations of using only two classifiers at a time for the late fusion are also evaluated. They are referred to as *State+Trajectory*, *State+Image*, and *Trajectory+Image*. For those three approaches, Equation 3.17 is modified to be the average of only the two corresponding failure probabilities. The ROC curves of the three individual models, the three reference fusion approaches consisting of two individual models, and the fusion of all three individual models are shown in Figure 3.23.

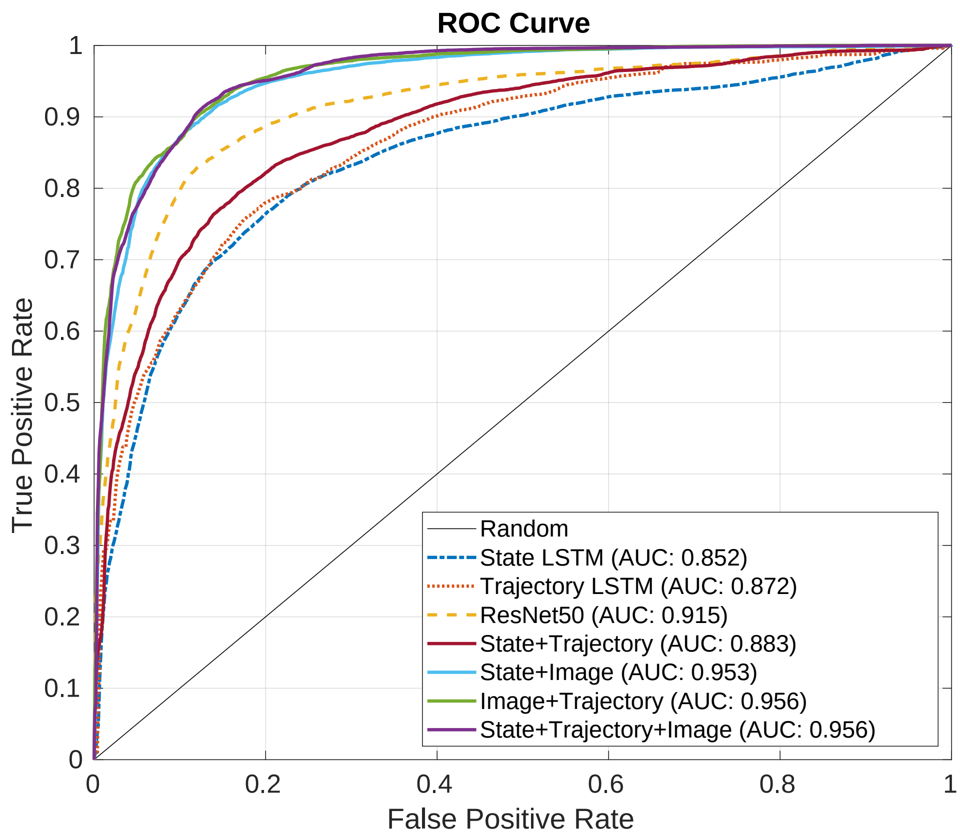


Figure 3.23 Comparison of the ROC curves of all individual introspective failure prediction models as well as all possible late fusion combinations.

Every individual model is improved by adding one or more other modalities via late fusion. The AUC of the *State+Trajectory* fusion outperforms the AUC of the *State LSTM* by 3.6 % and the AUC of the *Trajectory LSTM* by 1.3 %. Adding either the state-based or the trajectory-based classifier to the image-based classifier results in the most significant improvement, increasing the AUC of the *ResNet50* by at least 4.1 %.

For a comprehensive comparison, we also summarize the accuracy plus standard deviation, the AUC, the number of parameters, and the average inference time per prediction of all approaches in Table 3.8.

Model	**Accuracy**	**AUC**	**Parameters**	**Time**
State-based	78.8 % (±0.09)	85.2 %	0.02×10^6	3.8 ms
Trajectory-based	79.6 % (±0.08)	87.2 %	0.2×10^6	5.9 ms
Image-based	85.1 % (±0.05)	91.5 %	25.6×10^6	59.5 ms
State+Trajectory Fusion	81.1 % (±0.07)	88.3 %	0.2×10^6	9.7 ms
State+Image Fusion	88.2 % (±0.04)	95.3 %	25.6×10^6	63.3 ms
Image+Trajectory Fusion	88.8 % (±0.03)	95.6 %	25.8×10^6	65.4 ms
State+Trajectory+Image Fusion	89.1 % (±0.03)	95.6 %	25.8×10^6	69.2 ms

Table 3.8 Comparison of the average accuracy, AUC, number of parameters, and average inference time of the proposed individual and fusion-based introspective failure prediction methods.

The average accuracy of all models increases in a similar way as the AUC. The fusion of all three individual classifiers, *State+Trajectory+Image*, achieves the highest average accuracy at 89.1 %. While the accuracy is higher, the AUC of fusing all three models is identical to the fusion of the image-based and the trajectory-based model. The standard deviation of the *ResNet50* decreases by another 40 % by adding the trajectory-based classifier. While the accuracies of the *State LSTM* and the *Trajectory LSTM* are at least 6.9 % smaller than the accuracy of the *ResNet50*, the fusion of all three improves the *ResNet50* by 4.7 %. These results demonstrate that the late fusion approach is capable of combining the strengths of the individual models.

While the proposed late fusion improves the failure prediction performance, it also increases the computational complexity and average inference time. The inference time of the best-performing *State+Trajectory+Image* fusion approach is 16 % larger than the slowest individual approach of *ResNet50*. Due to the very compact nature of the proposed LSTM-based models, the number of parameters increases only by around 1 %. While the fusion-based approach is significantly slower, the maximum average inference time of 69.2 ms is still far below 100 ms, the maximum time allowed to achieve the desired prediction frequency of 10 Hz. It is also still faster than the best-performing state-of-the-art method *Predictive Variance* [16] evaluated in Section 3.3, which has an inference time of 81.4 ms.

Finally, the average accuracy over time is evaluated. Since the fusion of all three individual models performed best, we do not further consider the fusion of two individual models in the following. In Figure 3.24, the average accuracy of the individual models and the fusion-based approach is shown for both failure and success sequences.

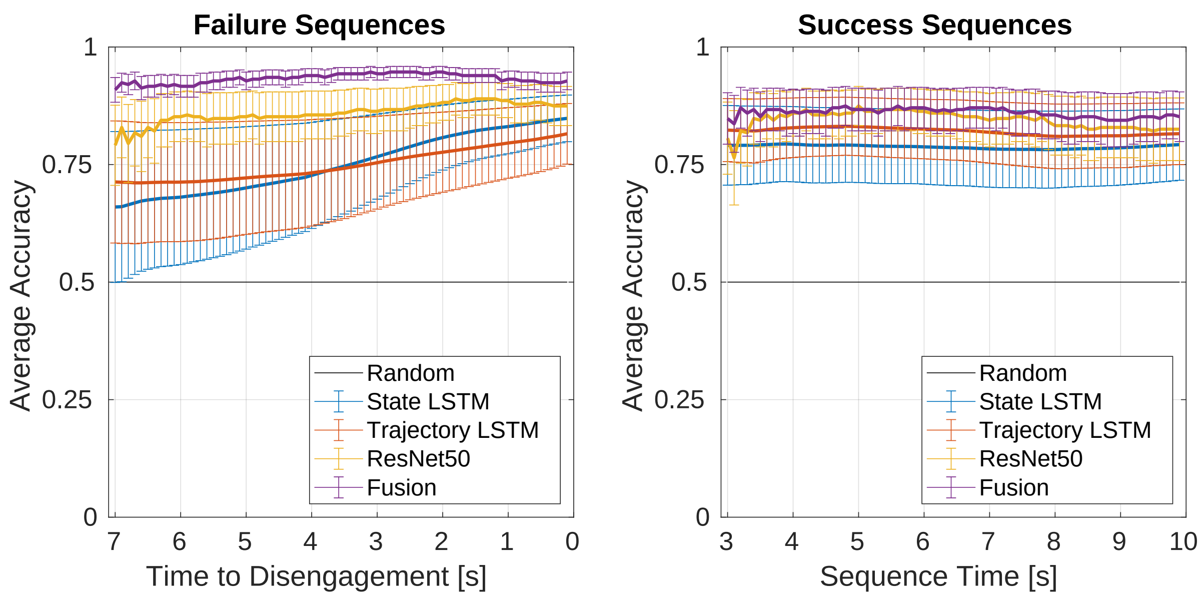


Figure 3.24 Average accuracy over time of the individual failure prediction models compared to late fusion.

The fusion-based approach outperforms all individual models for all time points in both failure and success sequences. In success sequences, the difference to the individual models is mostly constant. The largest difference can be observed in the failure sequences. While the models show more similar accuracies in the time right before a disengagement, the accuracy of the fusion-based approach remains high throughout the entire sequence. This indicates that the combination of the current visual features, the planned next three seconds of trajectories and the current physical state of the car is sufficient to consistently predict failures at an accuracy of almost 90 %, even seven seconds in advance of a failure.

3.5.3.2 Failure Cases

While the fusion-based approach correctly predicts almost 90 % of all failures, it still misses around 10 % of all disengagements. Same as for the individual approaches, we visualize the average speed and angle of the car both during correct and incorrect predictions in Figure 3.25. Since the fusion-based approach combines three significantly different input types, it is challenging to infer what causes those failure cases. No clear patterns are discernible among the incorrect predictions, neither for false positives nor for false negatives.

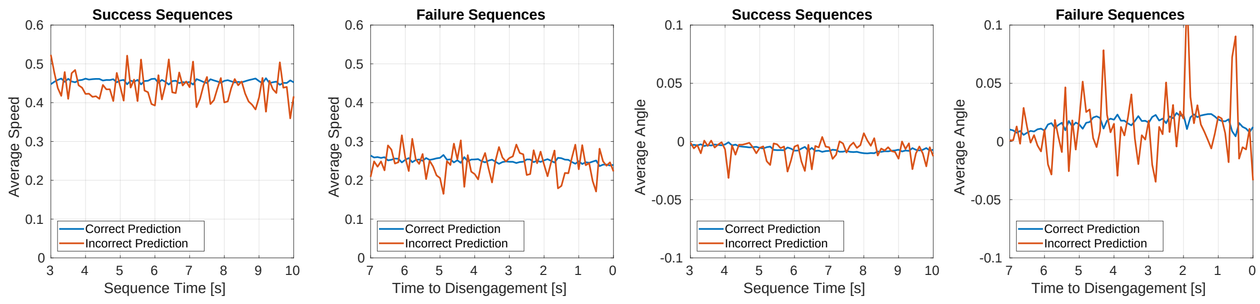


Figure 3.25 Visualization of the average speed and average angle of the vehicle over time for both correct and incorrect failure predictions made by the fusion-based approach.

While the lack of interpretability of the failure cases of the fusion-based approach makes it challenging to improve the model, it should be noted that it is unlikely for system-level failure prediction methods to be able to predict *all* failures in advance. While accuracies beyond 90 % could be possible, some spontaneous situations will likely remain unpredictable for the proposed introspective failure prediction approach. Some failures are caused by effectively random events, such as pedestrians suddenly appearing at the side of the road or crossing the road. In such cases, the environment can only give some general indication about the likelihood of such an event, for example if it is an intersection where pedestrians are to be expected. The specific moment in the scene that causes the system to disengage or that causes the human safety driver to take over control can be entirely unpredictable. The randomness introduced by other dynamic traffic participants is therefore a challenge inherent to the task of system-level failure prediction.

3.6 Chapter Summary

In this chapter, we introduced an introspective failure prediction framework for predicting system-level failures of autonomous systems. In the context of driving, we defined system-level failures as disengagements of the system, triggered either automatically by the system itself or manually by the human safety driver taking over control. The proposed concept is based on recording failures of the inspected system as well as successful behavior, to then train a classifier to learn how to distinguish between failure and success.

As a first implementation, a state-based approach was presented in Section 3.2. By training an LSTM-based classifier with sequences of state data from both success and failure sequences, failures could be predicted at an accuracy of over 78 %. Seven seconds in advance of a failure, the model was still capable of detecting over 75 % of all failures.

In Section 3.3, we improved this concept by considering images as a source of information about failures. To extract spatio-temporal patterns from sequences of images, the image sequences were processed into compact dynamic difference representations that were then used to train a CNN-based classifier. Representing image sequences as a single three-channel image allowed reusing the state-of-the-art *ResNet50* architecture pretrained on ImageNet. This approach improved the failure prediction accuracy to over 83 %, outperforming state-of-the-art image-based failure prediction methods by 32 %.

As a third approach, the trajectories planned by the car were used in Section 3.4. The disagreement of the car over time with itself is a source of information about failures that already contains a prediction about the future, making it a suitable choice for early failure prediction. Using planned trajectories instead of the actual state of the car improved early failure prediction by 2 % in an ROC analysis.

Finally, all three individual failure prediction models were combined in a late fusion model in Section 3.5. By averaging the individual failure probabilities, the overall prediction accuracy was improved significantly to 89.1 %. At less than 70 ms per prediction, the fusion model is capable of running in real time and is more than 17 % faster than previous state-of-the-art approaches. By combining the advantages of the individual approaches, the fusion-based introspective failure prediction model is capable of predicting failures at over 86 % accuracy seven seconds in advance.

4 Component-Level Failure Prediction

In this chapter, we present several approaches for detecting and predicting the failures of individual components of the perception module of an autonomous system. Specifically, we focus on semantic segmentation as one of the most critical components of the perception stack [170]. While failures of individual components do not necessarily lead to system-level failures that require human intervention, detecting them is an important component of a comprehensive failure prediction framework. Predicted failures from individual components can be aggregated and ultimately also used to predict system-level failures. Additionally, the proposed component-level failure prediction will be the basis for some of the failure correction strategies introduced in Chapter 5.

In the following, we first propose an introspective approach for predicting failures for semantic image segmentation. Then, we extend this idea to failure prediction for semantic video segmentation. Finally, we propose a method for predicting pixel-wise segmentation failures of future frames.

Some of the concepts and contributions of this chapter have been published in [22, 27].

4.1 Introspective Failure Prediction for Semantic Image Segmentation

In Chapter 3, we have used the concept of introspection to predict disengagements of an entire autonomous system. While the task of semantic image segmentation is significantly different, the same fundamental concept of learning from previous failures can be applied. For this, we propose to record pixel-wise failures of a given semantic segmentation model and then train an introspective model to predict for each pixel whether it will be correctly classified or not. Next, we introduce the concept of introspective failure prediction for semantic image segmentation in more detail. We first present the baseline semantic segmentation model for which we then learn to predict failures. Then, we summarize the data generation and training process of the introspective model.

4.1.1 Concept

The task of semantic image segmentation consists of training a model S_{semseg} with images I to predict the corresponding pixel-wise class labels L. The class labels can be any of N semantic classes. Semantic segmentation therefore corresponds to classifying each pixel in an image at once. While the state of the art in semantic image segmentation can achieve accuracies of beyond 90 % [171] on challenging data sets such as A2D2 [86] or Cityscapes [84], such models still inevitably make mistakes.

Those mistakes can be described as binary error maps E_{GT} obtained as the pixel-wise difference between the semantic prediction L_{pred} generated by the model S_{semseg} and the ground-truth labels L_{GT}. If the prediction and the ground-truth label are identical, the corresponding error map value is a 0, otherwise it is a 1. Now, the concept of introspection can be applied. The recorded error maps are used as pixel-wise ground-truth error labels E_{GT} alongside the corresponding image I to train an introspective failure prediction model S_{intro}. For each pixel, the goal is to predict whether the pixel was correctly classified or not. This is a binary classification task. Since the failure prediction model needs to assign each pixel of an image I a class label, the introspective failure prediction model S_{intro} also has to perform the task of semantic image segmentation. Thus, any semantic segmentation architecture can be used to implement S_{intro}. Regardless of the specific choice of the implementation of the baseline semantic segmentation model S_{semseg}, the semantic segmentation architecture of S_{semseg} is already available and can be reused for introspection. This allows for minimizing the amount of implementation effort.

The concept of introspective failure prediction can be summarized as follows. Given images I_{semseg} with pixel-wise annotations $L_{semseg,GT}$, a model S_{semseg} is trained to generate predicted pixel-wise class labels $L_{semseg,pred}$. After S_{semseg} is trained, its predictions $L_{intro,pred}$ are obtained for an image data set I_{intro}. By comparing the predictions $L_{intro,pred}$ to the actual labels $L_{intro,GT}$, the ground-truth error maps $E_{intro,GT}$ are obtained. Then, the introspective model S_{intro} is trained with the images I_{intro} as input and the error maps $E_{intro,GT}$ as targets. After training, the introspective model S_{intro} can be directly applied to a new test image I_{test} to obtain the predicted errors $E_{test,pred}$. The performance of S_{intro} can then be evaluated by comparing the predicted error map $E_{test,pred}$ to the actual error maps $E_{test,GT}$, which are obtained by comparing the labels $L_{test,GT}$ to the semantic prediction $L_{test,pred}$ obtained from feeding I_{test} into S_{semseg}. The process is visualized in Figure 4.1.

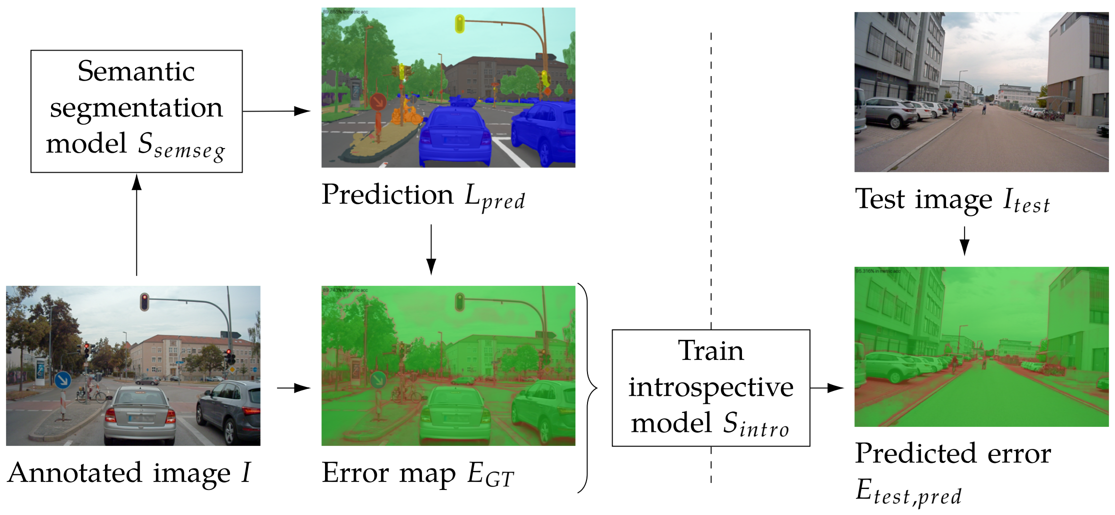

Semantic segmentation model S_{semseg}
Prediction L_{pred}
Test image I_{test}
Annotated image I
Error map E_{GT}
Train introspective model S_{intro}
Predicted error $E_{test,pred}$

Figure 4.1 Overview of the proposed workflow to train an introspective failure prediction model S_{intro} that predicts pixel-wise errors made by a baseline semantic segmentation model S_{semseg} (adapted from [22] © 2020 IEEE).

For simplicity, the same data set is used for the training of both S_{semseg} and of S_{intro} in Figure 4.1. In that case, the data sets D_{semseg} and D_{intro} are identical, but they do not have to be. In the following section, the design of the models S_{semseg} and of S_{intro} as well as the generation of the data sets D_{semseg} to train the segmentation and D_{intro} to train the introspective model are discussed in more detail.

4.1.2 Model Design

Next, we discuss the implementation of the individual models for the workflow outlined in Figure 4.1 in more detail. We first select a state-of-the-art semantic segmentation architecture as a baseline model. Then, we use the baseline model to generate failure data and then discuss the design of the introspective model. Both a black box approach that does not require any information about the baseline model and a method where the introspective model shares information with the baseline model are proposed.

4.1.2.1 Baseline Model

The proposed concept of introspective failure prediction can be applied to any semantic segmentation model. The semantic segmentation model can be treated as a black box, with only its pixel-wise predictions and the corresponding input images being needed to train the introspective model. However, having access to the inner layers of the model allows sharing intermediate representations of the input between the baseline model S_{semseg} and the introspective model S_{intro}. This allows S_{intro} to be better informed about how S_{semseg} reached its partially incorrect prediction.

For this work, we use an established state-of-the-art deep learning architecture as the implementation of S_{semseg}. The DeepLabV3+ model [72] has shown competitive performance on real-world driving data sets such as Cityscapes [84]. We select this architecture as the baseline for all further experiments. The encoder-decoder structure of DeepLabV3+ is shown in more detail in Figure 2.5 in Chapter 2. In practical applications, whichever model is already implemented and available in the autonomous system could be used as the baseline model in the proposed workflow.

4.1.2.2 Failure Data Generation

To generate error maps with a baseline model S_{semseg} trained on an image data set D_{semseg}, a corpus of images D_{intro} is required for which the error maps $E_{GT,intro}$ are then obtained. If a sufficiently large amount of labeled images is available, the images used to train S_{semseg} and the images used to train S_{intro} can be mutually exclusive. For example, the original training set from a given data set could be split up into two equally sized subsets, one to train S_{semseg} and one to train S_{intro}. The testing data set D_{test} can be used to test both models. The predictions $L_{test,pred}$ made by S_{semseg} on D_{test} can be compared to the ground-truth labels $L_{test,GT}$ to evaluate the performance of S_{semseg}, while also being used as ground-truth error labels $E_{test,GT}$ to compare to the error maps $E_{test,pred}$ predicted by S_{intro}. The approach to split up the original training set into two equally sized subsets is visualized in Figure 4.2.

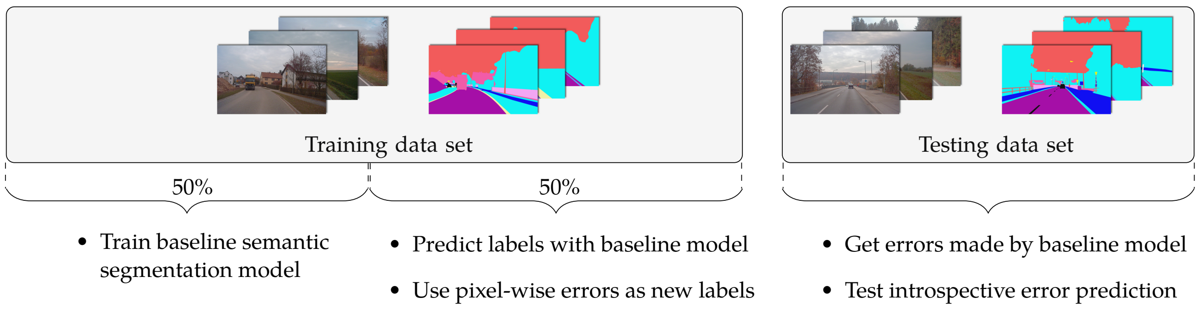

Training data set
Testing data set
50%
50%
• Train baseline semantic segmentation model
• Predict labels with baseline model
• Use pixel-wise errors as new labels
• Get errors made by baseline model
• Test introspective error prediction

Figure 4.2 Visualization of the generation of training and test data for a baseline semantic segmentation model S_{semseg} and an introspective failure prediction model S_{intro} from one large-scale image data set with pixel-wise labels (adopted from [22] © 2020 IEEE).

The data sets D_{semseg} and D_{intro} do not have to be equally distributed. Other ratios can be used depending on whether the baseline segmentation model or the failure prediction model is expected to be more challenging to train. If D_{intro} is different from D_{semseg}, the errors E_{intro} are made on images previously not seen by the baseline model S_{semseg}. Then, the introspective model S_{intro} effectively learns from test errors. While this can lead to a better failure prediction performance, it comes at the cost of harming the performance of S_{semseg}, since it has to learn from fewer training images.

Another important option is therefore to not keep the two sets exclusive. If the original data set is too small to be split into two subsets of sufficient sizes, the data sets D_{semseg} and D_{intro} can also be chosen to be identical. Then, they both comprise the entire available set of training images. In that case, the error $E_{intro,GT}$ made by S_{semseg} on the images is used to train S_{semseg}. Then, $E_{intro,GT}$ is the training error, which is typically lower than the testing error if the model has overfitted to the training set. The introspective model S_{intro} then learns what visual patterns have led to errors on the training set. If the testing error is larger than the training error, S_{intro} is trained with a different error distribution than it will encounter during testing. However, it can be argued that the errors made on the training set are errors that are most likely to also be made on the testing set. If repeatedly seeing an image during training was not sufficient for the baseline model S_{semseg} to correctly segment it, it will likely make those mistakes again on a test image with similar visual patterns. For this reason, reusing D_{semseg} to train S_{intro} still allows for predicting a significant amount of failures and is done in Section 4.2.

4.1.2.3 Introspective Failure Prediction Model

Once a data set D_{intro} and corresponding pixel-wise errors $E_{intro,GT}$ are obtained, the introspective failure prediction model can be trained. The task of assigning each pixel of an input image one of two classes, success or failure, can again be approached using semantic image segmentation. Since the accuracy of the baseline model S_{semseg} is typically beyond 90 %, those two classes are inherently imbalanced. While failures are rarer than successes, strategies such as frequency weighting of the loss function can be used to address the imbalance.

Any semantic segmentation architecture can be used to implement the introspective failure prediction model S_{intro}. We use the same architecture used for S_{semseg}. Besides avoiding additional implementation effort, reusing the same architecture allows for sharing the weights of the first layers can be shared among the baseline and the introspective model. This reduces the training time since the low level features learned by the first convolutional layers do not have to be learned from scratch. During testing, it increases the inference speed since the first layers need to be evaluated only once. Additionally, weight sharing gives the introspective model access to the features used by the baseline model for its prediction. If it is supplied the low level features extracted by the baseline model, the failure prediction model can directly operate in feature space and learn to detect which low level features lead to failures. Without weight sharing, the introspective model needs to learn extracting visual features that lead to classification errors from the raw input image itself. In a preliminary Receiver Operating Characteristic (ROC) curve analysis, weight sharing improved the Area Under Curve (AUC) by 4.8 % and is therefore used during all subsequent experiments.

We use the DeepLabV3+ architecture in all our experiments, which consists of an encoder and a decoder [171]. For the introspective model, we reuse the weights of the encoder and only train the decoder with the error maps E_{GT} as target labels. The introspective model S_{intro} can then be seen as adding a second decoder to the network. The encoder of DeepLabV3+ consists of a backbone network and an atrous spatial pyramid pooling module. The output of the backbone and of the feature pyramid is then fed into the baseline decoder to obtain S_{semseg} and fed into the introspective decoder to obtain S_{intro}. The softmax score associated to the failure class is used as a failure probability for each pixel. The resulting architecture is shown in Figure 4.3.

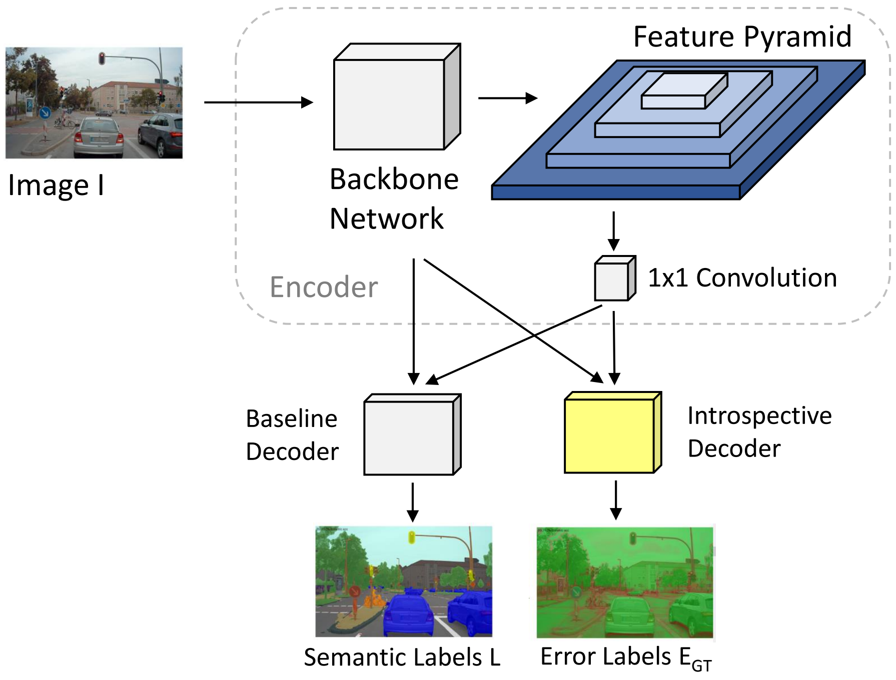


Figure 4.3 Overview of the proposed approach of implementing the introspective failure prediction model as a second decoder, sharing the trained encoder with the baseline DeepLabV3+ semantic segmentation model.

If the encoder of the baseline model is not available, the introspective model can also be trained entirely separately from S_{semseg} by also training the encoder from scratch. This allows for treating the baseline model as a black box. In practice, this can be desirable, for example if the semantic segmentation is supplied by a third party contractor and no direct access to the internal layers is available. In preliminary experiments, training S_{intro} from scratch consistently led to worse performance. Whenever possible, we therefore only use the approach where the encoder is shared between S_{semseg} and S_{intro} in the following.

4.1.3 Results

In this section, we present the results of the experiments conducted to evaluate the proposed introspective failure prediction approach for semantic image segmentation. We implement two reference state-of-the-art uncertainty estimation approaches for comparison. Then, we evaluate the performance of all models using a precision-recall analysis. Finally, we visualize the results.

4.1.3.1 Baseline Semantic Segmentation

First, we train a baseline semantic segmentation model S_{semseg} on a large-scale public data set of driving images with pixel-wise annotations. We select the A2D2 data set [86] for this purpose. The A2D2 data set includes over 40 000 driving images collected in German cities. Each image is manually annotated with pixel-wise labels from a total of 38 semantic classes. This makes it one of the largest public driving data sets available. The large number of annotated images allows splitting the training set from A2D2 into two equally sized subsets. A total of around 16 000 images is used to train the baseline model S_{semseg}, with another 16 000 images being reserved as training data for the introspective model S_{intro}. For validation of both the baseline and the introspective model, 4000 images are used. The final 4000 images constitute the testing set.

Using the first 16 000 training images, we trained a DeepLabV3+ model for a total of 135 epochs until the validation loss did not further decrease. For all hyperparameters, the official suggested training procedure from the authors of DeepLabV3+ [171] was followed. We used a public state-of-the-art implementation [172], reaching a test accuracy of 94.0 %.

4.1.3.2 Reference Approaches

We implement two reference approaches to compare the proposed approach to the state of the art. To the best of our knowledge, no explicit pixel-wise failure prediction approach for semantic segmentation is available. However, uncertainty estimation methods can be used for the same purpose if the predicted uncertainty is interpreted as a failure probability. We therefore use the state-of-the-art Monte Carlo (MC) dropout approach [10] and the Deep Ensemble approach [12] as two reference methods. MC dropout has been used to predict uncertain pixels for semantic image segmentation before [120], achieving state-of-the-art performance for that task. Since Deep Ensembles have been shown to outperform MC dropout in other tasks [12], we use them as the second state-of-the-art comparison.

For both reference approaches, the implementation suggestions from the original papers are followed. MC dropout is implemented by keeping the dropout layers already present in DeepLabV3+ active during testing and obtaining ten forward passes with different random dropout masks for each input image. For Deep Ensemble, four additional baseline models are trained with different random initial weights, yielding a total of five predictions per pixel. For MC dropout, the variance of the ten predictions obtained from the ten forward passes is used as a failure probability. For Deep Ensemble, the variance of the five predictions from the ensemble of five models is used.

4.1.3.3 Failure Prediction Performance

The trained baseline model S_{semseg} is used to predict the semantic map for each of the remaining 16 000 training images. The error maps obtained from those predictions are used to train the introspective model S_{intro}. The training loss for each pixel is inversely weighted with the frequency of the class of each pixel. Since less than 10 % of all pixels are failures, the loss for failure pixels is weighted with a factor of 10, while the loss for success pixels is not changed. Due to the reduced complexity of the binary classification task of failure prediction and the frozen encoder layers, the introspective model converged after 67 epochs.

In all subsequent experiments, the softmax score associated to the failure class is used as the failure probability for the introspective approach, while the variance is used as the failure probability for MC dropout and Deep Ensemble. Due to the class imbalance of failures and successes, we analyze the precision-recall curve of each method for the test set. First, we evaluate the average image-wise precision-recall curve. Then, we perform a precision-recall analysis for each individual class to investigate how well the proposed approach performs for the most safety-critical classes.

Image-Wise Evaluation For each of the 4000 test images, we calculate the precision-recall curve for each method. The curve for each image is calculated from the predicted failure probability of all pixels of the current image. The average precision-recall curve for each method is shown in Figure 4.4. With a baseline random classifier, the precision simply equals the percentage of positive test samples for all recall values. We also evaluate combinations of introspection with the two reference approaches of MC dropout and Deep Ensemble. This is done by averaging the predicted failure probability of the two combined approaches and using the resulting average as the score for the precision-recall analysis.

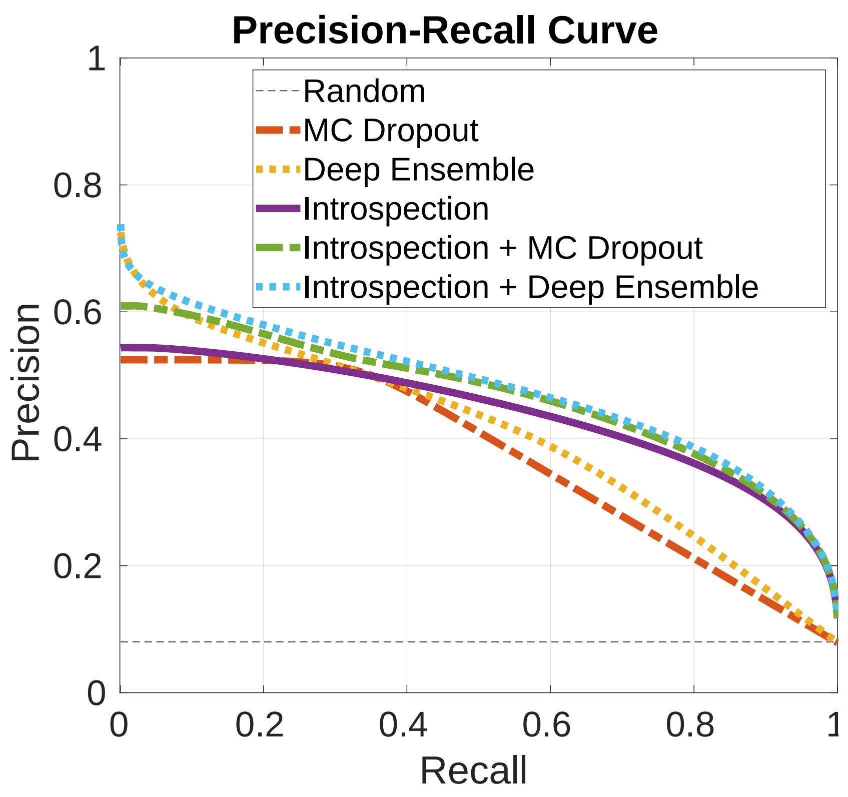


Figure 4.4 Precision-recall curves for the proposed introspective approach, the two reference approaches, and combinations of introspection and the reference approaches (adapted from [22] © 2020 IEEE).

MC dropout performs the worst along the entire curve, with an overall AUC of 0.37. Deep Ensemble achieves an AUC of 0.41. This confirms the findings from the literature [12] where Deep Ensemble outperformed MC dropout. Introspection significantly outperforms both reference approaches at an AUC of 0.44, most notably for recall values above 0.4. While Deep Ensemble performs better for low recall and high precision values, it drops off significantly for increasing recall values above 0.4. For failure prediction, a high recall value is generally more desirable than a high precision, since false alarms have less safety-critical consequences.

The combination of introspection with either of the two reference approaches further improves performance, reaching an AUC of 0.47 when combined with MC dropout and an AUC of 0.48 when combined with Deep Ensemble. This suggests that introspection detects different types of failures than the reference approaches. MC dropout and Deep Ensemble are both approaches that use the model output to predict its uncertainty. Introspection is based on patterns extracted directly from the input. By using such different information sources, different failures can be detected. While introspection alone outperforms the best-performing state-of-the-art approach by 3 %, combining it with the state of the art leads to an overall improvement of 6 %. This suggests that the concept of introspection is complementary to the output-based methods that are predominant in the field of uncertainty estimation.

Class-Wise Evaluation While introspection outperforms the state of the art when averaged over all classes in each image, the different semantic classes are not equally relevant from a safety standpoint. Dynamic classes such as *Car* or *Bicycle* are much more critical to be correctly classified than background classes such as *Building*. We therefore perform a class-wise evaluation next. Calculating the precision-recall curve for all pixels of each class at once is not feasible. At 4000 test images at High Definition (HD) resolution, the score vectors of common classes would be exceedingly large. Thus, we calculate the precision-recall curve of each class for batches of 100 images at a time and then average the resulting 40 curves. For a straightforward scalar comparison, we directly report the average AUC for each class.

28 of the remaining 38 classes each comprised less than 1 % of the pixels of the test images. Seven of those classes were omitted from the precision-recall analysis since together, they made up less than 0.1 % of the pixels of the test images. Those classes are *Speed bumper*, *Tractor*, *Animals*, *Electronic traffic*, *Slow drive area*, *Blurred area* and *Rain dirt*. Having such infrequent classes is a byproduct of having a large number of classes, with A2D2 having twice as many semantic classes as Cityscapes.

The results for the 31 analyzed classes are summarized in Table 4.1. For brevity, MC dropout is referred to as "MCD", Deep Ensemble is referred to as "DE", and Introspection is abbreviated to "I". The weighted average of the AUC of the individual classes corresponds to the overall AUC from the image-wise evaluation.

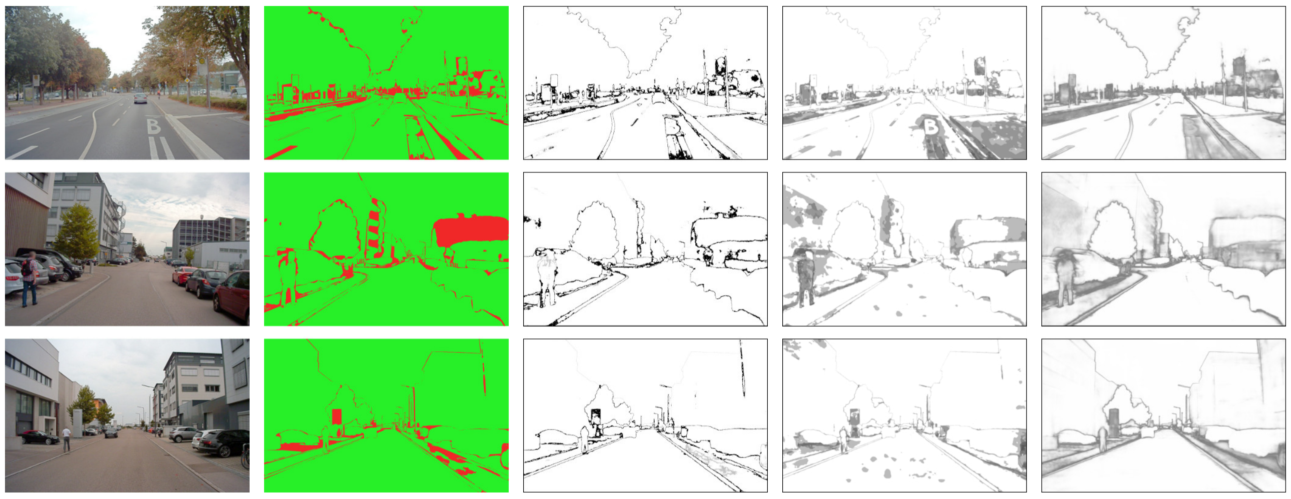
For the most common safety-critical class of *Car*, introspection outperforms the state of the art by over 9 %. While Deep Ensemble performs best for *Pedestrian*, introspection and combinations thereof again perform best for *Bicycle* and *Small vehicle*, the class of motorcycles and other smaller motorized vehicles. Regarding the other, mostly static classes, introspection or a combination of introspection with one of the two reference approaches mostly achieves the highest AUC. In total, using introspection leads to the best performance for 22 out of the 31 considered classes.

Class	MCD	DE	Introspection (I)	I+MCD	I+DE
Car	0.350	0.360	0.452	0.464	**0.497**
Pedestrian	0.576	**0.637**	0.553	0.597	0.608
Bicycle	0.513	0.515	0.516	**0.548**	0.545
Small vehicle	0.446	0.423	**0.581**	0.555	0.565
Truck	0.724	**0.784**	0.762	0.732	0.770
Traffic signal	0.471	0.530	0.593	0.566	**0.603**
Traffic sign	0.457	0.341	0.414	**0.465**	0.394
Utility vehicle	**0.611**	0.586	0.598	**0.611**	0.590
Sidebars	0.212	0.189	**0.244**	0.236	0.233
Curbstone	0.464	0.455	0.476	**0.510**	0.507
Solid line	0.487	0.480	0.511	0.512	**0.520**
Irrelevant signs	0.653	0.653	**0.673**	0.672	0.669
Road blocks	0.309	0.275	0.356	**0.366**	0.358
Non-driveable street	0.606	0.745	0.783	0.704	**0.796**
Zebra crossing	0.161	0.151	**0.184**	0.181	0.179
Obstacles/trash	0.806	**0.821**	0.765	0.783	0.776
Poles	**0.539**	0.521	0.498	0.531	0.510
RD restricted area	0.594	0.638	**0.709**	0.663	0.707
Grid structure	0.564	0.632	0.671	0.649	**0.684**
Signal corpus	0.602	**0.607**	0.577	0.588	0.585
Driveable cobblestone	0.501	0.507	0.526	**0.540**	0.531
Nature object	0.332	0.391	0.381	0.419	**0.437**
Parking area	0.678	**0.804**	0.663	0.665	0.718
Sidewalk	0.407	0.470	0.488	0.494	**0.555**
Ego car	0.273	0.289	0.333	0.328	**0.348**
Painted drive instruction	**0.538**	0.431	0.393	0.464	0.390
Traffic guide object	0.471	0.442	0.438	**0.525**	0.476
Dashed line	0.535	**0.615**	0.530	0.560	0.579
RD normal street	0.338	0.259	0.263	**0.357**	0.294
Sky	0.303	0.244	0.286	**0.339**	0.313
Building	0.346	0.346	0.332	**0.386**	**0.386**
All classes	0.373	0.408	0.440	0.467	**0.479**

Table 4.1 Area Under Curve (AUC) of the precision-recall curves for each individual class. Using introspection outperforms both MC dropout (MCD) and Deep Ensemble (DE) in 21 of the 31 semantic classes.

4.1.3.4 Visualization

Next, we visualize the predicted error maps from each investigated method compared to the ground-truth error maps made by the baseline semantic segmentation model in Figure 4.5. In Figure 4.5a), three sample images from the A2D2 data set are shown. The actual error of the prediction generated by S_{semseg} is pictured in Figure 4.5b). Correct predictions are highlighted in green, while failures are shown in red. The pixel-wise predicted failure probabilities from MC dropout in Figure 4.5c) show that MC dropout mostly detects class borders, while generally missing larger patches of misclassified pixels. The output of Deep Ensemble, visualized in Figure 4.5d), is smoother and also includes entire areas of predicted errors, such as the roof of the building on the right in the second row. However, multiple spurious failure predictions are present as well, such as seen on the middle of the road in the third row. Finally, introspection is shown in Figure 4.5e).



(a) Original Image **(b)** Actual Error **(c)** MC Dropout **(d)** Deep Ensemble **(e)** Introspection

Figure 4.5 Visualization of the compared failure prediction methods for three sample images (adopted from [22] © 2020 IEEE).

Introspection produces the smoothest predicted error maps and correctly detects multiple patches of misclassified pixels, such as the head of the pedestrian in the second row or the right sidewalk in the third row. The significant differences in the appearance of the predicted error maps can be explained by the fact that introspection is trained to classify visual features as failures, whereas the reference approaches rely on disagreements between multiple predictions. While the class borders tend to lead to disagreeing predictions, larger areas tend to be classified the same way by all forward passes in case of MC dropout and by all five models in case of Deep Ensemble. If the baseline model is consistently overconfident, both MC dropout and Deep Ensemble fail. Introspection operates on the features extracted from the input directly, making it more robust against overconfidence. Additionally, the failure predictions being mostly in patches instead of thin lines like for the reference approaches is beneficial for an intuitive understanding. If the edge of a pedestrian is predicted to be misclassified, it is difficult to judge for a human operator if the model has overlooked the pedestrian or not. With introspection, some areas are clearly predicted to be correct while others are consistently assigned a higher failure probability.

While introspection being based on visual features produces the most intuitive predicted error maps, it also means that only failures caused by visual patterns already encountered during training can be predicted. While this is a relevant limitation, novel objects are inherently challenging for autonomous vehicles. Dedicated systems such as Out-of-Distribution (OOD) detectors are generally more suitable for detecting such outliers. Introspection is designed to predict failures made on in-distribution data, which is the majority of data an autonomous system will face if deployed in its operative design domain.

4.1.3.5 Computational Complexity

Finally, the aspect of computational complexity is discussed. In mobile systems such as cars, real-time capabilities on restricted hardware are important for all deployed models. Since introspection, MC dropout, and Deep Ensemble all rely on running one or more forward passes of the baseline semantic segmentation model, their average inference time is inherently determined by the selected architecture and its implementation. However, in relative terms, introspection is always significantly faster than the two reference methods. MC dropout requires ten entire forward passes and the calculation of the variance of ten predicted semantic maps, increasing the inference time of the baseline model by at least a factor of ten. Deep Ensemble is faster, requiring only five forward passes to obtain the output of the ensemble. In contrast, introspection introduces only a single additional forward pass from running the introspective model. If the encoder is shared with the baseline model as done in this section, the additional inference time is further reduced to evaluating only a single additional decoder. Introspectiion for semantic image segmentation is therefore five to ten times faster than the state of the art during inference. Compared to the four additional models needed for Deep Ensemble in our experiments, only one additional model needs to be trained. While MC dropout does not require any training, the time saved by requiring one less model training procedure comes at the cost of an inference that is ten times more computationally complex than introspection in our example.

In summary, introspective failure prediction for semantic image segmentation is at least five times faster than the state of the art in our experimental setup and outperforms it by at least 3 %. Motivated by these promising results, we further explore this concept by introducing temporal information in the form of video input in the next section.

4.2 Introspective Failure Prediction for Semantic Video Segmentation

So far, introspective failure prediction for semantic segmentation is based only on spatial features. Learning which spatial features of an input image lead to misclassifications achieved state-of-the-art failure prediction performance in Section 4.1. However, driving is a dynamic task where temporal information is also highly relevant. In the context of driving, moving objects such as cars and pedestrians are the most safety critical and need to be classified correctly. Temporal information allows for inferring which objects are static and which are moving. This can improve both the initial semantic segmentation and the prediction of its failures, since small moving objects are typically misclassified more frequently than larger static structures.

In this section, we therefore extend introspective failure prediction for semantic segmentation to take sequential images in the form of videos as input. For this, we design a spatio-temporal model and introduce a new large-scale video data set to train the model. The integration of temporal information into the introspective failure prediction model is presented next.

4.2.1 Spatio-Temporal Model Design

There is a range of neural network architectures available for extracting spatio-temporal features from a sequence of images, as discussed in Section 2.1.2. Here, we use convolutional Long Short-Term Memorys (LSTMs) [78], which have shown state-of-the-art performance in semantic video segmentation [77]. Since introspective failure prediction for semantic segmentation can be approached as a semantic segmentation task itself, concepts from semantic video segmentation can be directly applied. Following [77], we include convolutional LSTM modules between the encoder and the decoder of the baseline model S_{semseg}. In a video sequence, the segmentation for the current image I_t is then obtained by also using the previous three images I_{t-1}, I_{t-2}, and I_{t-3}. The sequence length of four follows the established number in the literature [77, 80]. All four images are fed into the encoder. Then, the four sequential encodings, consisting of the features from the backbone and from the feature pyramid, are fed into the convolutional LSTMs. Afterwards, a baseline decoder is trained to predict the semantic target labels $L_{t,GT}$.

Once the resulting baseline semantic segmentation model S_{video} is completely trained, it is used to generate error maps as training data for the introspective decoder. The predicted labels $L_{t,pred}$ are compared to the ground-truth labels $L_{t,GT}$ to obtain the error map $E_{t,GT}$ of the current image. The introspective decoder receives the encoding of the image sequence $I_{t,...,t-3}$ from the baseline model S_{video} as input and is trained to predict the error labels $E_{t,GT}$. This way, the video-based introspective failure prediction model $S_{intro,video}$ is obtained. The resulting architecture for both the baseline and the introspective model is shown in Figure 4.6.

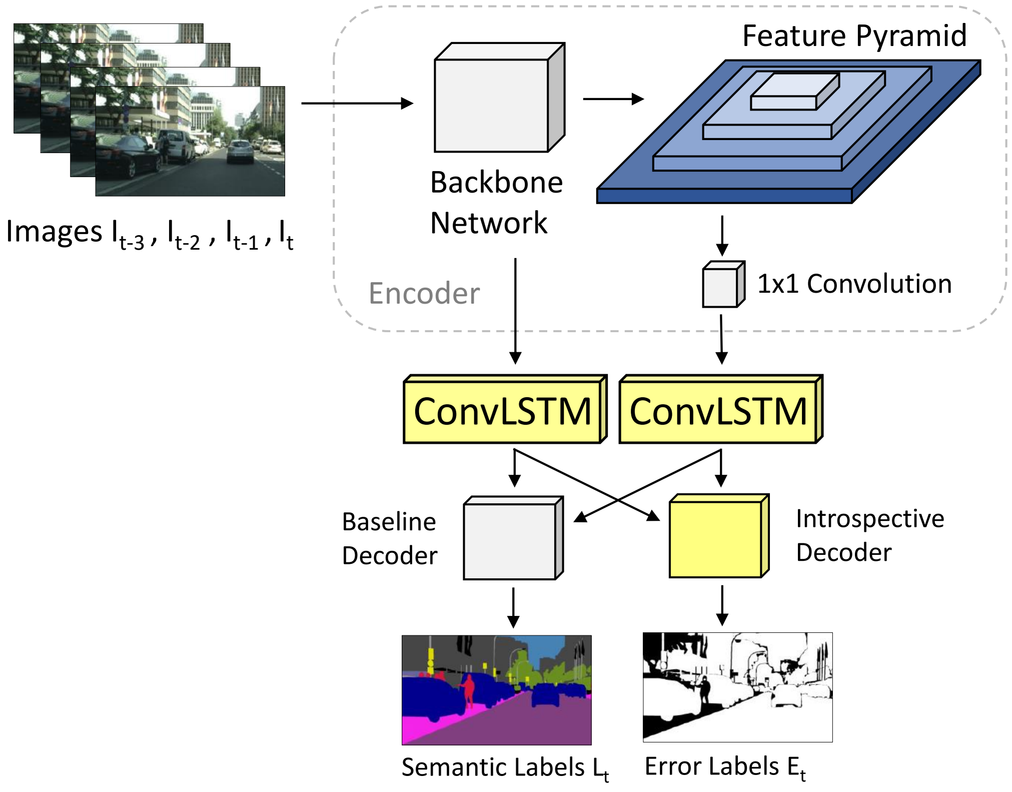


Figure 4.6 Overview of the spatio-temporal failure prediction architecture based on extending DeepLabV3+ with convolutional LSTMs and an introspective decoder.

As a second option, we also train an introspective model from scratch, without reusing the encoder from S_{video}. Being able to treat the baseline model as a black box is a relevant quality criterion since it allows to use spatio-temporal features for introspection even if the baseline model does not have a spatio-temporal encoder.

4.2.2 Densely Annotated Video Driving Data Set

For training the proposed video-based models, sequential images with pixel-wise labels are required. While some data sets such as Cityscapes [84] offer sequences of images leading up to each pixel-wise labeled frame, data sets where each frame is annotated are scarce. The CamVid data set [18] contains 900 sequential frames with pixel-wise annotations. The Highway Driving data set [19] contains 1200 such images, but only from highway environments. To the best of our knowledge, no large-scale data set of urban driving with semantic labels for each frame is available. We therefore generated a new data set to address this lack of large-scale data sets for semantic video segmentation. We use the Car Learning to Act (CARLA) simulator to create and record video sequences in an urban environment. For each frame of each sequence, the pixel-wise semantic labels are automatically provided by CARLA and stored alongside the images. We call the resulting data set the Densely Annotated Video Driving (DAVID) data set. A sample image as well as the corresponding semantic labels are shown in Figure 4.7.

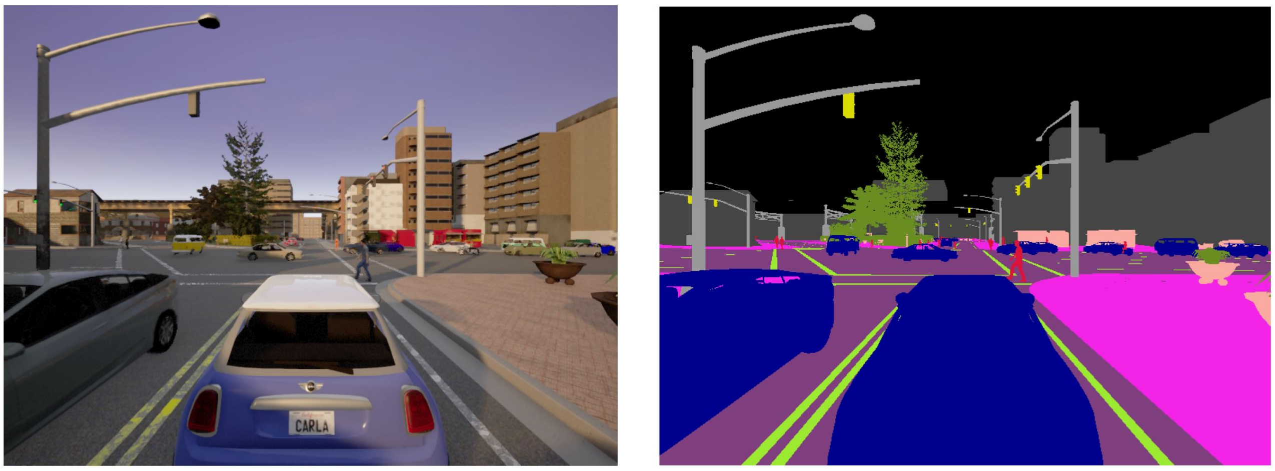


Figure 4.7 An example image from the DAVID data set alongside the automatically generated ground-truth semantic labels.

4.2.2.1 Data Set Contents

The DAVID data set contains 28 video sequences, which in total consist of 10 767 images. It is therefore at least nine times larger than both the Highway Driving and Camvid data sets. For each image, the pixel-wise semantic labels are available. On average, each video is 38.4 s long, allowing one or more complete traffic scenes such as crossing an intersection to be captured in each sequence. The frames were recorded at 10 Hz. Numerous weather conditions and driving scenarios were created for the 28 sequences. Half of the sequences contain sunny weather. Nine sequences were recorded with rain. The last five sequences have cloudy weather. The specific driving scenarios driven by the car in the default autonomous mode of CARLA are regular traffic, traffic jams, and traffic light sequences, which include both red and green lights as well as the transition between them.

In Table 4.2, the contents of each video sequence are summarized in detail. The specific duration of each sequence was selected to cover the given driving scenario as completely as possible, while avoiding extended stretches of uneventful driving on straight roads. The 28 sequences can be divided into training, validation, and testing sets as desired. In this work, we use 22 sequences for training, three for the validation set, and three for the testing set of all subsequent models.

4.2.2.2 Semantic Class Distribution

The weather and driving scenes of the video sequences were selected to capture a diverse and challenging driving environment. The default map of CARLA was used for all recordings since it contains an even distribution of structures such as buildings, vegetation, and intersections. To ensure a large amount of dynamic objects in all scenes, the number of spawned traffic participants was set to the maximum value of CARLA. This allows for driving scenes with as many encounters of cars and pedestrians as possible. In total, CARLA distinguishes between twelve semantic classes. For the few objects not belonging to those classes as well as for the sky, one additional unlabeled void class is also present in the semantic maps.

In Figure 4.8, all semantic classes present in DAVID are shown alongside the average relative frequency of the pixels from each class. The class *Road* is the most common, followed by *Unlabeled* which mostly describes the sky in each image. The other classes are distributed similarly to popular real-world data sets such as Cityscapes. For example, 0.99 % of all pixels in the images from Cityscapes are pedestrians [84]. In DAVID, 1.02 % of all pixels are pedestrians, making it a comparable representation of urban driving.

Index	Duration	Weather	Driving Scenario	Number of Images
0	30.4 s	Sunny	Driving	305
1	34.1 s	Sunny	Driving + Traffic Jam	342
2	36.2 s	Sunny	Driving	363
3	35.0 s	Sunny	Driving + Traffic Jam	350
4	33.4 s	Sunny	Driving	335
5	38.9 s	Sunny	Driving + Traffic Jam	390
6	39.6 s	Sunny	Traffic Light	397
7	44.2 s	Sunny	Traffic Light	443
8	39.3 s	Sunny	Driving	393
9	36.3 s	Sunny	Driving	364
10	43.7 s	Sunny	Driving	438
11	34.5 s	Sunny	Driving	346
12	37.9 s	Sunny	Driving + Traffic Light	380
13	32.5 s	Sunny	Driving	325
14	35.7 s	Rainy	Driving + Traffic Light	357
15	38.3 s	Cloudy	Driving	384
16	34.9 s	Rainy	Driving + Traffic Light	350
17	35.9 s	Rainy	Driving	360
18	37.4 s	Rainy	Traffic Light	374
19	43.0 s	Cloudy	Driving	431
20	36.5 s	Rainy	Driving	366
21	44.7 s	Rainy	Traffic Light	448
22	35.1 s	Rainy	Traffic Light	351
23	45.0 s	Rainy	Driving	451
24	44.3 s	Cloudy	Driving + Traffic Light	443
25	43.7 s	Cloudy	Driving	438
26	38.5 s	Rainy	Driving + Traffic Light	386
27	45.6 s	Cloudy	Driving	457

Table 4.2 Summary of the video sequences of the DAVID data set. The 28 sequences contain a total of 10 767 images with pixel-wise labels available for every frame, making DAVID around nine times larger than the largest existing data set for semantic video segmentation.

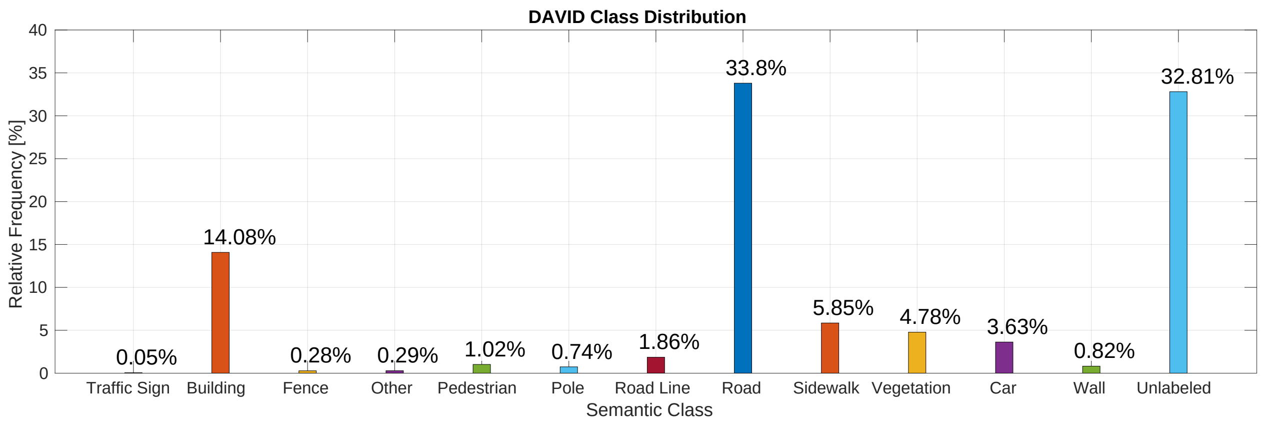


Figure 4.8 Overview of the semantic classes and their relative frequency in the DAVID data set. The frequency of critical classes such as *Pedestrian* is almost identical to real-world data sets such as Cityscapes.

4.2.3 Results

Next, we present the results of the experiments conducted to evaluate the proposed spatio-temporal failure prediction approach. First, we discuss the baseline semantic video segmentation model which consists of DeepLabV3+ extended with convolutional LSTMs. Then, we present the results of the spatio-temporal introspection approach and compare it to the state-of-the-art reference approach of Deep Ensemble as well as the single-image introspection approach from Section 4.1.

4.2.3.1 Semantic Video Segmentation

To allow the spatio-temporal introspective model to share the encoder with the baseline semantic segmentation model, both models need to be trained with video sequences instead of still images. As a baseline semantic video segmentation model S_{video}, we use the architecture shown in Figure 4.6. We train and evaluate on both Cityscapes as a benchmark real-world data set and on the DAVID data set as a larger data set where each frame has pixel-wise labels. Cityscapes provides a sequence of images leading up to each labeled frame. We follow the standard procedure of splitting up the images with publicly available labels into 2975 training images, 100 validation images and 400 test images. Due to the smaller size of Cityscapes compared to A2D2, we reuse the training set for training the introspective model. Since DAVID is around three times larger, we split the 22 training sequences into two subsets, using 11 sequences for training the baseline model and reserving another 11 sequences for training the introspective model. From the remaining six sequences, three sequences each are used for validation and testing.

The training of S_{video} is done using the same training procedure as before. We evaluate the performance of the baseline semantic video segmentation model by computing the mean Average Precision (mAP) on the test set. For reference, we also train the original DeepLabV3+ without convolutional LSTMs. The resulting mAP of the regular DeepLabV3+ and for spatio-temporal DeepLabV3+ for both data sets is summarized in Table 4.3. Similar to the literature on semantic video segmentation [77], incorporating temporal information consistently improves performance by up to 1 %. The relatively small increase can be explained by the models already being highly optimized for this task. In the case of the simulation-based DAVID data set, the baseline model performs significantly better than on Cityscapes, which can be explained by the less complex nature of simulation images compared to real-world data.

Model	**DAVID**	**Cityscapes**
DeepLabV3+	91.1 %	84.6 %
Spatio-temporal DeepLabV3+	**91.2 %**	**85.6 %**

Table 4.3 mAP of DeepLabV3+ compared to DeepLabV3+ extended with convolutional LSTMs, evaluated on the test set from Cityscapes and DAVID.

4.2.3.2 Spatio-Temporal Failure Prediction

Next, we present the results for the spatio-temporal introspective failure prediction approach. To generate the ground-truth error labels required for training the introspective model, the spatio-temporal DeepLabV3+ is used. On Cityscapes, the pixel-wise errors for the 2975 training images are obtained and used as targets for training $S_{intro,video}$. On DAVID, the ground-truth error maps for the 11 sequences reserved for introspection are generated and the introspective decoder is trained on the corresponding images. The ground-truth error maps generated by the spatio-temporal DeepLabV3+ on the validation set of both Cityscapes and DAVID are used for validation during training. The same training procedure as before is used to train the models until convergence.

As a reference approach from the literature, we again use Deep Ensemble, which outperformed MC dropout significantly in Section 4.1. As before, four additional models of the spatio-temporal DeepLabV3+ with different random initializations are trained and the pixel-wise variance of the resulting five predictions is used as a failure probability. We also compare to the single-image introspection approach from Section 4.1. Since the single-image approach does not include convolutional LSTMs in the encoder, it cannot share weights with the spatio-temporal baseline model. The single-image introspection model therefore needs to be trained from scratch without sharing any weights. For a fair comparison, we also train the spatio-temporal introspection once without reusing the weights from the spatio-temporal DeepLabV3+.

The failure prediction performance on DAVID and Cityscapes of the reference approaches, the spatio-temporal introspection approach trained from scratch, and the spatio-temporal introspection model with the shared encoder is summarized in Table 4.4. As a metric for the binary classification task, the AUC of the precision-recall curve is used as before.

Method	DAVID	Cityscapes
Deep Ensemble [12]	44.2 %	53.8 %
Single-Image Introspection [22]	55.3 %	67.3 %
Spatio-Temporal Introspection from scratch	57.0 %	71.5 %
Spatio-Temporal Introspection with shared encoder	**60.6 %**	**76.1 %**

Table 4.4 Average AUC values of the spatio-temporal failure prediction approach compared to single-image introspection and Deep Ensemble. Using temporal information consistently improves performance.

Similar to the evaluation on the A2D2 data set in Section 4.1, single-image introspection outperforms Deep Ensemble significantly by at least 11.1 %. The spatio-temporal introspection approach trained from scratch improves still-image introspection by 1.7 % to 4.2 %, demonstrating the benefits of considering temporal information. By sharing the encoder with the baseline semantic segmentation model, the spatio-temporal introspection approach further improves by 3.6 % to 4.6 %, outperforming single-image introspection by at least 5.3 %. Interestingly, the failure prediction performance on the real-world Cityscapes data set is significantly better at 76.1 % compared to the 60.6 % achieved on the simulation-based DAVID data set. This is explained by the better baseline performance on DAVID shown in Table 4.3. The better the baseline model performs, the fewer errors are available to learn from. Conversely, the more challenging and complex a segmentation task is, the better introspective failure prediction performs. This is a useful quality for deployment on real roads, where the environments are often highly complex as well.

Finally, a class-wise evaluation is performed. Since all versions of introspection significantly outperformed Deep Ensemble in the image-wise evaluation, we only report the results for single-image introspection compared to spatio-temporal introspection in the following. We compute the average AUC from the precision-recall curve for each individual class. The results for the DAVID data set are shown in Table 4.5. Interestingly, the spatio-temporal approach is slightly worse for dynamic classes such as *Car* or *Pedestrian*. However, the performance for *Pedestrian* is already very high at 87.2 %, making it more challenging to further improve for that class. The improvements of the spatio-temporal approach are most evident in street-related classes such as *Road* and *Sidewalk*. Segmentation errors in such areas are commonly caused by temporary shadows or occlusions. The sequential input offers information about the appearance of such detrimental factors, allowing the failure prediction model to detect them more precisely.

For Cityscapes, the spatio-temporal approach does improve the performance for the dynamic classes of *Pedestrian* and *Car* by 1.5 % and 10.0 %, while also improving *Road* and *Sidewalk* by at least 6.8 %. This suggests that on more complex real-world images, the addition of temporal information is beneficial both on average and for the most safety-critical classes.

Semantic Class	**Single-Image Introspection**	**Spatio-Temporal Introspection**
Traffic Sign	50.2 %	44.0 %
Building	56.7 %	61.6 %
Fence	22.3 %	25.9 %
Other	73.0 %	79.2 %
Pedestrian	88.9 %	87.2 %
Pole	74.1 %	75.0 %
Road Line	57.4 %	64.0 %
Road	52.3 %	58.8 %
Sidewalk	53.9 %	60.9 %
Vegetation	58.1 %	59.2 %
Car	61.0 %	56.2 %
Wall	42.9 %	55.2 %
Average	**55.3%**	**60.6%**

Table 4.5 Class-wise AUC values of the spatio-temporal introspection approach compared to single-image introspection, evaluated on the DAVID data set.

Semantic Class	**Single-Image Introspection**	**Spatio-Temporal Introspection**
Road	73.5 %	88.5 %
Sidewalk	81.6 %	88.4 %
Building	70.0 %	79.2 %
Wall	83.2 %	76.2 %
Fence	86.3 %	75.5 %
Pole	90.1 %	86.2 %
Traffic Light	38.1 %	29.5 %
Traffic Sign	57.8 %	52.8 %
Vegetation	75.0 %	76.1 %
Terrain	72.5 %	71.9 %
Sky	10.8 %	2.5 %
Person	67.0 %	68.5 %
Rider	89.0 %	83.1 %
Car	47.5 %	57.5 %
Truck	63.5 %	66.2 %
Bus	79.0 %	91.3 %
Train	73.3 %	66.7 %
Motorcycle	85.9 %	68.7 %
Bicycle	93.1 %	85.3 %
Average	**71.5%**	**76.1%**

Table 4.6 Class-wise AUC values of the spatio-temporal introspection approach compared to single-image introspection, evaluated on the Cityscapes data set.

4.3 Predicting Future Segmentation Errors

Temporal information allowed to improve the performance of introspective failure prediction for the current image. In addition to this, the temporal information present in video sequences can also be used to extrapolate into the future. In the literature, video sequences have been used successfully to predict the semantic maps of future frames [80, 81]. In this section, we apply this concept to introspective failure prediction and use it to predict the failures of future frames. Accurately predicting failures of semantic segmentation before they happen can give the system time to adjust its perception module or to alert the human driver of the impending failure. Next, we present our proposed approach for future failure prediction.

4.3.1 Error Map Extrapolation

The state of the art in future semantic map prediction uses sequences of previous predicted semantic maps to extrapolate into the future. We follow the architecture proposed by Rochan et al. [81] for extrapolating from a given sequence of previous predicted error maps to predict a future error map. The architecture takes a sequence of the last four predicted error maps $E_{t-3,,...,t}$ as input. The error map sequence is fed through four convolutional layers. After each convolutional layer, the resulting feature map sequence is fed into a convolutional LSTM as well as into the next convolutional layer. The output of the last convolutional LSTM is then fed into a 1×1 convolutional layer followed by an upsampling deconvolution layer. The result is concatenated with the output of the next convolutional LSTM. This process is repeated twice until all four convolutional LSTMs have been concatenated and the original resolution is restored. The target labels for the input error map sequence are the future ground-truth pixel-wise errors E_{t+n}, that are n frames in the future. The resulting architecture is visualized in Figure 4.9.

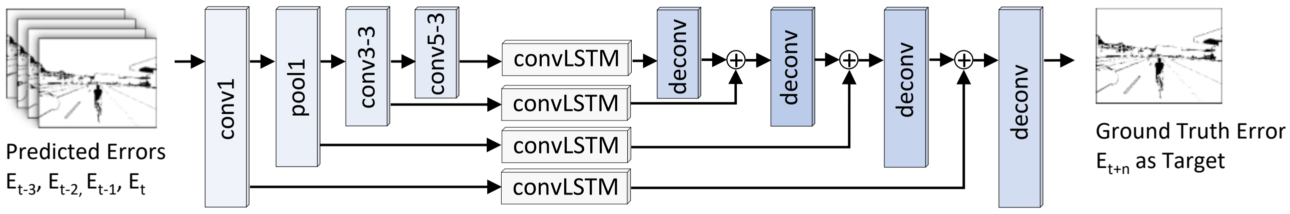


Figure 4.9 Overview of the architecture that uses a sequence of previous predicted error maps $E_{t-3,,...,t}$ to predict a future error map E_{t+n} that is n steps in the future (adapted from [27] © 2021 IEEE).

The choice of the target ground-truth error map E_{t+n} determines the behavior of the resulting model. Any integer value can be selected for n. In related work of future semantic map prediction, semantic maps up to 0.2 s in the future are predicted [81]. To evaluate how far the extrapolation can still achieve useful results, we use frames of up to 0.5 s in the future. Public data sets with sequential semantic maps commonly have a frequency of 10 Hz, same as the proposed DAVID data set. To predict up to 0.5 s into the future, we therefore train one model each with the ground-truth future error maps $E_{t+n,GT}$, $n \in [1, 5]$.

While the main goal of the proposed architecture is to predict future errors, the target can also be chosen to be the ground-truth error map E_t of the current frame, with $n = 0$. The model is then trained to generate the ground-truth error map from the predicted error map of the current frame plus the previous three predictions. When the architecture from Figure 4.9 is trained like this, the resulting model works similar to a denoising autoencoder. The model receives a partially incorrect error map and tries to reconstruct the ground-truth error map from it. It can therefore be seen as refining the initial error map prediction. While not the focus of this section, we include the special case of $n = 0$ in the subsequent experiments.

4.3.2 Results

We evaluate the proposed future failure prediction approach for semantic video segmentation on two data sets. First, we use the DAVID data set as before, which has been created for this purpose. For an evaluation on a real-world data set, we cannot use Cityscapes anymore. Ground-truth error maps are required for every single frame of each sequence, which requires ground-truth semantic labels for every frame. Cityscapes only offers semantic labels for one frame per sequence. Therefore, we use the largest publicly available video data set with pixel-wise labels for each frame, the Highway Driving data set [19]. It contains 1200 images among 20 video sequences collected on highways. We use 15 sequences for training both the semantic video segmentation model and the spatio-temporal failure prediciton model. Of the remaining sequences, two are used for validation and three are used for testing.

For both data sets, a baseline segmentation model S_{video} and an introspective failure prediction model $S_{intro,video}$ is trained as described in Section 4.2. For the DAVID data set, the same models as in Section 4.2 are used. For the Highway Driving data set, the spatio-temporal DeepLabV3+ achieves an mAP of 91.84 %. The high performance is due to the simpler, repetitive structure of highways compared to urban driving. In a precision-recall analysis, the corresponding spatio-temporal introspection model achieves an AUC of 77.7 %.

With all necessary models trained, the predicted and ground-truth error maps are obtained for all frames. Then, a future failure prediction model S_{future} is trained with target labels from n frames in the future. Its input is the predicted error sequence $E_{t-3,...,t}$ and its target is the ground-truth error map $E_{t+n,GT}$. We train one model with $n = 0$ as a potential refinement of the current prediction. For future failures, we train one model each with $n = 1, 2, .., 5$.

To the best of our knowledge, no approach for predicting future failures of semantic segmentation is available in the literature. As a simple baseline predictor, we therefore copy the predicted error map E_t for future time steps $t + 1, t + 2, ..., t + 5$. This allows determining if the future failure prediction offers any benefits over simply using the current failure prediction to make decisions about the next five frames.

We evaluate the proposed future failure prediction models S_{future} as well as the baseline predictor *Copy* E_t using a precision-recall analysis. The AUC of all models for $n = 0, 1, ..., 5$ evaluated on the DAVID data set is shown in Table 4.7.

	t	t+1	t+2	t+3	t+4	t+5
Copy E_t	60.6 %	57.5 %	55.4 %	54.2 %	53.4 %	53.0 %
S_{future}	**75.8%**	**72.9%**	**71.4%**	**69.6%**	**67.3%**	**64.4%**

Table 4.7 AUC of the proposed approach to predict failures of images $n = 0, 1, ..., 5$ frames in the future, evaluated on the DAVID data set.

The refinement model trained with the current ground-truth labels at $n = 0$ significantly improves the failure prediction performance from 60.6 % to 75.8 %. This is potentially due to the low initial failure prediction performance, where large gains are comparatively easy to achieve. For future frames up to $n = 5$ time steps in the future, the corresponding S_{future} model consistently improves the performance by at least 11.4 %. Notably, the performance for failures five frames in the future is still higher than the performance of the original spatio-temporal introspective failure prediction model for the current frame. While DAVID is simulation-based and thus less challenging than real-world data sets, these results demonstrate the potential of using sequences of previous predicted error maps to extrapolate into the future.

The performance on the Highway Driving data set is shown in Table 4.8. For real-world images, the refinement approach of training S_{future} with the ground-truth error maps of the current frame does not improve performance. A possible explanation is the labeling strategy used for the Highway Driving data set, where all class boundary areas are labeled as "void" and not used during evaluation. Since the class boundaries are typically where many misclassifications occur, not counting those pixels reduces the effect that a refinement of the class borders could have had.

For the main purpose of predicting failures from future frames, S_{future} improves the baseline predictor *Copy* E_t significantly by at least 4.5 %. Five frames in the future, the corresponding S_{future} model outperforms *Copy* E_t by 8.2 %, indicating that the model is capable of accurately extrapolating into the future.

	t	t+1	t+2	t+3	t+4	t+5
Copy E_t	**77.7%**	69.4 %	66.3 %	64.3 %	63.2 %	62.3 %
S_{future}	74.3 %	**73.9%**	**73.4%**	**72.4%**	**71.6%**	**71.5%**

Table 4.8 AUC of the proposed approach to predict failures of images $n = 0, 1, \ldots, 5$ frames in the future, evaluated on the Highway Driving data set.

4.4 Chapter Summary

In this chapter, we introduced an introspective failure prediction approach for component-level failures. The task of semantic segmentation was used as one of the most relevant components of the perception module of an autonomous system.

First, we developed a framework for predicting pixel-wise failures of semantic image segmentation in Section 4.1. The framework is based on the concept of introspection. Pixel-wise failures of a given baseline semantic segmentation model are recorded and used as target labels for an introspective failure prediction model. The introspective model can also be implemented as a semantic segmentation model. By reusing the encoder of the baseline segmentation model, the introspective model is faster to train, faster in inference, and shows improved performance. Evaluated on the A2D2 data set, the proposed introspection approach outperforms the state of the art by over 3 %, while being five times faster during inference.

In Section 4.2, we extended this approach by using video data as input. For this, we extended the baseline semantic segmentation model with convolutional LSTMs to learn from spatio-temporal features. The introspective model is extended in the same way. To allow training the semantic video models on a large data set, the large-scale Densely Annotated Video Driving (DAVID) data set was generated in the CARLA simulator. The proposed spatio-temporal failure prediction model outperforms the state of the art by at least 16.4 % and outperforms single-image introspection from Section 4.1 by at least 5.3 % when evaluated on Cityscapes and DAVID.

Finally, in Section 4.3, the spatio-temporal information available in video input was exploited to extrapolate into the future and predict pixel-wise failures for future frames. An encoder-decoder architecture with convolutional LSTMs was trained with sequences of the previous predicted error maps as input and the ground-truth future error map as target labels. Both on DAVID and the Highway Driving data set, the proposed approach outperforms a baseline predictor by at least 8.2 % when predicting failures for images five frames into the future.

5 Failure Prediction Applications

In Chapter 3 and Chapter 4, failure prediction approaches for both system-level and for component-level failures were designed, resulting in a comprehensive framework that monitors an autonomous system on multiple levels. The most straightforward application of such a failure prediction framework is to warn a human operator in advance to give them enough time to prepare for taking over control. In addition to preparing a takeover maneuver, information obtained from failure prediction can also be used to correct or adjust the system, ideally avoiding a takeover from being necessary in the first place.

In this chapter, we present three approaches that build on the insights from the proposed failure prediction methods to either correct detected failures or to preemptively avoid predicted failures. We propose an approach for correcting failure regions in semantic segmentation and a reverse error modeling approach for correcting pixel-wise errors made by a semantic segmentation model. Then, we introduce an introspective sensor monitoring system where failures of individual sensors are predicted to adjust a fusion-based object detection system accordingly.

Some of the concepts and contributions of this chapter have been published in [24, 28, 29].

5.1 Improving Visual Scene Perception Using a Two-Stage Approach

When visualizing the pixel-wise introspective failure prediction for semantic segmentation in Section 3.3, it can be seen that introspection is capable of detecting entire areas of failures in addition to the typically erroneous class boundaries. In this section, we propose to localize such failure regions and then reclassify them to improve the overall visual scene perception. For this, we follow the strategy humans use to improve the perception of complex scenes. Human vision follows a two-stage approach termed the *Zoom Lens Model* in the literature [20]. We introduce this model and the proposed application to computer vision next.

5.1.1 Zoom Lens Model

Since human perception is the source of most ground-truth annotations used in computer vision tasks, the strategy employed in human vision to understand complex scenes is a useful source of inspiration. Early investigations of human perception showed that not everything in a scene is processed simultaneously. A first analysis of the scene is used to then focus attention on individual locations [173]. Specifically, the first assessment of a scene is performed at an overall low resolution power, with the attention allocated across the entire visual field [174]. Then, local visual cues are followed to inspect individual areas with increased mental processing power. This two-step process of performing an initial global assessment followed by a zoomed-in local refinement was termed the *Zoom Lens Model* [20]. Dividing the scene perception into two stages allows humans to quickly understand the general contents of a scene in a first look. The second, focused look at initially unclear or confusing areas allows to then obtain a detailed understanding of the entire scene without needing to physically move.

We propose to apply this two-stage process to computer vision and implement a two-stage visual scene perception framework directly inspired by the *Zoom Lens Model*. To the best of our knowledge, our work is the first to use this concept for semantic segmentation. In the literature, some related concepts such as analyzing different parts of an image at different resolutions have been used before. For example, Tesla performs object detection once on the entire image, and then a second time on high-resolution crops of the center of the image [175]. Their approach lacks any information about which regions have not been properly assessed in the first step. The concept of next-best view prediction [176] does rely on assessing the short-comings of the current view to then add new perspectives for an improved perception. For this, physical movement of the sensor is required. In contrast, the proposed approach operates on the same image. This is similar to humans not needing to move to better understand a scene.

The first stage of the proposed approach that mimics the human *Zoom Lens Model* is semantic segmentation of the entire image. This corresponds to the first global assessment at an evenly distributed, but overall low resolution. Then, pixel-wise failure prediction is used to extract local failure regions. For the second, focused look, we use a local classifier trained with the most relevant objects to be expected in the scene. The local classification of each failure region is then used to update the initial semantic prediction from the first stage. Since the first step consists of already available state-of-the-art semantic segmentation, we focus on the second step in the following.

5.1.2 Failure Region Detection and Correction

The second stage in the human *Zoom Lens Model* is based on local visual cues to decide where to look twice. For semantic segmentation, we use pixel-wise failure prediction such as proposed in Chapter 4 to provide such cues. We obtain local failure regions from a pixel-wise predicted error map, classify each failure region and then perform an update. Next, we discuss the proposed pipeline for predicting distinct failure regions for a given image and its predicted semantic map.

5.1.2.1 Failure Region Prediction

In Chapter 4, we proposed several approaches for predicting pixel-wise failures of semantic segmentation. Now, we are interested in obtaining entire regions with a high failure density. We propose a multi-step pipeline for predicting failure regions. An overview of the proposed workflow is given in Figure 5.1. The Region of Interests (ROIs) of a pixel-wise failure prediction are filtered, clustered, and processed to obtain a set of distinct failure regions. We explain each step in more detail in the following.

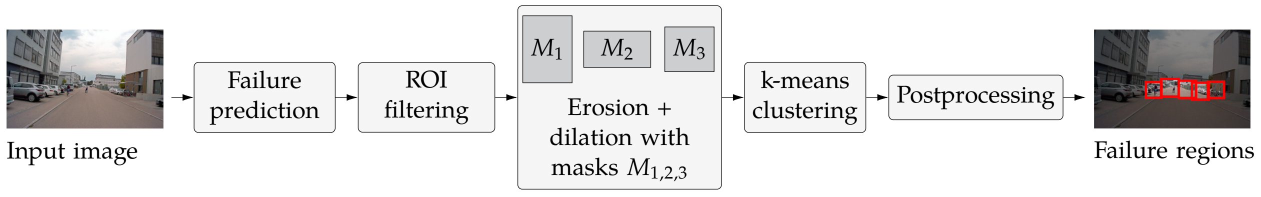


Figure 5.1 Summary of the pipeline to obtain distinct failure regions for the predicted semantic segmentation of a given input image. Predicted pixel-wise failures are filtered with a predicted ROI mask. Then, erosion and dilation masks are applied to remove spurious predictions. After clustering and a final postprocessing step, a set of bounding boxes of local failure regions is obtained (adopted from [24] © 2020 IEEE).

Failure Prediction To extract failure regions from the predicted semantic segmentation, we first predict its pixel-wise failures. For this step, we use the state-of-the-art introspection approach introduced in Section 3.3. The encoder from the semantic segmentation model used to obtain the predicted semantic map from the first stage of the scene perception is frozen and reused for the introspective model. Then, pixel-wise error maps generated with the baseline semantic segmentation model are used as target labels for training the introspective decoder. The introspective decoder assigns each pixel a failure probability. A visualization of the resulting pixel-wise failure prediction for a sample image from the A2D2 data set is shown in Figure 5.2. To better highlight the impact of each step in the proposed failure region detection pipeline, a failure probability of 0 is shown as black and a predicted probability of 1 is shown as white in this section.

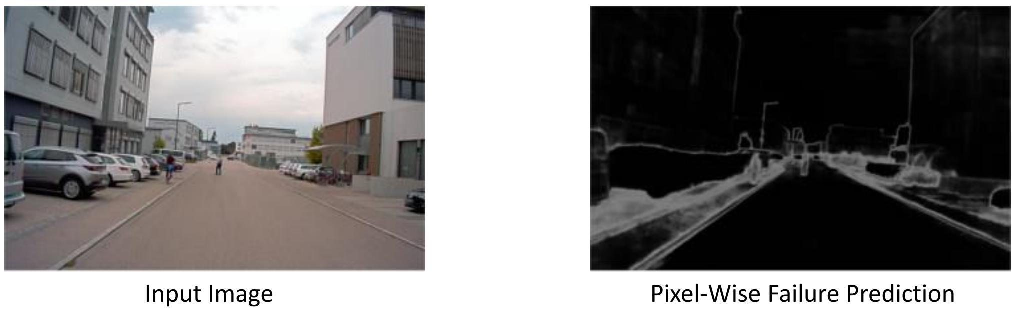


Figure 5.2 Visualization of the pixel-wise failure prediction (right) for a given input image (left), which is the basis for the proposed failure region detection system.

ROI filtering Not every predicted pixel-wise failure is relevant for driving. Segmentation errors made in the background or at the edges of the image are likely to be less safety critical, for example. To automatically select the most relevant regions based on the visual features of the image, we use a state-of-the-art ROI prediction model for the next step of the pipeline. Xia et al. [99] proposed an ROI model trained with recorded human gaze data on a large-scale driving data set. Their model allows predicting which areas are the most interesting from a human's perspective. We use the predicted ROIs as a filter to only keep the predicted failures within interesting regions. Relying on ROI prediction does not necessarily increase the complexity of the system. Such a model is likely to be implemented in some form already in the vehicle, since ROI prediction has been shown to improve the performance of driving models [177].

We visualize the effect of filtering the failure prediction with the predicted ROIs in Figure 5.3. Four distinct ROIs are visible, all of which correspond to critical objects on the road, such as the closest cars on both sides of the road as well as the pedestrians on the street. After the ROI filtering, the more irrelevant predicted failures corresponding to the sidewalk or the buildings in the background are removed.

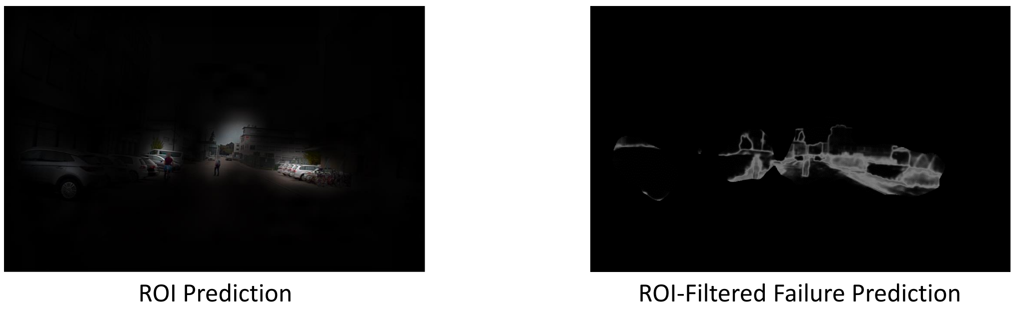


Figure 5.3 Visualization of filtering the pixel-wise failure prediction with the predicted ROIs (left). For the sample image, the filtered failure prediction (right) contains only failures corresponding to areas that contain cars or people.

Smoothing While the ROI-filtered failure prediction is focused on relevant areas, these areas still contain irrelevant, spurious failure pixels. A common source of unwanted predicted failure pixels are object boundaries. Boundaries are often misclassified, but do not correspond to entire failure regions. Predicted failure patches with an insignificant spatial structure, e.g., thin lines or individual pixels, should therefore be removed to avoid being detected as a region.

First, the failure prediction is binarized. The failure probabilities at this point still range from 0 to 1. Pixels with a low failure probability can be removed to further focus on relevant failures. Since false positives are generally less critical in safety-critical failure prediction tasks, we empirically select a low threshold of 0.3 for the binariziation. Any failure probability above this threshold is set to 1 and the rest is set to 0.

The binary failure map is then smoothed to remove disconnected, spatially insignificant failure regions. We use two morphological operations, erosion and dilation. First, erosion removes smaller structures, then dilation restores the original shape of larger structures that were not filtered out by the erosion. For both operations, we apply one horizontal mask M_1, one vertical mask M_2, and one square mask M_3, as shown in Figure 5.1. For images in HD resolution, we chose the pixel-wise dimensions of the masks as $M_1 = 5 \times 15$, $M_2 = 15 \times 5$ and $M_3 = 10 \times 10$. The effect of binarizing and smoothing the ROI-filtered failure prediction is shown in Figure 5.4. Thin lines are largely removed, with mostly connected failure regions remaining.

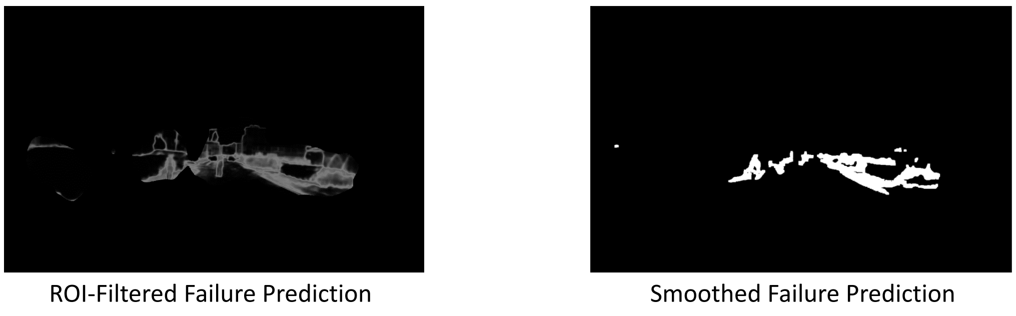


Figure 5.4 Visualization of binarizing and smoothing the ROI-filtered failure prediction (left) with three erosion and dilation masks. The smoothed failure prediction (right) mostly consists of connected failure regions.

Clustering Finally, the filtered, binary map can be clustered into distinct failure regions. We use the well-established k-means algorithm [178]. For the number of clusters to be generated for each image, we select $k = 6$. This value is highly scenario-dependent and was tuned for the driving context. It should be noted that it is rare that more than six objects are both critical and misclassified at the same time, considering the accuracies of beyond 90 % exhibited by state-of-the-art semantic segmentation models on challenging driving data sets [84]. In preliminary experiments, a higher value for k did not result in more useful detected failure regions, but led to excessively small clusters that did not contain distinct objects anymore. In Figure 5.5, the clusters generated with k-means on the sample image are visualized with a red bounding box around the pixels belonging to each cluster. Figure 5.5 shows only five clusters, since the sixth one was removed in the final postprocessing step explained next.

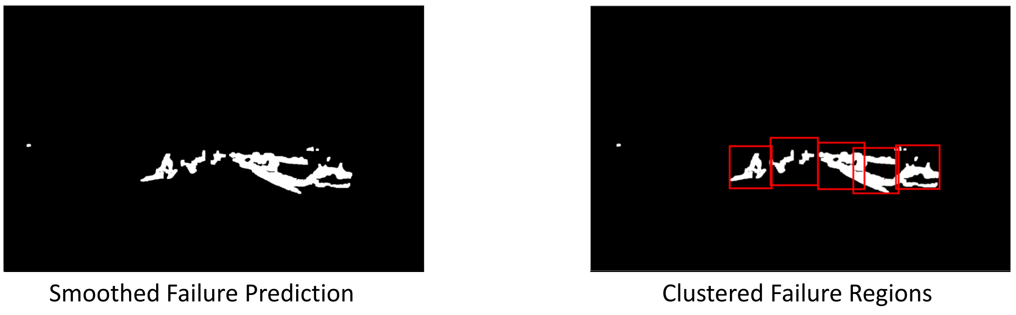


Figure 5.5 Visualization of the clusters obtained from the smoothed failure prediction using the k-means algorithm. Each cluster is shown via a red bounding box around the pixels belonging to the respective cluster.

Postprocessing A last postprocessing step ensures that only useful and valid clusters are used in the final failure region list. Several requirements that the clusters have to fulfill are verified. First, we define a minimum cluster size to avoid clusters that do not contain distinguishable visual features. Thus, any cluster smaller than 5 % of the image width is removed. The sixth cluster from Figure 5.5 was removed due to this requirement, since it comprised only the small white area on the far left of the image. Second, excessively large clusters that are bigger than 25 % of the width and thus typically contain multiple objects are removed as well. Third, even clusters with a reasonable size can contain visual contents that do not correspond to any objects. A simple entropy-based filter is applied to address this. The Shannon entropy [179] of each grayscale image patch is calculated as a straightforward measure of how distinct the visual features in each region are. Any cluster with an entropy of less than 6 bits per pixel is removed. Regions with an entropy below this empirically selected value typically have uniform and blurred contents. The remaining clusters are used as a list of detected failure regions. Next, a local classifier is designed to allow for reclassifying each detected failure region.

5.1.2.2 Failure Region Classifier

For correcting the detected failure regions, we use a classifier as an equivalent to the second look humans perform in the *Zoom Lens Model*. Compared to the global segmentation, a local classification is a less complex task. Only one object needs to be classified in each patch. The design of the failure region classifier depends on the data set used to train the semantic segmentation model. In the context of failure prediction for autonomous driving, the most relevant classes are road participants. They include cars, bicycles, pedestrians, motor bikes and larger vehicles such as trucks. To obtain images that contain only those classes, bounding box annotations are required. Since most autonomous vehicles are equipped with object detectors, such labels are likely available in the training set anyway. Then, the bounding box image patches from each road participant class are cropped out of the training images and used to train a road participant classification network. Despite the postprocessing in the failure region detection, not all clusters contain road participants. To address this, we add a *Background* class that consists of randomly sampled crops from the background of the training images. For the resulting task of image classification, any state-of-the-art architecture such as ResNet50 [40] can be used. An example is shown in Figure 5.6, where each of the five remaining clusters obtained from the failure prediction is classified as either a road participant or as background.

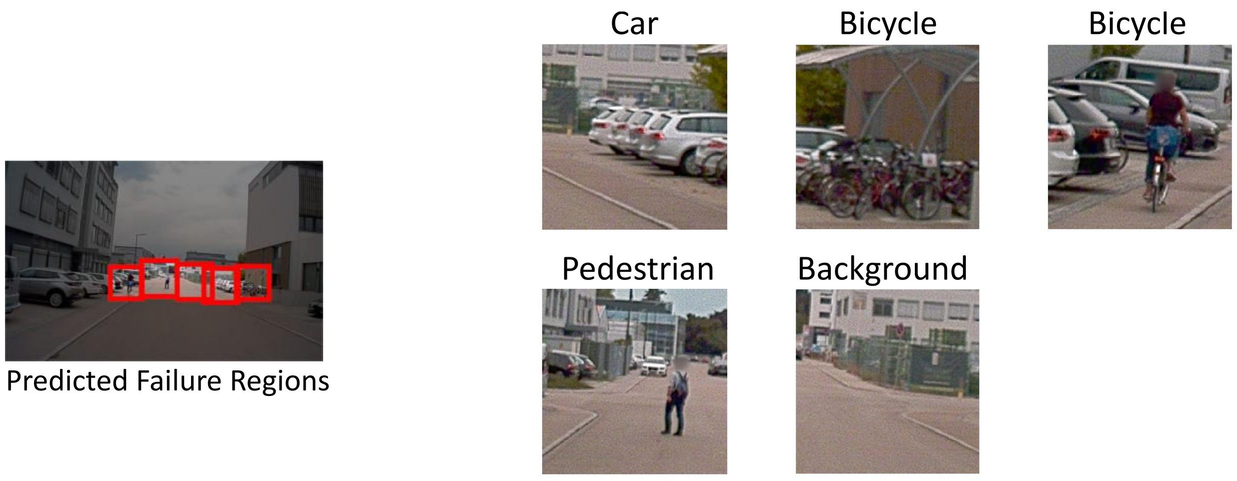


Figure 5.6 Example for the classification of the predicted failure regions.

5.1.2.3 Semantic Segmentation Update

The information from the classification can now be used to update the predicted semantic map on a pixel-wise level. This requires localizing the object that caused the classifier to make its prediction. In Chapter 3, Class Activation Maps (CAMs) were used to visualize which features the introspective models used to make their prediction. Here, we use CAMs to localize which pixels resulted in the current classification. The effectiveness of CAMs to localize objects in images has been shown before in works such as [102]. We filter the smoothed failure prediction with the CAM from the classification of the corresponding input image, removing all failure pixels where the CAM value is below 0.5. The resulting CAM-filtered failure prediction contains only those failure pixels that are part of the classified objects. Irrelevant pixel-wise failures in the background of the patch are thus removed. This process is visualized in Figure 5.7 for an exemplary failure region containing a pedestrian. The CAM demonstrates that the lower part of the pedestrian was used by the classifier to reach its prediction. Thus, all predicted failures among the lower part of the person remain, while the pixel-wise failures of the cars in the background are filtered out.

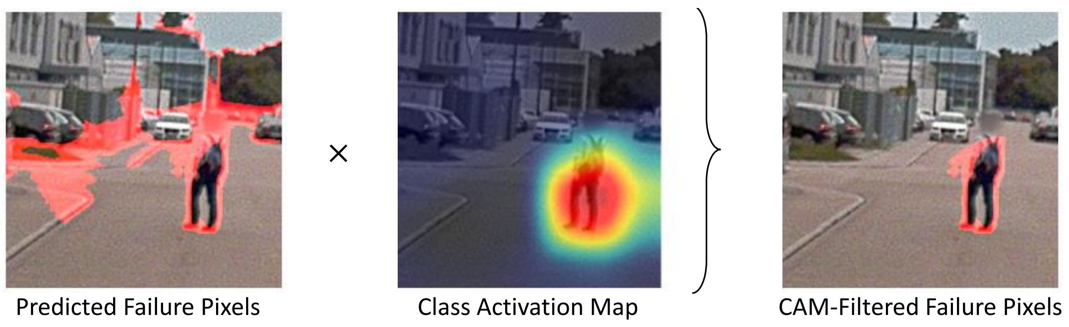


Figure 5.7 Visualization of the CAM filtering of the smoothed failure prediction.

Finally, the predicted semantic segmentation from the first step of the visual scene perception can be updated with the results of the second stage. All pixels of the CAM-filtered failure prediction are changed to the class predicted by the classifier. Even if the classifier did not classify the patch correctly, only predicted failure pixels are changed, reducing the likelihood of introducing new errors to the predicted semantic map. The update of the semantic map is visualized in Figure 5.8. The input image and the ground-truth labels are shown as well as the output of the first and second stage of the proposed approach. Most notably, the pedestrian (yellow) is initially misclassified as a bicycle (orange). In the updated semantic map, the person is now largely changed to the class *Pedestrian* as seen by the yellow pixels introduced to the area.

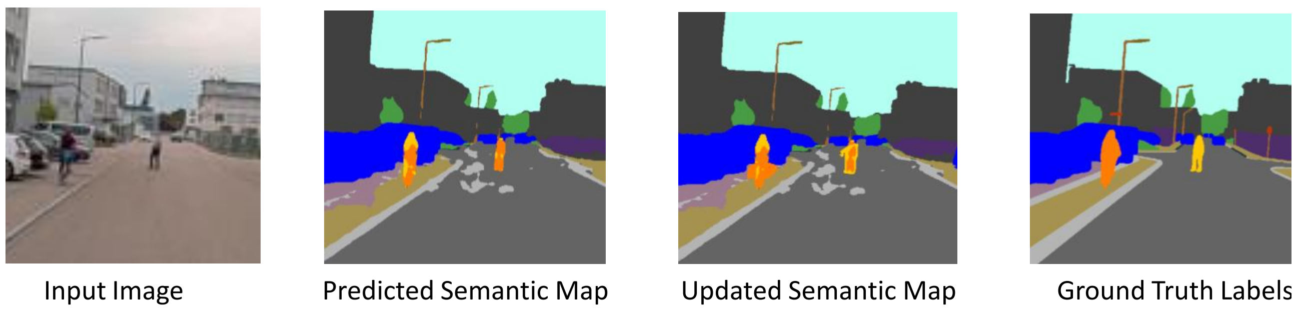


Figure 5.8 Visualization of the update of the predicted semantic map. The pedestrian (yellow) is initially misclassified as a bicycle (orange), which is partially corrected in the update step.

5.1.3 Results

Next, we present results from the experiments we performed to evaluate the proposed two-stage approach for improving visual scene perception. Since this approach is based on the introspective failure prediction method introduced in Section 3.3, we again evaluate on the A2D2 data set [86]. For the first stage of the proposed approach, a DeepLabV3+ model is trained on the same 16 000 training images as in Section 3.3. The training set also provides bound box annotations. These are required to obtain the training images for the road participant classifier. The other 16 000 images from the original training set of A2D2 are used to train the introspective decoder. We use the same models trained in Section 3.3. The proposed second stage for improving the semantic segmentation is then applied and evaluated on the 4000 test images. The baseline DeepLabV3+ model has a test accuracy of 94.0 %. While this is an overall high performance, the 6 % of incorrect predictions can contain safety-critical objects such as pedestrians, motivating the application of the proposed approach. The following experiments are thus focused on detecting and correcting the failure regions among these most challenging 6 % of pixels that were misclassified by the DeepLabV3+ model.

5.1.3.1 Failure Region Prediction and Reclassification

The failure region prediction pipeline shown in Figure 5.1 is applied to each test image to obtain a list of failure regions for each image. Pixel-wise failures are predicted using the introspection model from Section 3.3. Then, the pre-trained ROI prediction model from [99] was applied to the input image and the resulting failure prediction was filtered with the predicted ROIs. The filtered failure map was binarized and smoothed using the three morphological masks M_1, M_2 and M_3, which we applied first as an erosion and then as a dilation. Then, six clusters are extracted from the resulting failure map using the k-means algorithm. Clusters that were too small, too big, or where the pixels had a low Shannon entropy [180] were removed.

To reclassify the obtained predicted failure regions, a road participant classifier is trained using the cropped bounding box patches of the classes *Car*, *Pedestrian*, *Bicycle*, *Small vehicle*, and *Truck* from the training set. We focus on dynamic classes, since they constitute the most safety-critical objects in the context of driving. As a none-of-the-above class, we also add *Background* as a final class to the training set for the classifier. The training images of this class are generated by cropping squares of sizes between 5 % and 25 % of the image width from the parts of the training images that do not contain any dynamic road participant class. Since *Car* is far more common than the other classes, we randomly downsample the cropped bounding box images from the class *Car* by removing every second bounding box. This way, the most common class *Car* is less than ten times as frequent as the least common class of *Truck*. The resulting data set used for training and validating the road participant classifier contains almost 18 000 images. Its contents are summarized in Table 5.1.

Class	**Number of Images**
Car	4589
Pedestrian	2570
Bicycle	1897
Small vehicle	498
Truck	1518
Background	6773

Table 5.1 Summary of the data used to train the failure region classifier. The bounding boxes from the most common class *Car* were undersampled by a factor of 2 to avoid an excessive class imbalance.

Using the data shown in Table 5.1, we then train a *ResNet50* classifier pre-trained on ImageNet [41]. We use 90 % of the data set for training and 10 % for validation. For testing, the predicted failure regions from the testing set will be used. To address the class imbalance, the loss was weighted inversely with the relative frequency of each class during training. We train the *ResNet50* until the validation loss has plateaued for five epochs. The resulting validation accuracy is 96.0 %. If a dynamic class is present in the predicted failure region, this result suggests that the road participant classifier will be able to accurately classify it.

Next, each predicted failure region from the 4000 test images is classified with the finetuned *ResNet50*. Each region predicted to be *Background* is discarded since no safety-critical object can be recovered from such areas. A total of 4686 failure regions are classified as one of the dynamic road participant classes. This corresponds to just slightly more than one partially misclassified road participant per test image, which is in line with the overall high accuracy of 94.0 % of the DeepLabV3+ model. From these 4686 failure regions, over 78 % actually contained a road participant, meaning that around every fifth predicted participant is a false positive. In Table 5.2, we summarize the classification accuracy for each individual class from the 3664 predicted failure regions that did contain a road participant.

Class	**Number of Failure Regions**	**Accuracy**
Car	1255	99.1 %
Pedestrian	663	63.4 %
Bicycle	743	34.3 %
Small vehicle	100	45.0 %
Truck	903	41.1 %
Σ	3664	63.8 %

Table 5.2 Classification accuracy for each road participant class present in the predicted failure regions.

The average accuracy is 63.8 %. The most common class of *Car* is correctly classified almost every time at 99.1 %. While *Truck* achieves the lowest accuracy at 41.1 %, this is partially explained by trucks often being misclassified as *Car*. Such an incorrect prediction would still be useful from a safety point of view.

An overall accuracy of 63.8 % is significantly lower than the validation accuracy of 96.0 %. This is due to the inherently challenging nature of failure regions. During training, the bounding boxes contained all instances of the dynamic classes. During testing, the patches classified by the model are predicted failure regions. They only contain areas where the first stage of semantic segmentation already made significant mistakes. We therefore focus on the most challenging visual features present in each image, such as blurry, ambiguous, or occluded areas. Of such areas, the classifier is capable of still classifying almost two thirds correctly.

5.1.3.2 Semantic Segmentation Update

The output of the failure region classifier is effectively an object detection list and can already be used by combining it with the object detection system of the vehicle, for example. Here, we additionally use the failure region classifications to perform the proposed pixel-wise update of the predicted semantic map. For each failure region classified as containing a road participant, we use the filtered, smoothed pixel-wise failure prediction used to cluster the corresponding region. Then, the CAM of the *ResNet50* is obtained and used to filter the failure prediction. Only the failure pixels belonging to the predicted class remain. These failure pixels are then changed to the predicted semantic class in the initial semantic prediction generated by the DeepLabV3+ model.

Class	Initial Prediction	Updated Prediction
Car	89.0 %	88.8 %
Pedestrian	67.2 %	67.7 %
Bicycle	63.7 %	64.2 %
Small vehicle	76.1 %	77.0 %
Truck	86.3 %	86.3 %
All classes	78.8 %	78.7 %

Table 5.3 Class-wise and overall accuracy of the initial predicted semantic segmentation generated with the DeepLabV3+ model and the updated semantic prediction.

The class-wise accuracy of the dynamic classes and the average accuracy over all classes of the initial semantic prediction and the updated semantic prediction is summarized in Table 5.3. Except for *Car*, the average accuracy is improved for all dynamic classes. Interestingly, the average accuracy over all classes is slightly reduced by 0.1 %. The drop in overall accuracy is largely caused by background classes such as *Sky*, *Road*, and *Building* having lower accuracy values after the update. This is due to the class boundaries of dynamic objects often being changed to the dynamic class, and thus affecting some of the background pixels behind the object.

While the class-wise accuracy does indicate that the update step mostly improved performance, averaging the accuracy over all 4000 test images is not the most insightful metric. On average, less than one failure region per image was updated. The average accuracy thus cannot increase significantly. Considering we focus on the 6 % of misclassified pixels, this is also not the intention of the proposed approach. Instead, individual failure regions where a safety-critical object was overlooked by the initial segmentation should be recovered. To evaluate the performance of the proposed approach in this regard, we next analyze how many road participants were improved on an instance level. For each road participant present in the failure regions, we measure the absolute accuracy change among the pixels belonging to each participant. The absolute number of road participants at changes in accuracy from 5 % to 50 % is shown in Figure 5.9.

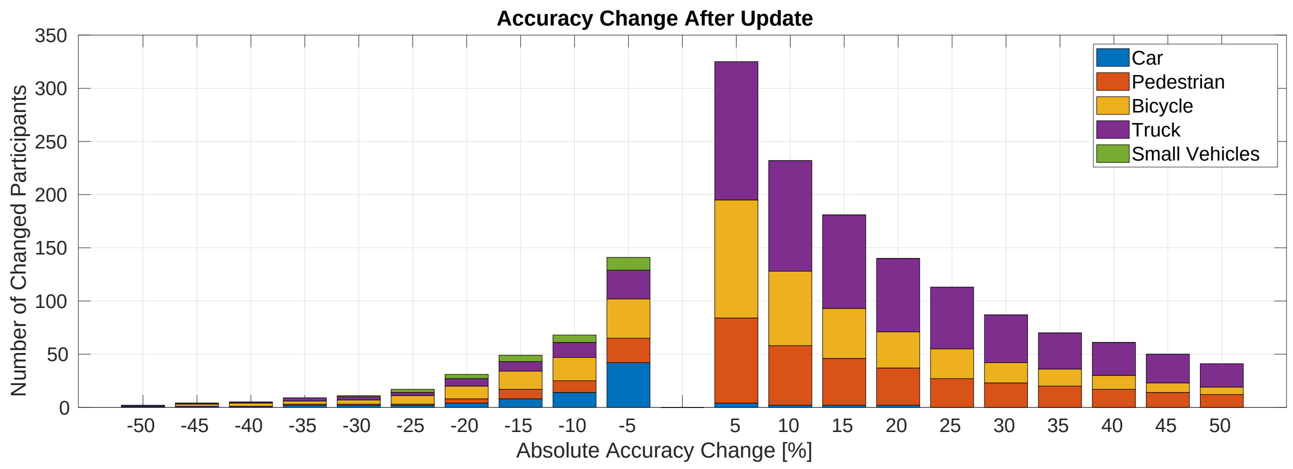


Figure 5.9 Number of road participants where the update step changed the accuracy by 5 % to 50 % in either direction. For instance, a total of 325 road participants are improved by 5 % or more, while the accuracy of 141 road participants decreases by 5 % or more (adopted from [24] © 2020 IEEE).

Figure 5.9 shows that in absolute terms, the improvement from the update step is significant. For instance, the absolute classification error from 232 road participants is reduced by 10 % or more. This means that at least one tenth of the area in the image associated to these objects now contains the correct semantic predictions, ensuring that the object is not overlooked during subsequent trajectory calculations. Considering that a total of 4000 test images was evaluated, this result indicates that in more than one out of twenty images, a misclassified dynamic object can be recovered this way. At an initial accuracy of 94.0 %, this is a useful improvement, since most images do not contain significant errors that need updating. However, in the comparatively rare case that a road participant is missed, it is safety-critical to be able to correct these errors, as done by the proposed approach.

The improvements are most prominent in the smaller and less frequent classes of *Pedestrian* and *Bicycle*. Only few instances of the most common class *Car* are improved, which is due to the first semantic segmentation stage making the fewest errors for this class and thus missing almost no cars that the proposed approach could then recover. While the accuracy improvements significantly outnumber the decreases in accuracy as shown in Figure 5.9, the absolute error of 68 participants also increases by 10 % or more. It is important to note that this decrease in accuracy only means that one road participant class was changed into a different one. No information about the general presence of a safety-critical dynamic object is lost from the semantic prediction this way.

Finally, we visualize the results of the proposed approach. Visualizing the updated maps allows for a better understanding of what causes the introduced false positives and why they are not necessarily an issue from a safety point of view. In Figure 5.10, we show five examples from the testing set. In Figure 5.10 a), the original input image is shown. Five scenes that cover a range of driving scenarios such as construction work, pedestrians crossing the street, and open driving are selected. Figure 5.10 b) shows the ground-truth semantic labels as reference. Most importantly, *Pedestrian* is shown in yellow, *Bicycle* is shown in orange and *Car* is shown in blue. In Figure 5.10 c), the initial semantic map predicted by DeepLabV3+ is visualized. The high testing accuracy is evident, with the overall semantic structure of each sample image being correctly predicted. Finally, Figure 5.10 d) shows the result of the proposed second stage of visual scene perception.

In the first row, the construction workers in the middle of the road are largely missed by the initial semantic prediction, potentially due to their orange outfits resembling the visual patterns of the orange safety markings on the truck next to them. The introspective failure prediction correctly detects those visual features as problematic and outputs a high failure probability for this area. The failure region detection then clusters this area as a distinct failure region, which is classified as *Pedestrian* by the *ResNet50*. After changing the predicted failure pixels, a distinct yellow area is introduced into the updated semantic map, indicating the presence of pedestrians on the road. Notably, the introduced yellow pixels also overwrite some of the correct background pixels of the building and the truck, while not all pixels of the pedestrians are correctly changed to yellow. The resulting average accuracy is therefore not necessarily increased, but the improvement regarding safety are dramatic. Without the update, the system would have made the dangerous assumption that no people are on the road.

The second row shows a similar effect, with two pedestrians standing between cars at the side of the road. They are again missed in the initial step, but the ambiguous features of this area are assigned a high failure probability and thus a failure region is detected. After the update, a significant yellow area is present. The system again introduces a large amount of false positives, but ensures that the car is aware of the people that could potentially enter the road at any moment.

The third row demonstrates the effect of the update step for the class *Bicycle*. In the third row, the cyclist on the right is difficult to spot among the shadows even for a human. Again, the failure prediction assigns this complex area a high failure probability. The local classifier then only needs to decide if that area contains a dynamic object or is just blurry background. The classifier correctly detects the bicycle. The CAM-filtered failure pixels changed in the updated map are again not entirely pixel-wise accurate, incorrectly changing several *Road* pixels to the *Bicycle* class. This comes at the benefit of now having the knowledge that a bicycle is present in this area, even if the blurry visual features do not allow a pixel-accurate localization.

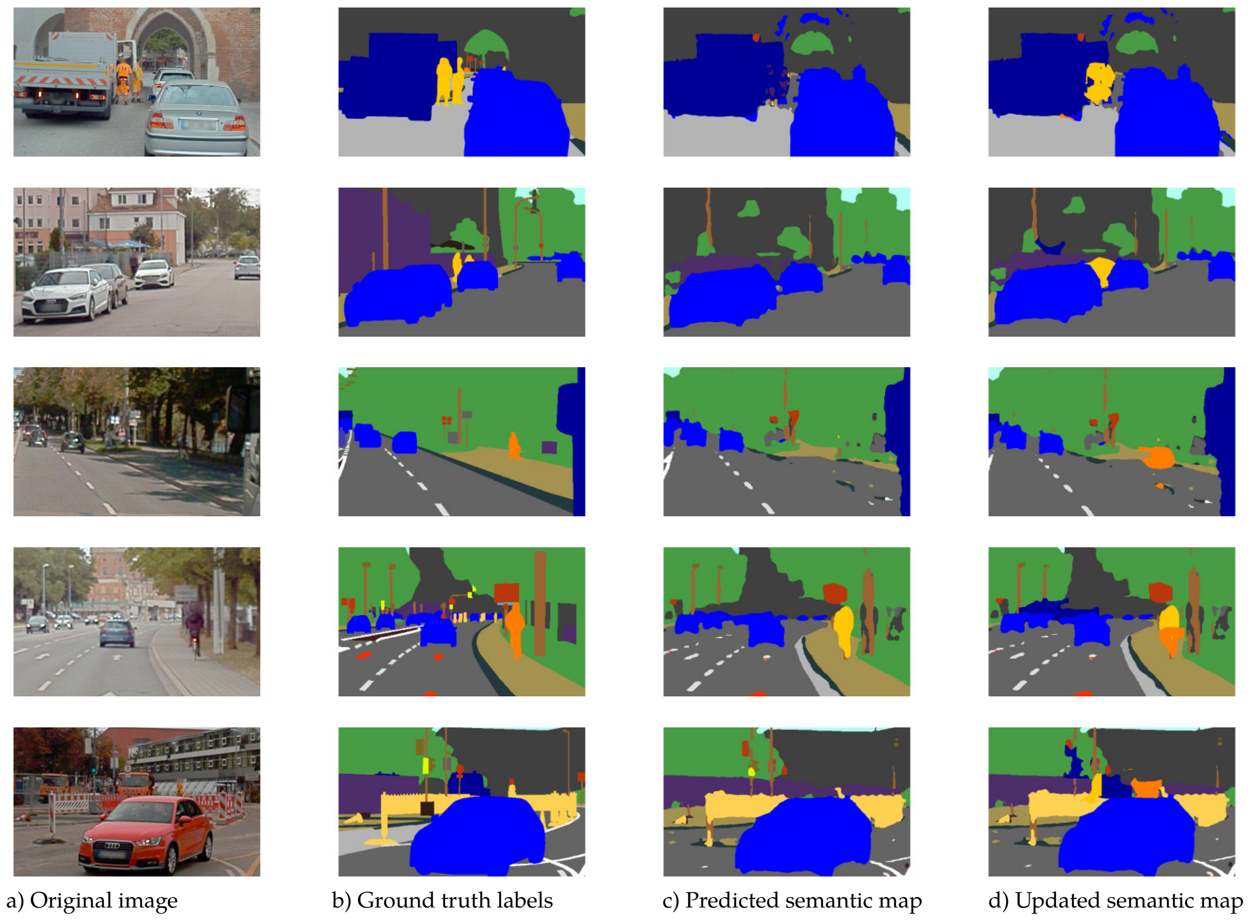


Figure 5.10 Visualization of the proposed update of the semantic prediction for five exemplary images. The input image (a) and its ground-truth labels (b) are shown next to the initial semantic prediction by DeepLabV3+ (c). The updated semantic prediction is shown in (d). *Car* is shown as blue in the semantic maps, *Pedestrian* is yellow, *Bicycle* is orange and *Truck* is dark blue (adopted from [24] © 2020 IEEE).

In the fourth row, the bicycle on the right is misclassified as a pedestrian. This can be a safety-critical mistake since unlike a pedestrian, a bicycle could join traffic on the road at any time. Interestingly, only the bike is detected as a failure region and updated. This demonstrates that only the pixels most relevant to the predicted class are changed, in this case the pixels belonging to the bike. While the final semantic map contains both a *Pedestrian* and a *Bicycle* area in this region, the knowledge that a bicycle is present at all can avoid making incorrect assumptions about the behavior of this road participant.

Finally, the fifth row shows an improvement for the class *Truck*. The truck is behind the red car and again difficult to spot. In the updated semantic map, the overlooked truck is mostly corrected as seen by the dark blue color associated to *Truck*. Before the update, the truck was classified as the same *Fence* class that surrounds it. Construction sites are an especially challenging environment for autonomous vehicles. Being aware of every potentially dynamic traffic participant is critical for safe driving. While this highlights the benefits of the proposed approach, the updated map also shows a major limitation. Next to the truck, an orange *Bicycle* area is added due to misclassifying the round lights from the traffic signs as wheels. While assuming a bicycle to be present would likely only make the system drive more slowly, such false positives need to be addressed in future work to avoid overly cautious behavior.

5.2 Reverse Error Modeling for Improved Semantic Segmentation

In this section, we present another approach for correcting pixel-wise failures of semantic image segmentation. We again follow a two-stage process of first generating an initial semantic prediction and then applying a correction step. Instead of predicting failure regions and reclassifying them, we now propose to train a model to directly correct the semantic prediction on a pixel-wise level. For this, we first introduce the novel concept of reverse error modeling for semantic segmentation.

5.2.1 Concept

State-of-the-art semantic image segmentation models inevitably make mistakes. Such mistakes can be caused by model imperfections or by problematic input, such as complex or compromised images. We propose an approach that learns to reverse the pixel-wise errors made by an arbitrary given semantic segmentation model. To achieve this, we reframe a semantic segmentation model as an error function that is applied to the ground-truth semantic labels, thus generating the slightly incorrect semantic prediction. Then, a model can be trained to approximate the inverse of this error function. Then, we train a model to approximate the reverse error function. The concept of introspection introduced in Section 3.3 is also based on learning from the recorded mistakes of a given model. Instead of then predicting a pixel-wise failure probability, we now aim for predicting the correct class instead. For this, we propose the following concept.

The task of semantic image segmentation typically consists of training a model S to assign each image I in a data set a predicted semantic label map L_{pred}. The ground-truth labels L are used as targets during training. This standard workflow of obtaining a semantic prediction L_{pred} can be written as

$$L_{i,pred} = S(I_i). \tag{5.1}$$

In practice, the prediction L_{pred} will not be identical to the ground-truth labels L. The differences between L and L_{pred} are typically measured using metrics such as the accuracy or mean Intersection over Union (IoU). During training, this difference is minimized by minimizing a loss function such as the cross entropy. After training, the model error is not further changed, meaning that given an image I, the model S will always make the same errors. These errors made by S can also be written as an explicit error function applied to the ground-truth labels L. Based on this logic, we reframe the workflow of semantic segmentation as an error function E_S that is applied by S to the ground-truth labels L to generate the predicted labels L_{pred}:

$$L_{pred} = E_S(L). \tag{5.2}$$

During testing, the ground-truth labels L are not available. Only the predicted labels L_{pred} are known. However, Equation 5.2 establishes a formal connection between the ground truth and the predicted labels. If the inverse of E_S existed and was known, the ground-truth labels L_i could be reconstructed from the predicted labels $L_{i,pred}$ by applying the inverse error function E_S^{-1} to the predicted labels:

$$L_i = E_S^{-1}(L_{i,pred}). \tag{5.3}$$

The error function E_S is a complex function that is defined by the semantic segmentation model S. If S is implemented as a deep neural network, the error function E_S is intractable and cannot be derived analytically. It is also usually not invertible. However, Equation 5.3 can be approximated. We propose to use a deep neural network for the task of learning the inverse error function E_S^{-1} as accurately as possible. To learn E_S^{-1}, the approximating network needs to receive the predicted labels L_{pred} as input and be given the ground-truth labels L as its target. The error reversal model can then learn the error patterns in the predicted labels L_{pred} and what the corresponding ground-truth labels are.

Since the input consists of semantic images and the output is almost identical to the input, convolutional autoencoders are a natural choice for this task. The concept of error reversal modeling is then similar to the idea of denoising autoencoders [74]. Denoising autoencoders are given images with an added noise function such as Gaussian white noise and learn to reconstruct the noise-free image from the noisy version. A common application are camera images, where the denoising autoencoder removes the noise caused by the sensor of the camera. For the proposed approach, the semantic segmentation model is considered to be a source of noise introduced to the ground truth, same as how a camera sensor introduces noise to the depiction of a real-world scene.

The model error of the segmentation model S is a source of noise in the predicted labels L_{pred} that is always present. As a second source of segmentation errors, we also consider the case of compromised input. Specifically, we investigate image compression a common reason for a degradation of image quality. In mobile systems such as autonomous vehicles, compressing the raw image input is often necessary due to the restricted resources available, either to efficiently store the data locally or to send them to the backend for further processing. We therefore also propose error reversal modeling as a method to improve the semantic segmentation of compressed images. In the literature, there are several approaches that optimize the image compression scheme itself to preserve the performance of feature detectors or machine learning models on compressed images [181, 182]. The proposed approach is independent of the compression scheme. It can be applied as a final post-processing step to any segmentation model that is fed with images compressed with an arbitrary compression algorithm. The only requirement is a data set of recorded erroneous predictions of a given model. Then, an error reversal model can be designed as discussed in the next section.

5.2.2 Error Reversal Model Design

We implement the error reversal model as an autoencoder, which we refer to as Error-Reversing Autoencoder (ERA) in the following. The goal of the ERA is to extract features and learn error patterns from the semantic predictions in order to reconstruct the ground-truth semantic labels. Since the input of predicted semantic labels is already a high-level abstraction of the original RGB image, the model does not need to have the same depth and complexity as the baseline semantic segmentation model S. In the literature, there is one comparable work where a post-processing autoencoder is trained with manually degraded semantic maps to reconstruct the ground-truth semantic maps [76]. While their model is a denoising autoencoder and does not perform model-specific error reversal, we follow their basic architecture approach of a convolutional encoder-decoder scheme. Since error reversal is more complex than learning a statistical noise function, we introduce several additions. Most importantly, [76] used fully connected layers as the latent layers of their autoencoder. They used this model for medical imaging with three semantic classes. Since driving data sets contain more classes recorded in a significantly more varied environment, we propose to use a fully convolutional architecture. This way, no spatial information is lost.

The number of downsampling layers in the encoder was determined empirically. Since the semantic input is already highly abstract, increasing the depth too much deteriorates the performance of the ERA. The number of channels needs to be selected according to the number of semantic classes. Data sets such as Cityscapes have 19 semantic classes, meaning at least this amount of channels should be available to not compress the information too early. We propose to use a convolutional layer with 32 channels followed by another convolutional layer with 64 channels as the encoder. The impact of adding further encoder layers or increasing their size will be investigated during the experiments.

The decoder consists of the same number of transposed convolutional layers to upsample the latent feature map to the original resolution and to reconstruct the target ground-truth labels. For all layers, a kernel size of 3 was used. As an activation function, Rectified Linear Unit (ReLU) is used after each layer except for the last. The semantic prediction needs to be one-hot encoded to ensure that all classes have the same spatial distance to each other. The input layer thus takes an image with n channels as input, where n is the number of semantic classes. The resulting architecture is summarized in Table 5.4.

Layer	Channels	Kernel	Stride
Conv2D	n : 32	3×3	2
Conv2D	32 : 64	3×3	2
ConvTranspose2D	64 : 32	3×3	2
Conv2D	32 : 32	3×3	1
ConvTranspose2D	32 : 32	3×3	2
Conv2D	32 : n	3×3	1

Table 5.4 Summary of the proposed fully convolutional encoder-decoder architecture for the implementation of the proposed Error-Reversing Autoencoder (ERA).

To obtain the input for the proposed architecture, the predicted labels L_{pred} are one-hot encoded, meaning every pixel in the semantic image is replaced by an $n \times 1$ vector. This vector consists of zeros except for the i-th element being a 1, where $i \in [0, n-1]$ is the index of the semantic class of the corresponding pixel.

While the one-hot encoded semantic predictions could be directly used as input for the ERA, we propose to additionally use the predicted softmax probabilities P_{pred} generated by S given the input image I. The softmax probabilities P_{pred} encode the confidence of the model in its predictions. While the softmax score does not correspond to a true probability and is prone to overconfidence [10], it does increase the amount of information provided to the ERA. If an area of the predicted labels L_{pred} is incorrect despite having high softmax scores, the ERA can learn about this overconfidence since it has the ground-truth labels L available during training. This is conceptually similar to introspection as introduced in Section 4.1, although now, we aim to predict the correct class instead of the failure probability for each pixel.

To combine the one-hot encoded predicted labels L_{pred} with the predicted softmax scores P_{pred}, we multiply each one-hot vector with the score of the corresponding class. This results in softmax-weighted one-hot encoded predicted semantic maps, which are finally used as input to the ERA. The ERA is then trained to reconstruct the ground-truth labels L from the input. We summarize the entire proposed workflow for training the ERA in Figure 5.11.

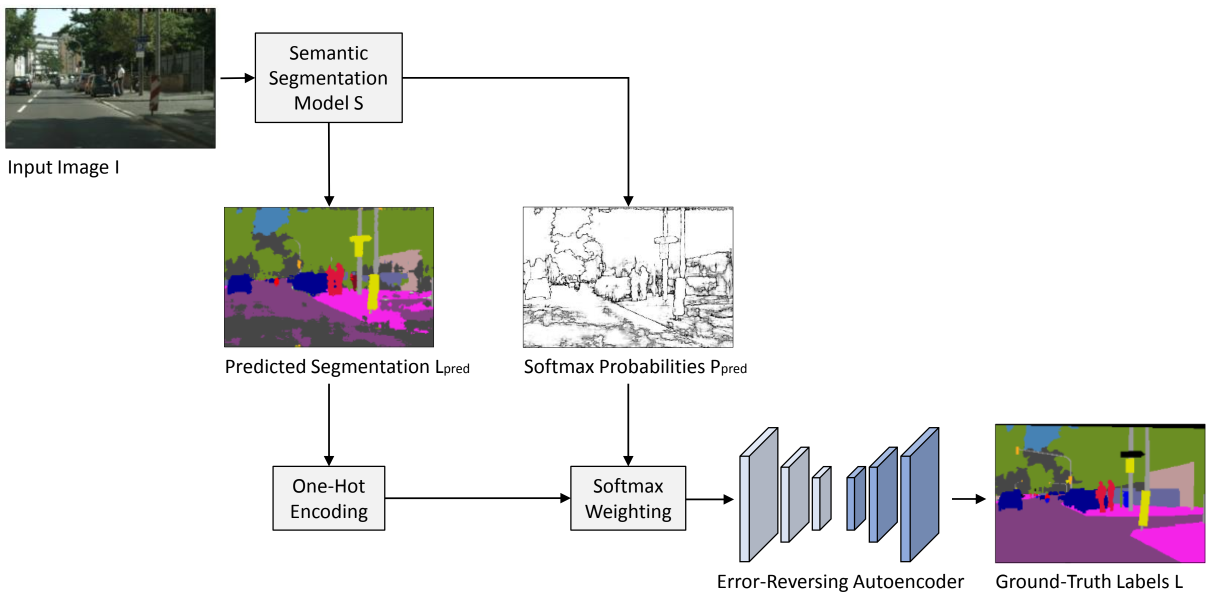

Input Image I
Semantic Segmentation Model S
Predicted Segmentation Lpred
Softmax Probabilities Ppred
One-Hot Encoding
Softmax Weighting
Error-Reversing Autoencoder
Ground-Truth Labels L

Figure 5.11 Overview of the workflow for training the proposed Error-Reversing Autoencoder (ERA).

At a total of six layers, the proposed architecture is very compact and fast to train. We exploit the fact that semantic maps are already abstract features extracted from the original RGB image. The ERA mainly has to learn the shapes of the individual classes and their spatial context. For example, pedestrians have the same morphological structure, even if shadows or noise in the input obscures part of the person in the image. A partially predicted pedestrian in the predicted semantic labels can then be completed into the most likely full human shape by the ERA.

The low complexity of the proposed architecture allows it to be added to an existing semantic segmentation pipeline without adding a significant overhead. Since training is fast as well, it is feasible to train multiple ERAs for different purposes and use the most fitting one depending on the scenario. For example, one ERA can be trained with the semantic predictions for uncompressed images, while additional ERAs can be trained for predictions made on increasingly compressed input. During deployment, the correct ERA can then be selected based on the meta information about the compression of the input image. This is significantly cheaper than training an entire baseline semantic segmentation model for each level of compression, which could be practically infeasible considering the effort required to train state-of-the-art segmentation architectures. Next, we evaluate the performance of the proposed approach.

5.2.3 Results

In this section, we present the results of the experiments conducted to evaluate the proposed reverse error modeling. First, we compare the proposed approach to a state-of-the art post-processing approach for semantic image segmentation and evaluate the design choices of the proposed architecture. Then, we investigate the application of using error reversal on the semantic segmentation of compressed images.

5.2.4 Baseline Comparison

For all subsequent experiments, we use a publicly available implementation of DeepLabV3+ [172] with a MobileNetV2 backbone [183], same as in Chapter 4. DeepLabV3+ is trained on the 2975 training images from Cityscapes. On the test set, it achieves a mean IoU of 72.1 %. We follow similar works on the Cityscapes data sets and downsample all images to a resolution of 256×512 [84, 184, 185]. While the mean IoU is not affected by the downsampling, the training and testing speed is increased significantly.

The training hyperparameters are selected empirically. All autoencoders are trained for 50 epochs at a learning rate of 0.001. If the validation loss did not decrease for 10 epochs, the learning rate is reduced by a factor of 10.

The autoencoders are trained with the semantic maps predicted by the trained DeepLabV3+ on the training set as input and the corresponding ground-truth labels as targets. Similar to introspection, a dedicated set for obtaining the predicted labels L_{pred} would be desirable, but is not feasible for most public data sets due to their limited size. While using the predictions on the training images trains the autoencoders with a slightly different error distribution than they will encounter during testing, being able to reverse the training error is still a useful ability. Any type of error made on the training set is likely to be made on the testing set as well, thus allowing the ERA to correct the predictions on the testing set too.

As a baseline comparison, we use the state-of-the-art approach for semantic segmentation post-processing proposed by Larrazabal et al. [76]. Their work outperformed existing work on postprocessing of predicted semantic maps in the field of medical imaging [186, 187, 75]. They trained a denoising autoencoder, referred to as *postDAE*, to denoise the semantic predictions for chest X-ray images. As a source of noise for the training images for the *postDAE*, they added manual degradations to the ground-truth semantic labels. This includes switching random pixels to other classes, adding and removing random shapes, or dilating and eroding the class borders. While their approach improved the semantic segmentation of medical images, the proposed error reversal approach is intended for the more complex task of segmenting natural images in the driving context. We implement *postDAE* as described in [76]. The architecture of *postDAE* consists of a fully connected latent space. Since manual degradations of the ground-truth labels are used as input, no softmax score is available. We therefore can analyze three main differences of the proposed approach to *postDAE*: the use of the actual erroneous semantic predictions as opposed to manual degradations, the use of a fully convolutional architecture, and the pixel-wise weighting with the softmax score.

Firstly, we investigate the impact of using a fully convolutional architecture instead of fully connected latent layers, while still using manual degradations. We refer to the resulting model as $postDAE_{\text{fulconv}}$. Secondly, we apply the proposed concept of reverse error modeling and use the actual erroneous predictions of the semantic segmentation model as input. We train one model without softmax weighting, referred to as ERA_{hard}, and one model where the softmax score is integrated into the input by weighting the one-hot encoded semantic predictions, referred to as ERA_{soft}.

Finally, we investigate the choices made regarding the size of the proposed architecture for the ERAs. Firstly, we add an additional downsampling layer to the encoder and a corresponding upsampling layer to the encoder. The resulting deeper model is referred to as ERA_{deep}. Secondly, we keep the original number of layers, but increase the channel size by a factor of two. We refer to the resulting larger model as ERA_{large}. After training all autoencoders as outlined above, we apply them to the semantic predictions on the test set obtained from the DeepLabV3+ model. In Table 5.5, we compare the resulting mean IoU of all methods.

Method	**Mean IoU**
postDAE [76]	65.8
$postDAE_{\text{fulconv}}$	70.7
ERA_{hard}	72.1
ERA_{deep}	71.7
ERA_{large}	72.3
ERA_{soft}	**72.6**

Table 5.5 Mean IoU [%] of all evaluated post-processing autoencoder approaches. The proposed compact, fully convolutional model ERA_{soft} trained with both the predicted semantic maps as well as the softmax scores outperforms the state-of-the-art *postDAE* by 6.8 %.

The state-of-the-art *postDAE* performs the worst at a mean IoU of 65.8 %. While it allowed for improvements in the context of medical imaging in [76], the domain of driving is significantly more complex and requires a more elaborate approach. Each of the proposed concepts further improves performance when used for postprocessing the semantic predictions. Using a fully convolutional architecture for $postDAE_{\text{fulconv}}$ increases the performance by 4.9 %. The introduction of reverse error modeling as opposed to using manual degradations again improves the performance, with ERA_{hard} outperforming *postDAE* by 6.3 %. Finally, additionally using softmax weighting for the input achieves the best performance, reaching a mean IoU of 72.6 %. While each proposed change to the state-of-the-art *postDAE* improves performance, only the combination of all proposed concepts manages to outperform the performance of the baseline DeepLabV3+, which achieved a mean IoU of 72.1 %. This is due to DeepLabV3+ already being a highly optimized architecture, requiring a careful ERA design to improve its performance.

Finally, the deeper model ERA_{deep} and the larger model ERA_{large} both perform worse than the architecture proposed in Table 5.4. This indicates that increasing model complexity is not the right approach for improving performance, while making suitable design choices such as introducing the softmax scores or using a fully convolutional architecture does yield improvements.

5.2.5 Error Reversal for Compressed Images

On uncompressed images, the proposed error reversal modeling improves the semantic segmentation performance of DeepLabV3+ to 72.6 %, compared to 65.8 when using *postDAE* [76]. While this is already a useful result, we next apply ERAs to the predicted segmentations of compressed images, where more segmentation errors are expected. Any compression scheme used in practice can be used. We select the popular Joint Photographic Experts Group (JPEG) compression method to generate compressed versions of the training and testing set. To investigate the effect of reverse error modeling on different levels of compression, we compress once at 90 %, once at 50 %, and once at 10 % JPEG quality. For each compression level, we obtain the semantic predictions from DeepLabV3+ and use them to train a dedicated ERA, referred to as ERA_{90}, ERA_{50} and ERA_{10}. The ERA trained with semantic predictions for the uncompressed, original images is referred to as ERA_{orig} in the following. As a final comparison, we train one more ERA with the semantic predictions from both the original images and all three compressed data sets. The resulting ERA_{all} allows to investigate whether training multiple ERAs for different compression levels is beneficial or if a single model trained with all levels suffices.

In Table 5.6, we summarize the performance of all models when applied to all compressed versions of the test set. As baseline comparisons, the performance of the state-of-the-art *postDAE* as well as of the unprocessed DeepLabV3+ model on the original and the compressed images are shown as well.

Method	**Original**	**JPEG90**	**JPEG50**	**JPEG10**	**Average**
DeepLabV3+	72.1	69.3	47.9	11.3	50.2
postDAE	65.8	62.4	43.3	10.3	45.6
ERA_{all}	70.1	68.0	52.3	**19.9**	52.6
ERA_{orig}	**72.6**	70.0	48.1	11.1	50.5
ERA_{90}	72.2	**70.3**	49.3	11.9	50.9
ERA_{50}	70.0	68.8	**55.2**	17.2	**52.8**
ERA_{10}	49.5	48.6	41.7	**19.9**	39.9

Table 5.6 Mean IoU [%] of the proposed post-processing with Error-Reversing Autoencoders (ERAs) trained for varying compression levels, applied to both the original test set and all compressed versions. As a baseline, we compare to the unprocessed DeepLabV3+ and the state-of-the-art *postDAE* [76].

Same as for the uncompressed images, *postDAE* is not capable of improving the performance of the unprocessed DeepLabV3+ for any compression level. For the original, uncompressed testing set as well as for the *JPEG90*, *JPEG50*, and *JPEG10* versions, the ERA trained for that specific compression level achieves the best performance. For uncompressed images, ERA_{orig} improves DeepLabV3+ by 0.5 % and outperforms *postDAE* by 6.8 %. For *JPEG90*, ERA_{90} outperforms the baseline approaches by at least 1.0 %. The mean IoU achieved by ERA_{50} on *JPEG50* is at least 7.3 % larger than the baselines, while ERA_{10} achieves an improvement of 8.6 % or more on *JPEG10*.

On each individual data set, ERA_{all} performs worse than the compression-specific ERAs. When averaging the performance over all compression levels, ERA_{all} performs better than most of the other approaches, but is still outperformed by ERA_{50}. This indicates that even if only a single ERA is available, training a model with a moderate amount of compression is a potential alternative to generating a large data set that contains a range of different compression levels. It should be noted that due to the compact size of the proposed ERA architecture, it is practically feasible to train and store a dedicated ERA for each compression level even on a mobile system such as an autonomous vehicle. Averaging the performance of the best ERA over all compression levels achieves an average mean IoU of 54.5 %, outperforming the original DeepLabV3+ model by 4.3 %.

For a more detailed analysis of the improvements the proposed ERAs offer, we show the change of the mean IoU for each individual class present in the Cityscapes data set in Table 5.7. For each column, the best-performing ERA trained with the data set compressed at the corresponding JPEG quality level is used.

Semantic Class	**Original**	**JPEG 90%**	**JPEG 50%**	**JPEG 10%**
Road	**0.1**	**0.1**	**14.9**	**42.3**
Sidewalk	**0.4**	**0.4**	**8.2**	**5.2**
Building	**0.2**	**0.5**	**23.1**	**23.5**
Wall	**1.9**	**3.9**	**3.7**	−0.3
Fence	**1.0**	**1.5**	**9.1**	**0.4**
Pole	**1.2**	**1.3**	**1.3**	−0.5
Traffic Light	**0.1**	−0.7	**2.8**	**0.3**
Traffic Sign	−0.1	**0.2**	**5.6**	**4.1**
Vegetation	**0.2**	**0.7**	**3.2**	**26.7**
Terrain	**0.7**	**0.9**	**5.9**	**1.8**
Sky	−0.4	−0.3	**5.1**	**19.7**
Person	−0.4	−0.1	**1.7**	**7.0**
Rider	−0.3	−0.9	**1.9**	−0.6
Car	0	−0.1	**9.0**	**29.0**
Truck	**1.8**	**2.4**	**10.5**	−2.8
Bus	**1.5**	**3.3**	**14.7**	−0.5
Train	**2.1**	**3.5**	**10.1**	**3.3**
Motorcycle	**0.8**	**2.2**	**6.0**	−0.8
Bicycle	−0.6	−0.2	**1.7**	**5.5**
Average	**0.5**	**1.0**	**7.3**	**8.6**

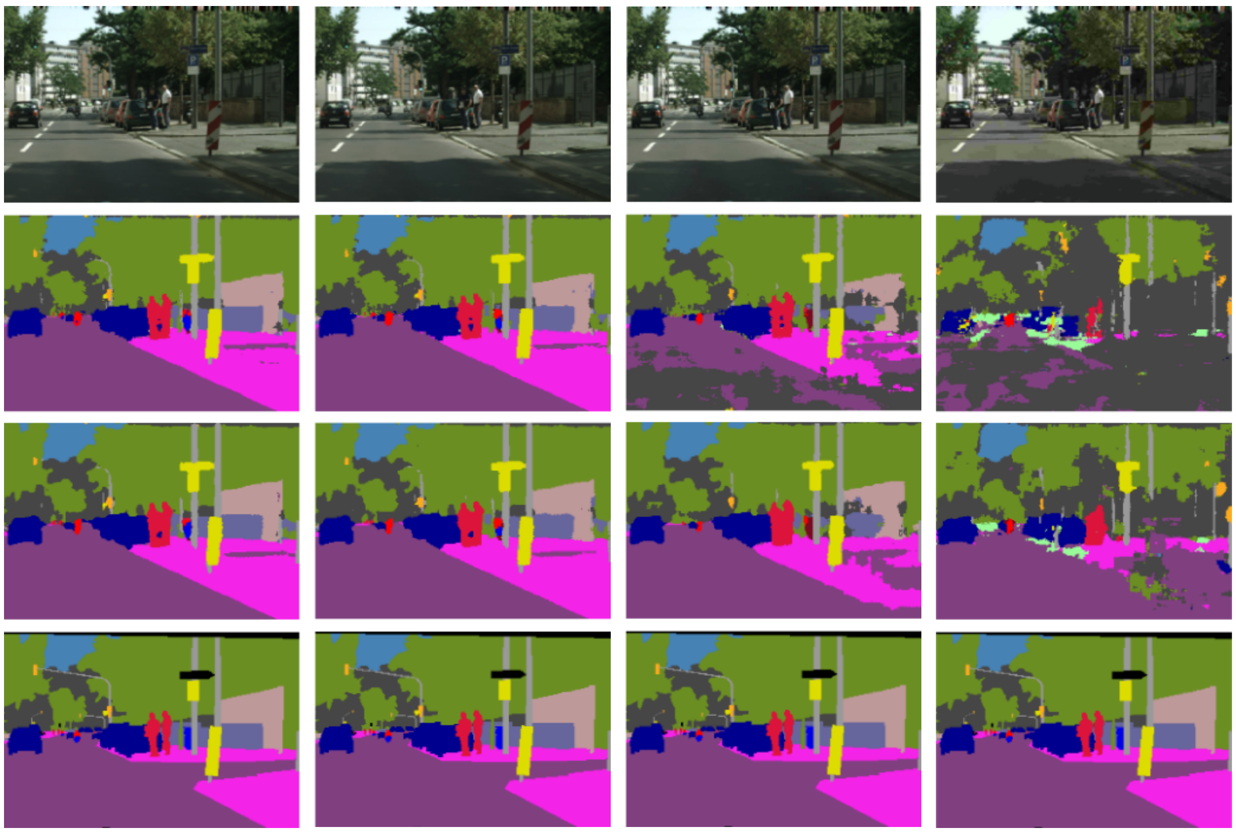
Table 5.7 Class-wise absolute mean IoU change on the original and the compressed test sets after applying the corresponding best-performing autoencoder. Improved values are shown in bold. A larger compression of the input leads to a larger improvement from the error reversal.

For each compression level, the error reversal improves the performance for at least 13 of the 19 classes. The most significant improvements occur for highly structured classes such as *Road*, *Sidewalk*, or *Building*. The visual patterns of those classes are comparatively simple and easier to reconstruct than more dynamic classes. Interestingly, for dynamic classes such as *Pedestrian* and *Bicycle*, the postprocessing only starts improving the mean IoU for higher compression levels. This suggests that the error reversal is not capable of correcting the subtler errors made on the original images, but can restore the overall shape even of smaller and varied classes when the image input was compressed significantly.

Finally, the effect of the proposed error reversal is visualized for a sample test image from Cityscapes in Figure 5.12. The four columns correspond to the original version of the image, the compression at *JPEG90*, the compression at *JPEG50*, and the compression at *JPEG10*. The first row shows the input image, the second row is the semantic prediction by DeepLabV3+ and the third row depicts the output of the corresponding ERA trained for that compression level. The fourth row shows the ground-truth semantic labels.

The increasing compression level significantly impacts the semantic prediction by DeepLabV3+, resulting in a highly spurious output for *JPEG10* where details such as the pedestrian (red) are almost entirely missed. The error reversal consistently improves the semantic prediction, most notably managing to establish the overall structure of the road and buildings even for the highest level of compression. While details such as the pedestrian cannot be completely restored, the few pixels correctly labeled as *Pedestrian* in the prediction by DeepLabV3+ are enough for the ERA to restore the general shape of that class.

For less extreme levels of compression, the visual impact of the error reversal is less significant, but still visible. For the uncompressed image, the semantic prediction by DeepLabV3+ is already highly similar to the ground truth, but minor details such as the spurious errors for the fence on the right of the image are still partially corrected by the ERA.



(a) Original **(b)** JPEG 90 % **(c)** JPEG 50 % **(d)** JPEG 10 %

Figure 5.12 Visualization of the error reversal applied to a sample test image. The columns correspond to the original images (a), images compressed at JPEG 90 % quality (b), JPEG 50 % quality (c), and JPEG 10 % quality (d). The top row shows the input image, the second row depicts the semantic predictions by DeepLabV3+, the third row shows the result of the corresponding ERA, and the fourth row shows the ground truth labels for reference (adapted from [29] © 2022 IEEE).

5.3 Improving Multimodal Object Detection with Individual Sensor Monitoring

As the final contribution of this thesis, we use the concept of introspective failure prediction to improve multimodal object detection. For this, we monitor each individual sensor in a fusion system and predict if it would lead to detection failures. The weighting of the individual sensors in the fusion step is then adjusted based on the predicted performance. Next, we introduce the general concept of the proposed individual sensor monitoring.

5.3.1 Concept

In addition to semantic segmentation, object detection is a core component of perception systems in autonomous driving. An accurate assessment of which objects are present in the environment is important to ensure safe driving. Since the environment of the car is often dynamic and complex, relying only on a single sensor such as a camera is not always sufficient for robust object detection. To address this, different sensors can be combined using multimodal fusion. The goal of multimodal fusion is to combine the strengths of the different sensors, most commonly cameras and Light Detection and Ranging (LIDAR) sensors. Combining multiple sensor modalities in a fusion-based deep neural network is a popular approach for increasing robustness that achieves state-of-the-art detection results [59].

In object detection models based on deep learning, the sensor inputs can be fused at various stages of the network architecture. Regardless of whether late or early fusion is used, the fusion is usually done by concatenating or adding the feature vectors extracted from each type of input [21, 64]. While the fusion is performed to combine the strengths of multiple sensors, it is not always beneficial to use all available sensors. If the input from an individual sensor is occluded, ambiguous, or compromised in some way, the features extracted from that input should not be used to the same extent as the other available sensors. The typical fusion by concatenation or addition in the state of the art does not offer any way of reducing the influence of problematic sensor input. Instead, the fusion network is implicitly expected to use the best features from the different inputs. We propose to improve multimodal object detection by explicitly predicting if an individual sensor is generating problematic input and to then reduce its weight in the fusion step accordingly. In addition to the fusion combining the different strengths of all sensors, the proposed system also becomes aware of the different weaknesses of each sensor.

To achieve this, we implement an introspective monitoring model M for each sensor. For each sensor, this requires removing all other sensors from the fusion-based architecture and to then train the architecture as a single-sensor object detection model. This allows to record the object detection performance of the given architecture when only the current individual sensor is used. Each monitoring model M is then trained with the recorded object detection performance of using only that single sensor. Thus, the introspective model learns to predict the performance of each individual sensor based only on the raw sensor input. If a sensor input is predicted to result in a low object detection performance, the features extracted from that input will be reduced accordingly in the fusion step.

The concept of monitoring each sensor is comparable to a watchdog system [91]. While a watchdog system monitors if a sensor is still physically functioning, the proposed system monitors if a sensor is still useful. In this work, we focus on fusion of an RGB camera and a LIDAR sensor. Thus, we implement two sensor monitoring models, M_{RGB} and M_{LIDAR}. The camera and the LIDAR features are weighted with the performance predicted by M_{RGB} and M_{LIDAR}. The region proposals and classification taking place in the subsequent object detection network is then based on the weighted fused feature vector. In Figure 5.13, we visualize the general approach.

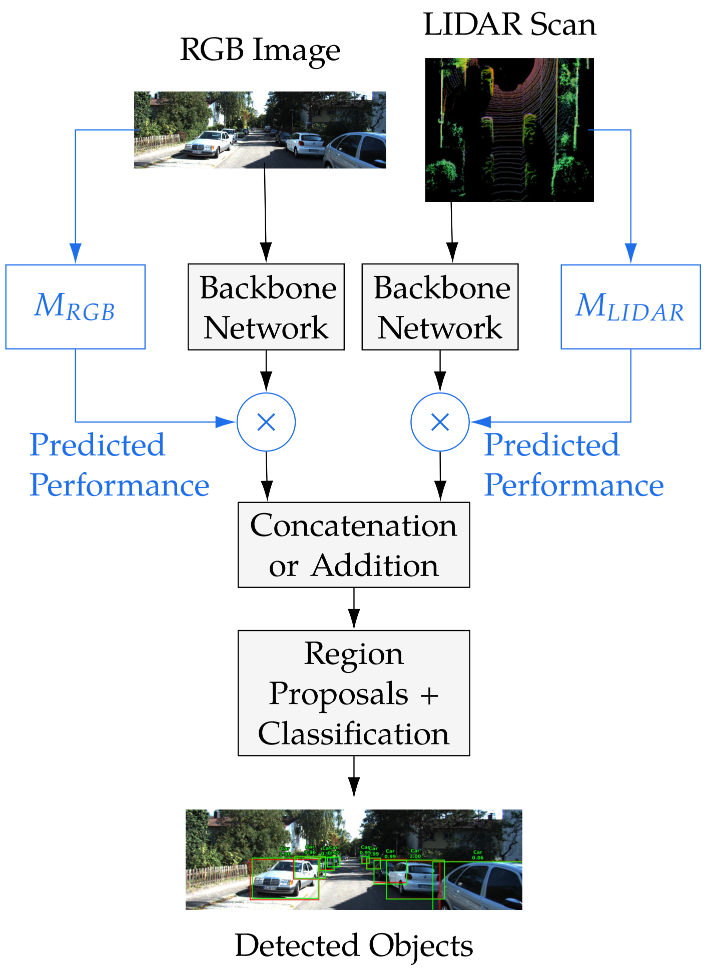


Figure 5.13 Overview of the proposed individual sensor monitoring (blue) that can be added to any standard multimodal object detection architecture (gray). For each sensor, an introspective model M is trained to predict the performance of that sensor. In the fusion by concatenation or addition, the features extracted from each sensor input are weighted accordingly to improve the subsequent object detection (adopted from [28] © 2022 IEEE).

5.3.2 Single Sensor Performance Prediction

Next, the design of the sensor monitoring models M_i that predict the performance of the individual sensors S_i is discussed in more detail. Given a system with n sensors, each sensor $S_i, i \in [1, n]$, requires its own monitoring model M_i. The goal is to be able to predict the object detection performance P_i of each sensor S_i individually. A single-sensor object detection network D_i trained only with the sensor readings R_i obtained with the sensor S_i is required for each sensor. For this, the available multimodal model D_{fusion} can be used. We propose to remove all sensors S_j, $j \neq i$, from the multimodal network D_{fusion} to obtain the single-sensor detector D_i. This has the advantage of not requiring additional implementation effort to create D_i. It also ensures that the detection performance for the individual sensor S_i is as similar to its performance in the fusion network D_{fusion} as possible, since the architecture of the backbone, region proposal, and classification parts of the network are identical in D_i and D_{fusion}. Only the amount of input streams is changed from n to 1.

The single-sensor detection models D_i allow recording the detection performance P_i associated to each individual sensor S_i. Next, we discuss two approaches for implementing the sensor monitoring models M_i that generate the predicted detection performance $P_{i,pred}$. First, an approach based on the confidence score of D_i is discussed. Then, an introspective approach that operates directly on the raw sensor readings R_i is introduced.

5.3.2.1 Confidence Score

The confidence score associated with the prediction made by an object detection architecture D_i is a readily available, straightforward estimate of the performance P_i of the model D_i. While the specific computation of the confidence score can vary depending on the selected architecture, a confidence score is typically available without any additional implementation. In object detection, each predicted object is assigned its own score. To obtain an overall performance prediction $P_{pred,conf}$, we average the confidence scores from all detected objects. The performance prediction model $M_{i,conf}$ is then simply the average confidence score of the output of D_i. While this is straightforward to implement, the entire single-sensor network D_i needs to be evaluated to obtain a predicted performance for the sensor S_i. Since the proposed sensor monitoring is intended for systems with multiple sensors, this would require running multiple additional object detection frameworks only to obtain the weights for the fusion of the actual object detector D_{fusion}. In mobile systems such as autonomous vehicles, a less resource-demanding approach is desirable. Ideally, the performance should be predicted based on the sensor reading R_i directly. For this, we next use the concept of introspection as a significantly faster alternative.

5.3.2.2 Introspection

In Chapter 3, we used introspection to predict system-level failures. In Chapter 4, we used it to predict pixel-wise misclassifications. Now, we apply the concept to object detection to predict the performance of individual sensors. First, a training set of recorded performances is required to obtain the target labels for the introspective model. For this, we run each single-sensor model D_i on the training set and obtain the detection performance P_i for each sensor reading R_i. Any metric can be used to quantify the performance. Here, we use the mean IoU as a metric for the performance P_i. Then, each sensor input R_i can be assigned a new, introspective label to obtain a training set for an introspective model $M_{i,intro}$. The scalar values of P_i could be used as labels directly, resulting in a regression task for the introspective model. Preliminary results showed that implementing introspection as a regression performs significantly worse than implementing it as a classification task, however. Regressing the performance P_i from the raw sensor reading R_i tended to result in mean reversal. Inputs that led to a very high or low performance were not sufficiently detected by a regression approach. Thus, we implement $M_{i,intro}$ as a binary classification model. Same as in Chapter 4, we define a *Failure* and a *Success* class. Each sensor reading R_i is assigned to one of those two classes depending on how high its mean IoU is. As a threshold, we use the average of all mean IoUs on the training set. This results in a data set consisting of sensor inputs R_i and binary class labels. Since R_i is either an RGB image or a LIDAR point cloud that can be projected into 2D, any state-of-the-art image classification network can be used to implement the introspective model $M_{i,intro}$. Same as in Chapter 3, we use the ResNet50 architecture [43] to implement the required binary image classifier. For each sensor S_i, a ResNet50 architecture is trained to classify each reading R_i as either *Success* or *Failure*. To obtain a continuous value, we use the softmax score associated to the class *Success* as the predicted performance $P_{i,pred}$.

The introspective approach operates directly on the raw sensor data and does not require to run any object detection model before generating a prediction. This also makes introspection less prone to overconfidence, which is an inherent issue of using the model-based confidence score as a performance predictor. Instead of using the output of a potentially overconfident model, introspection only uses the visual features in the input R_i to predict the performance of the corresponding sensor S_i. If no challenging or problematic features are present, the introspective model ideally assigns the input a success probability of 1, in which case the monitoring does not affect the fusion at all. However, if any features are present that resemble features that have led to low mean IoU values during training, the introspective model will generate a lower success probability and the problematic visual features will be given a reduced weight during the subsequent fusion, with D_{fusion} then relying more on the other available sensors. The implementation of how a given fusion network is dynamically adjusted using the proposed sensor monitoring models is presented next.

5.3.3 Dynamically Adjustable Fusion

The goal of the proposed sensor monitoring is to improve a given fusion-based object detection system. This is achieved by making the fusion dynamically adjustable, so that the influence of problematic sensor input is reduced. In this section, we discuss how the monitoring models M_i can be integrated into the architecture of a multimodal object detector. First, we discuss the proposed approach in general and then show an implementation based on a state-of-the-art architecture.

Given a fusion-based object detector D_{fusion} with n sensors S_i, $i \in [1, n]$, we first generate n single-sensor detection models D_i by removing all sensors except for S_i from the input layer. The performance P_i for each sensor reading R_i is recorded and used to generate binary introspective labels for R_i, which are then used to train the introspective model $M_{i,intro}$. As a more expensive alternative, the confidence score of D_i can be used to obtain $M_{i,conf}$.

For training the fusion-based model D_{fusion}, the sensor monitoring models are not yet used. They are only required during testing, when the actual single-sensor performance P_i is not available. During training, the ground-truth performance values P_i are used as weights. It depends on the architecture of D_{fusion} where to incorporate the weighting. In general terms, D_{fusion} extracts features F_i from each sensor reading R_i. The features F_i from all sensors S_i are then combined. F_i can be either low-level features in case of an early fusion architecture, or high-level features in case late fusion is employed. The proposed system can be applied regardless of which fusion strategy is used.

The individual features F_i are eventually combined into fused features F_{fused}. While many fusion operations are possible, the two most common choices in the literature are either addition or concatenation. F_{fused} can thus be calculated as

$$F_{fused} = \sum_{i=1}^{n} F_i \qquad \text{or} \tag{5.4}$$

$$F_{fused} = (F_1, \ldots, F_n). \tag{5.5}$$

At this step, we introduce the single-sensor detection performances P_i as weights for the corresponding features F_i. During testing, the performances P_i are not available and need to be predicted using the introspective model $M_{i,intro}$ to calculate the fused features. We therefore refer to the weighted combination of all features F_i as F_{intro}, calculated as

$$F_{intro} = \frac{1}{\sum_{i=1}^{n} P_i} \sum_{i=1}^{n} P_i F_i \qquad \text{or} \tag{5.6}$$

$$F_{intro} = \frac{1}{\sum_{i=1}^{n} P_i} (P_i F_1, \ldots, P_n F_n). \tag{5.7}$$

This way, D_{fusion} learns to use the optimal weighted combination of the features F_i during training. During testing, the actual performance P_i is substituted with the predicted performance $P_{i,pred}$. Either $M_{i,intro}$ or $M_{i,conf}$ can be used for obtaining the predicted weights. The specific architecture of D_i and the shape of the features F_i depend directly on the chosen fusion-based architecture. While the proposed concept is generally applicable, we next visualize it integrated into a specific state-of-the-art architecture. We use Aggregate View Object Detection (AVOD) [21], a fusion-based object detection network with competitive results. It is based on camera and LIDAR input, thus we have $n = 2$ sensors. The fusion is performed following Equation 5.6. In Figure 5.14, we show an overview of the AVOD architecture together with the proposed individual sensor monitoring. In state-of-the-art models such as AVOD, the different sensor inputs are only implicitly weighted based on the values of the neurons learned during training. The neuron weights are fixed once training is complete, meaning the fusion cannot be adjusted to the input anymore. The proposed individual sensor monitoring changes the fusion for every new input, allowing to predict and use the best weighted combination of the different sensor inputs based directly on the current input data.

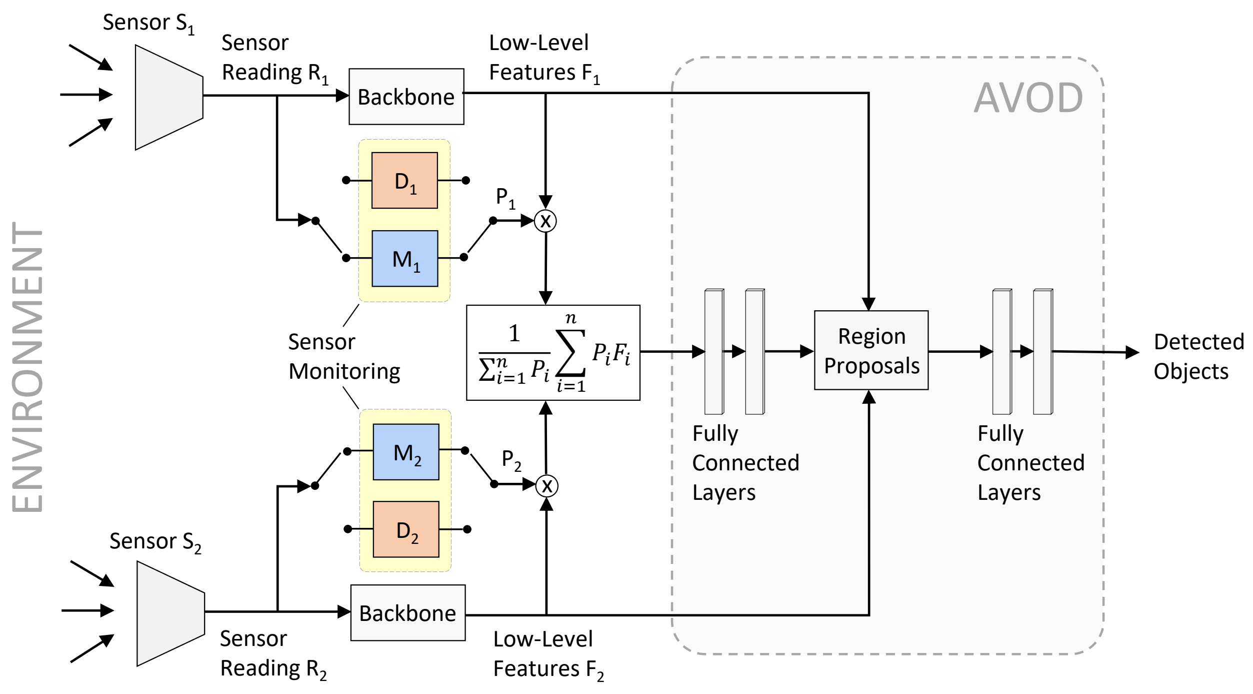


Figure 5.14 Overview of the proposed dynamically adjustable fusion with $n = 2$ sensors, applied to the AVOD architecture [21]. The features F_i extracted from the sensor readings R_i are weighted with the performances P_i obtained from the single-sensor models D_i. During testing, the performances are predicted using sensor monitoring models M_i (adopted from [28] © 2022 IEEE).

5.3.4 Results

In this section, we present the results of the experiments conducted to evaluate the proposed approach. First, we introduce the setup of our experiments. Then, we discuss the performance of the proposed single-sensor performance prediction models. Next, we use the performance prediction models for the multimodal object detection with individual sensor monitoring and compare the performance to the baseline of AVOD without sensor monitoring. Finally, we investigate the capability of the proposed approach to detect and handle the most challenging inputs.

5.3.4.1 Experimental Setup

The proposed approach is designed to improve a given fusion-based object detection model. We apply it to the AVOD architecture, which has shown state-of-the-art performance at feasible inference times [21]. Any other object detector based on two or more sensors could be used as well. We use a public implementation of AVOD [188] trained on the KITTI data set [82] as the baseline for all experiments. KITTI contains over 15 000 RGB images as well as LIDAR scans recorded on urban German roads. From that data, 7481 images and LIDAR scans come with publicly available bounding box annotations for object detection. Annotations of the classes *Car*, *Pedestrian*, and *Cyclist* are available. While some related work trained a dedicated model for each class [21], we train AVOD with all three classes at once to obtain a real-time capable system. We use 80 % of the labeled 7481 samples for training and the remaining 20 % for testing, following other works on KITTI [64, 58]. The official recommendations are used for all training parameters. Both the mean IoU and the mean Average Precision (mAP) as suggested by the authors of AVOD will be used as metrics for the evaluation of the object detection task. The performance prediction is evaluated using a Receiver Operating Characteristic (ROC) curve analysis, same as in Chapter 3. A Tesla T4 GPU from Google Colab is used to train and evaluate all models.

5.3.4.2 Sensor Performance Prediction

The two sensors used by AVOD are a camera and a LIDAR scanner. Thus, we require a camera-based and a LIDAR-based performance prediction model to monitor the two sensors. We implement both introspection and the approach based on the confidence score. First, the single-sensor AVOD models are trained. This is done by removing the LIDAR input for $AVOD_{RGB}$ and by removing the RGB image input to obtain $AVOD_{LIDAR}$. After reducing the architecture to a single-sensor model, both models are then trained in the same way as the fusion-based model. $AVOD_{RGB}$ achieves a mean IoU of 67.6 % on the test camera images, while $AVOD_{LIDAR}$ achieves a mean IoU of 73.3 % based on the LIDAR scans. $M_{RGB,conf}$ is then the confidence score averaged over all objects detected by $AVOD_{RGB}$, while $M_{LIDAR,conf}$ uses the average confidence score from $AVOD_{LIDAR}$.

To obtain training data for the introspective models $M_{RGB,intro}$ and $M_{LIDAR,intro}$, the mean IoU values obtained on the training samples are used. For both the camera-based and the LIDAR-based monitoring model, the average performance on the training set is used to classify each sample as *Failure* or *Success*. For the camera images, 21.5 % are below that threshold and thus labeled as *Failure*. For the LIDAR scans, 27.1 % are labeled as *Failure*. For both sensors, a ResNet50 architecture is trained to classify each training sample. For the LIDAR data, the scans are first projected to Bird's Eye View (BEV). Both introspective classifiers are trained for 100 epochs at a learning rate of 0.0001. We use the Adam optimizer to minimize the binary cross entropy loss.

Both the confidence-based and the introspective approach for performance prediction are evaluated on the test set using an ROC curve analysis. For the introspective models, the softmax score associated to the class *Success* is used as a continuous value for the performance prediction. The resulting Area Under Curve (AUC) values for both camera and LIDAR data are summarized in Table 5.8.

	Camera	**LIDAR**
Introspection	0.88	0.66
Confidence Score	0.96	0.83

Table 5.8 The AUC values of the two proposed performance prediction approaches for both sensors.

The confidence score achieves a better performance prediction than the introspective models. This indicates that the output of the model itself is more informative than only the raw input data, which is the only input for introspection. While the AUC values of introspection are lower, the absolute results of 0.88 and 0.66 for camera and LIDAR are still sufficiently high to be useful in practice. In contrast, the confidence score is very expensive to obtain since an entire object detection network needs to be run. For introspection, only a single classification is needed.

For both approaches, the performance prediction of the camera-based detection model performs better than for the LIDAR-based object detector. This can be explained by the better detection performance of $AVOD_{LIDAR}$, which outperforms the camera-based detector by 5.7 % in mean IoU. Thus, there are fewer significant failure patterns to learn from. This corresponds to the observation made in Chapter 4 that the worse the baseline model performs, the larger the improvements achieved by introspection become.

5.3.4.3 Multimodal Object Detection with Sensor Monitoring

Now, the sensor monitoring models are integrated into the fusion-based AVOD architecture. During training, the actual performances for each individual sensor are used to weight the fusion as specified in Equation 5.6. As shown in Figure 5.14, the weights are applied in the early fusion of the features, before generating region proposals. During testing, the weights required for the fusion are obtained using the sensor monitoring models. We refer to the fusion model using the confidence score as

			AP_{3D} [%]			AP_{BEV} [%]		
Method	Runtime	Class	Easy	Moderate	Hard	Easy	Moderate	Hard
AVOD	0.37 s	Car	80.56	70.59	61.87	89.45	72.24	71.27
		Pedestrian	61.71	54.40	47.90	65.05	57.14	55.35
		Cyclist	67.24	48.86	48.58	67.24	49.26	48.90
		mAP	**69.84**	**57.95**	**52.78**	**73.91**	**59.55**	**58.51**
introAVOD	0.43 s	Car	88.09	79.14	77.66	90.40	88.47	80.66
		Pedestrian	62.43	54.58	47.75	65.75	57.17	55.37
		Cyclist	65.06	47.56	46.03	65.27	47.70	46.34
		mAP	**71.86**	**60.43**	**57.15**	**73.80**	**64.45**	**60.79**
confAVOD	1.07 s	Car	88.63	85.35	77.98	90.44	89.36	88.15
		Pedestrian	62.76	54.91	52.93	70.19	61.87	55.22
		Cyclist	66.80	49.34	48.97	66.59	55.35	50.16
		mAP	**72.73**	**63.20**	**59.96**	**75.74**	**68.86**	**64.51**

Table 5.9 Summary of the object detection performance on KITTI for the baseline AVOD, the *introAVOD*, and the *confAVOD*. The average precision for both *3D* and BEV detection is shown for all three difficulty categories. Averaged over all categories, both proposed sensor monitoring approaches improve the 3D detection performance by at least 5.0 % and the *BEV* detection performance by at least 3.7 %.

confAVOD. When the weighting is done using the introspective prediction, we refer to the fusion model as *introAVOD*. As a baseline, we also train a regular AVOD without any weighting.

We summarize the results for all models on the 20 % of KITTI samples reserved for testing in Table 5.9. The evaluation on KITTI is typically done for three difficulty categories, which are "easy", "moderate" and "hard". The categories correspond to different minimum sizes and maximum levels of occlusion for the objects considered during the evaluation. AVOD offers both *3D* and BEV detection. Table 5.9 contains the mAP of both types of detection for the three classes *Car*, *Pedestrian*, and *Cyclist*. The mAP over all classes is also shown. Finally, we report the average runtime per inference of the baseline AVOD, the *introAVOD* and the *confAVOD*.

The two proposed dynamically adjustable fusion models *confAVOD* and *introAVOD* both significantly outperform the baseline AVOD model. For *3D* object detection, *introAVOD* improves the performance averaged over all three difficult categories by 5.0 %. For the BEV detection, the performance gains are 3.7 %. When using the confidence scores as the sensor monitoring models, the resulting *confAVOD* improves *3D* detection by 8.5 % and BEV detection by 8.9 %, again averaged over all difficulty categories. The larger performance gains are in line with the better performance of the confidence-based performance prediction models shown in Table 5.8.

For the individual classes, the proposed models achieve improvements for most of them. The proposed *introAVOD* shows the largest improvements for the class *Car*. The *Pedestrian* class is also improved slightly, whereas the detection performance of the class *Cyclist* is slightly decreased. This can be explained by the typically small size of cyclists in the sensor readings. Such objects are challenging to detect in the sparse LIDAR point clouds. However, the overall higher performance of the LIDAR-based object detector results in the LIDAR features receiving a larger weight during fusion. This results indicates that training performance prediction models for each individual class instead of the average of all classes could be a useful extension of the evaluated approach.

While *introAVOD* mostly improves the classes *Car* and *Pedestrian*, the second proposed approach of *confAVOD* also improves the baseline for the *Cyclist* class for most difficulty categories. While the performance of *confAVOD* is better than *introAVOD* in almost every regard, it comes at the critical cost of a much higher computational complexity. The average inference time of *confAVOD* is 1.07 s per input, which is almost three times as slow as the baseline AVOD at 0.37 s. In contrast, the *introAVOD* approach only requires 0.43 s per inference, an almost negligible increase of 0.06 s compared to the baseline. This is explained by *confAVOD* requiring to run three entire object detection frameworks, whereas *introAVOD* only introduces two additional classifications. The two proposed sensor monitoring approaches therefore offer a trade-off between inference speed and detection performance. The *confAVOD* approach achieves the largest improvement of up to 8.9 % at the cost of an almost tripled inference time, while *introAVOD* still achieves a gain of up to 5.0 % at an inference time increase of just 16 %.

5.3.4.4 Challenging Input Analysis

The proposed approach was designed with the goal of enabling a fusion system to be dynamically adjusted when faced with problematic input. To investigate whether the proposed sensor monitoring accurately detects the most challenging sensor readings and whether it is capable of successfully adapting the fusion, we next focus on the performance for the inputs predicted to be the most challenging. Due to the excessive inference time of *confAVOD*, we focus only on the practically more feasible *introAVOD* in this section.

For each sensor, we obtain the samples where the corresponding introspective model predicted a performance of less than 0.5. Among the camera images, $M_{RGB,intro}$ predicts such a low performance for a total of 284 images. For the LIDAR scans, $M_{LIDAR,intro}$ generates a predicted performance $P_{intro,LIDAR}$ of less than 0.5 for 171 input samples. Considering the overall better performance of the LIDAR-only model compared to the camera-only model, the predicted number of challenging LIDAR inputs to be lower was expected. For a straightforward scalar comparison, we use the mean IoU to evaluate all models on the 284 challenging camera images and the 171 challenging LIDAR scans, respectively. The results for the single-sensor models, the baseline AVOD and *introAVOD* are shown in Table 5.10.

Model	**Test Set**	$\mathbf{P_{intro,RGB} < 0.5}$	$\mathbf{P_{intro,LIDAR} < 0.5}$
AVOD$_{\mathbf{RGB}}$	67.6	16.4	42.5
AVOD$_{\mathbf{LIDAR}}$	73.3	57.8	43.3
AVOD	78.7	63.4	57.1
introAVOD	**82.3**	**67.2**	**66.8**

Table 5.10 Mean IoU [%] of the single-sensor models, the baseline AVOD and the proposed *introAVOD* approach, evaluated on the samples with the lowest predicted performance as well as on the entire test set for comparison.

As expected, both fusion-based models significantly outperform the single-sensor models regardless of the input. More interestingly, the performance of both single-sensor models significantly drops when evaluated on the samples with a predicted low performance. Specifically, the mean IoU of $AVOD_{LIDAR}$ on the LIDAR scans with a low predicted performance drops by 40.9 %. For $AVOD_{camera}$, the performance on the images with a low predicted score drops even further, by 75.7 %. The introspective models were thus capable of accurately predicting which input samples would lead to a low single-sensor performance.

Another interesting observation is that the samples leading to a reduced performance for one sensor also result in a lower performance for the other sensor. $AVOD_{RGB}$ evaluated on images where the corresponding LIDAR scan received a low prediction dropped by 37.1 % in performance compared to the test set. $AVOD_{LIDAR}$ evaluated on LIDAR scans where the corresponding images are predicted to be problematic dropped by 21.1 %. This demonstrates a correlation between the performance of the two separate sensors. If one sensor reading is challenging, the other on is likely challenging as well, for example due to the presence of occluded objects that neither sensor is capable of detecting.

Despite the observed correlation, the performance drop is much smaller for single-sensor models evaluated on samples where the other sensor is predicted to be problematic. This indicates that the sensor readings are still complementary to each other. This fact is exploited by the proposed dynamically adjustable fusion, where the fusion network places a higher weight on the sensor with a higher individual performance. While the proposed *introAVOD* already outperforms the baseline AVOD by 4.6 % on the entire test set, the improvements of the sensor monitoring on challenging inputs increase even further. For inputs where the camera image was predicted to be challenging, the gain increases to 6.0 %. When the LIDAR scan is predicted to be problematic, *introAVOD* outperforms the baseline by a significant 17.0 %. These findings indicate that the overall improvements introduced by the sensor monitoring largely occur for problematic sensor input. This is in line with the fact that for inputs with a high predicted performance, the sensor monitoring does not change the fusion. For pristine input, AVOD and *introAVOD* are effectively identical when the predicted weights are close to one. Only when the input is problematic does the sensor monitoring change the fusion, resulting in the observed gains of up to 17.0 %.

5.4 Chapter Summary

In this chapter, we proposed several applications based on the insights from the failure prediction approaches developed in Chapter 3 and Chapter 4. We developed three approaches for improving either the semantic segmentation or the object detection module of an autonomous vehicle.

First, we designed a two-stage visual perception system inspired by the way humans perceive complex scenes. The first stage corresponds to state-of-the-art semantic image segmentation. Then, pixel-wise introspective failure prediction is used as the basis for a failure region detection system. We reclassify the regions detected this way with a local road participant classifier and propose a method for updating the failure pixels corresponding to the predicted dynamic object. Despite an overall segmentation accuracy of 94 % in the first stage, the proposed second stage still allows to recover initially misclassified road participants in over one out of twenty test images.

Next, we used recorded erroneous predictions as input to train a model to directly correct pixel-wise errors instead of only predicting the pixel-wise failure probabilities. For this, we proposed the concept of Error-Reversing Autoencoder (ERA). A fully convolutional autoencoder is trained to reconstruct the ground-truth labels from the semantic prediction generated by a given semantic segmentation model. This approach improves the mean IoU of the state-of-the-art DeepLabV3+ model on Cityscapes by 0.5 %. As a second application, we trained ERAs with semantic predictions made on compressed images. For high levels of compression, the proposed approach improves the mean IoU by 8.6 %, allowing to restore the general semantic structure of the scene even for highly compromised initial semantic predictions.

Finally, we applied the concept of introspection to multimodal object detection. For each individual sensor, we trained a sensor monitoring model to predict the detection performance of just that sensor. Then, we trained a fusion-based network, where the fusion of the camera and LIDAR input is weighted with the corresponding single-sensor performance. During testing, the two sensor inputs are weighted with the predicted single-sensor performance, allowing to place a lower weight on problematic sensor input with a low predicted performance. This approach improves the state-of-the-art AVOD model by at least 3.7 % on the KITTI data set. For inputs predicted to be the most challenging, the improvement increases to up to 17 %, demonstrating that the proposed sensor monitoring approach is capable of detecting problematic input and can adjust the fusion accordingly.

6 Conclusion

Failures in autonomous systems are inevitable. In autonomous driving, such failures can have safety-critical consequences. The concept of failure prediction is an important potential answer to this challenge. Despite the benefits accurate failure prediction can offer, the topic has received comparatively little attention over the last decade. Most notably, the challenge of predicting failures seconds in advance is largely unaddressed in the literature. In this thesis, we thoroughly analyzed the concept of failures in autonomous systems and reviewed the state of the art in failure prediction. Based on this analysis, the failure prediction framework shown in Figure 1.2 was designed and a range of failure prediction approaches was proposed. The overall objective is to detect as many failures as possible, both automatically and predictively. To conclude this thesis, we summarize the key findings, outline the major limitations, and suggest directions for future work.

6.1 Summary

The entire pipeline of a typical autonomous system was considered for the design and implementation of automatic and predictive failure detection for autonomous driving. Depending on the stage of such a pipeline, different types of failures can be predicted. We distinguished two main ways of approaching failure prediction, a system-level and a component-level approach.

To predict failures on a system level, we propose a black box failure prediction approach. We use the concept of introspection, which is based on learning from previously recorded failures. We define disengagements as system-level failures, since human safety drivers will usually prevent actual crashes. Disengagements refer to both the system giving up control automatically and to the human safety driver intervening. We use over 2500 disengagements recorded by BMW development vehicles on public urban roads. We use the ten seconds leading up to each disengagement as the failure data for all approaches. To obtain a complete data set, we also sample the same number of successful sequences containing ten seconds of undisrupted driving. We use this basic framework to implement four concrete failure prediction models. Firstly, we use sequences of the state data of the car to predict failures. The car state includes the speed, acceleration, steering angle, and angular velocity. We split up the ten seconds of each sequence into shorter three second samples. Each sample is labeled as either *Failure* or *Success* depending on whether the corresponding sequence ended in a disengagement or not. Then, an LSTM-based classifier is proposed which learns to classify a new three second sample. Secondly, we apply the same concept to image data. Three second samples of video data are reduced to a single dynamic difference image consisting of three consecutive difference images. Then, a CNN-based classifier is trained to classify the current video sample as *Failure* or *Success*. As a third source of information about impending failures, the trajectories planned by the vehicle are used. We again split the ten second sequences into multiple three second samples, each one containing thirty consecutive planned trajectories. We propose an LSTM-based classifier to learn to assign a failure probability to a new sequence of planned trajectories. Finally, we combine the state-based, image-based, and trajectory-based approaches in a fusion-based model. We use late fusion and average the individual failure probabilities. We compare to two state-of-the-art disengagement and crash prediction approaches from the literature [16, 15]. While each individual method outperforms the state of the art already, the proposed fusion model outperforms the best-performing reference approach by over 25 %. Additionally, it detects failures two seconds earlier than the earliest existing approach. Seven seconds in advance, the introspective fusion model detects over 86 % of all failures.

While predicting system-level failures is beneficial for ensuring the safety of an autonomous system, it does not give any insights into what caused the failure. As the second main approach to failure prediction in autonomous driving, we propose component-level failure prediction methods. We introduce a pixel-wise failure prediction approach for semantic segmentation, again using the concept of introspection. Given an arbitrary segmentation model, we record pixel-wise failures and use the resulting pixel-wise errors as new targets to train an introspective model. We implement the introspective model as a semantic segmentation model, allowing for reusing the trained encoder from the baseline segmentation model. By explicitly learning from recorded failures, the proposed approach significantly outperforms the output-based state of the art, which relies on general uncertainty estimation [10, 12]. Combining introspection with the state of the art further improved the overall performance, indicating that the proposed method predicts complementary types of failures. Next, we extended this approach with convolutional LSTMs to take video sequences as input. This allows the introspective model to learn from spatio-temporal features, outperforming single-image introspection by over 4 % in a precision-recall analysis. Finally, we exploit the temporal information available in videos to predict future pixel-wise failures. We apply concepts from the field of future semantic map prediction, using a spatio-temporal architecture to predict a future error map given a sequence of the previous four predicted error maps. Since public video data sets with pixel-wise annotations are limited in size, we introduce the Densely Annotated Video Driving (DAVID) data set. At around 11 000 images, DAVID is nine times the size of the largest public real-world data set [19]. The proposed approach allows for accurately predicting failures up to five frames into the future, establishing the first existing approach for predicting future failures of semantic segmentation.

Finally, we used concepts from failure prediction to correct the system instead of just predicting its failures. We propose two methods to improve semantic segmentation and one method to improve object detection. Firstly, we use the *Zoom Lens Model* employed in human vision as inspiration to design a two-stage approach to visual scene perception. The first stage is global semantic segmentation. In the proposed second stage, we use pixel-wise failure prediction as the basis for a failure region detection pipeline. We combine Region of Interest (ROI) filtering, smoothing, and clustering to obtain distinct failure regions. Then, a failure region classifier is trained to reclassify each failure region. Each failure pixel from the Class Activation Map (CAM) of the corresponding classification is updated to the predicted class. Applied to a semantic segmentation model that achieved a test accuracy of 94.0 % on 4000 test images, this approach recovers at least 10 % of the pixels of over 200 misclassified road participants. Secondly, we propose to directly correct the failures of semantic segmentation on a pixel-wise level by using reverse error modeling. We reframe a semantic segmentation model as an error function applied to the ground-truth labels. Then, we propose Error-Reversing Autoencoders (ERAs) in order to approximate the corresponding reverse error function, which allows to reconstruct the ground truth from the semantic predictions of the given segmentation model. We propose reverse error modeling both as a way of refining an arbitrary segmentation model and as a way of significantly improving the performance when given compressed images. By training a dedicated ERA for each expected compression level, the mean IoU of the segmentation of highly compressed images can be increased by over 8 %. For both the original and the compressed images, we compare to the state of the art in post-processing semantic predictions [75, 76]. We outperform the best-performing reference method by at least 6 %. Thirdly, we use introspective failure prediction to improve fusion-based object detection. For each individual sensor, we train a single-sensor detection model, record the corresponding performance, and then train an introspective monitoring model. During fusion, we predict the performance of each individual sensor using the monitoring models and weight the fusion of the different sensors with the predicted performance. This allows the fusion-based detector to rely more on inputs predicted to achieve a high performance. Applied to the state-of-the-art Aggregate View Object Detection (AVOD) architecture [21], the proposed sensor monitoring improves the mean IoU by almost 4 %. For inputs where the individual sensor performance is predicted to be lowest, the monitoring improves the performance by up to 17 %. This demonstrates its capabilities of detecting problematic input and adjusting the fusion accordingly.

6.2 Limitations

We compared all proposed concepts to existing state-of-the-art methods to validate their contribution towards the goal of accurate failure prediction. While all methods proposed in this thesis significantly outperformed existing approaches or were the first to address the respective problem, they still have some limitations both in performance and applicability.

The introspective disengagement prediction framework proposed in Chapter 3 requires extensive test drives where disengagements are encountered before a model can be trained. While we argue that such test drives are necessary for the development of an autonomous vehicle anyway, requiring months of driving in advance means that introspection cannot be used right away for this purpose. In this thesis, we were able to train with thousands of disengagements due to having access to the data from the BMW research fleet. For projects that have less hardware available, it can be difficult to collect data on a similar scale to then obtain models with a similar performance.

Regarding the pixel-wise introspective failure prediction approach for semantic segmentation introduced in Chapter 4, an important limitation is that only failures based on visual patterns seen in the training set can be detected. This is problematic for rare or novel classes that are not sufficiently present in the training set. The output-based state-of-the-art methods we compared to our model showed better performance for rare classes where few samples were available during training. In most classes and on average, introspection outperformed these output-based methods. Nevertheless, this limitation is inherent to the concept of introspection and needs to be addressed, for example by combining introspection with an output-based approach. Furthermore, the approach for predicting future pixel-wise failures introduced in Section 4.3 requires intense labeling efforts since every frame in the training videos needs semantic labels. While we introduced a large-scale data set generated in a simulator to validate the potential of this approach, deployment in real-world applications comes with the requirement of costly frame-wise labeling.

The two-stage approach for detecting and correcting failure regions inspired by the human *Zoom Lens Model* is based on pixel-wise introspective failure prediction and therefore shares its limitations. Additionally, the failure detection pipeline proposed in Section 5.1 consists of multiple consecutive processing steps, many of which require manual tuning. Several thresholds need to be selected, such as for the binarization, the ROI masks, and the CAM filtering. While the values used in this thesis follow intuitive reasoning, they make it challenging to apply this concept out of the box. The overall complexity is also an issue. While the proposed second stage requires mostly two classifications and an ROI prediction, the increased complexity can be problematic on restricted embedded systems.

Regarding the concept of ERAs proposed in Section 5.2, the main limitations are again related to the underlying concept of introspection. An ERA can only learn to reverse the error function observed on the training set. The unknown error function the semantic segmentation model will apply on unseen data cannot be learned directly. Thus, the ERA can only reconstruct a subset of the errors on the test set.

Finally, the individual sensor monitoring introduced in Section 5.3 is mostly limited by the current use of a single performance weight per sensor. While the overall performance of the LIDAR tended to be better than the camera, the LIDAR-based detection was occasionally worse for objects with small bounding boxes such as bicycles. Nevertheless, the overall performance was often correctly predicted to be higher and the LIDAR input was assigned a higher weight. This sometimes resulted in a decreased performance for small objects. While the improvements averaged over all bounding boxes were significant, a more elaborate weighting that considers individual classes or the properties of individual bounding boxes is required to avoid occasional performance drops.

6.3 Future Work

The goal of this thesis was to investigate the topic of failure prediction for autonomous systems and to develop a framework capable of preemptively detecting as many failures as possible. While each of the proposed methods achieved state-of-the-art performance and can already be used to predict a range of potential failures during driving, there is still room for further research. Next, we outline ideas for improving the proposed system-level and component-level failure prediction approaches. Additionally, we suggest Out-of-Distribution (OOD) detection and a combination of system-level and component-level approaches as further directions of future work.

System-level failure prediction The disengagement prediction system developed in Chapter 3 achieved an accuracy of over 86 % seven seconds in advance. Due to the inherent randomness of some failures such as sudden moves by pedestrians, further improving the system's accuracy is challenging. However, one option would be to relax the black box assumption and use more information about the environment from the internal scene perception of the vehicle. When treating the car as a black box, the introspective system needs to extract all required semantic information from the camera images itself. Supplying the disengagement prediction system with object lists or predicted movements of traffic participants could lead to an introspective system that is both more aware of the environment and also more aware of the current beliefs of the inspected system. Both factors could be beneficial for predicting a system-level failure of the inspected system.

Besides improving the accuracy, there are several other directions in which this research can be continued. It is possible to distinguish between failures caused by the system automatically returning control to human and failures where the human safety driver took over control. The latter are more safety-critical, since the vehicle would not have stopped driving without human intervention. Being able to predict such failures is highly relevant for Level 3 and above. Additionally, while the applications proposed in Chapter 5 mostly rely on component-level failure prediction, system-level failure prediction could also be applied for more than just generating takeover requests. An interesting future work would be to run the proposed disengagement prediction system in the vehicle and switch to a more cautious secondary driving function when the predicted failure probability crosses a threshold. The seven seconds of time to prepare once a failure is detected would allow for enough time to adjust the system to avoid the failure in the first place. The introspective model can monitor the system adjustments, allowing for a feedback loop to ensure that the adjustments reduce the failure probability.

Component-level failure prediction For the component-level failure prediction, the proposed introspective approach outperformed the state of the art while also being the first approach for predicting future misclassifications of semantic segmentation. Nevertheless, the main limitation of only being able to predict previously seen mistakes as outlined in Section 6.2 should be addressed in future work. For rare or novel visual patterns, an additional approach besides introspective failure prediction is required. Combining introspection with pixel-wise novelty detectors such as [150] could lead to a more comprehensive pixel-wise failure prediction.

The applications of the component-level failure prediction proposed in Chapter 5 are another direction for future work. The proposed individual sensor monitoring from Section 5.3 could also be applied to semantic segmentation. LIDAR-based semantic segmentation has shown state-of-the-art performance [73]. The point-wise introspective failure prediction could then be used to decide which sensor modality to use for segmentation. Applying the proposed introspective approach to a LIDAR-based system would also serve to validate the approach for other sensor modalities, since the experiments conducted for this thesis were performed mostly on camera data.

Instead of monitoring and weighting the current sensor input, another potential direction is to use the failure prediction to decide if new sensor input is required. The *Zoom Lens Model* used as inspiration for the two-stage approach in Section 5.1 could be implemented in a more literal way, by using a dedicated zoom lens to obtain new sensor input specifically for the regions with the highest failure probability. Such a physical zoom lens model could also be combined with the future failure prediction for semantic segmentation introduced in Section 4.3. The 0.5 seconds that pixel-wise failures can be predicted in advance can be enough to physically adjust a sensor and ensure that more detailed information is obtained for problematic areas.

OOD detection In this thesis, we mostly implemented failure predictions for failures caused by epistemic or aleatoric uncertainty. The system-level failure prediction approach from Chapter 3 can detect some novel failures by detecting irregularities in the physical state of the car, which happen regardless of whether the failure is due to model uncertainty, input uncertainty, or OOD data. However, if the system is not aware that it is entering a new environment, the state might not reflect this until it is too late. A promising addition to the proposed disengagement prediction system could be a dedicated OOD detection system. Existing options such as the ReSonAte framework [147] could be combined with introspective models. Similarly, the component-level failure prediction approaches all rely on having recorded failures to learn from. Pixel-wise failures caused by novel objects are challenging to detect this way. OOD detection for individual components such as semantic segmentation could be a solution for this limitation of introspection.

A different direction for future work would be to use OOD detection methods, but use them explicitly for failure prediction. Samples that lead to failures could be classified as OOD and samples that achieve a good performance could be considered in-distribution. Then, any OOD detection approach that relies on learning a representation of a given distribution could be applied. If a new environment is predicted to be in-distribution, this would imply a good system performance. A detected OOD sample would directly indicate a system failure. Existing OOD detection systems typically only determine if an input resembles the in-distribution data used during training, regardless of the performance the system exhibits for the input. A more task-specific interpretation of OOD detection is therefore another interesting direction for improving failure prediction.

Combined failure prediction framework Finally, all failure prediction approaches proposed in this thesis could be combined and implemented in an actual vehicle to evaluate the overall failure prediction capabilities. As outlined in Figure 1.2, the failure probabilities predicted by each of the proposed modules can be propagated throughout the system and aggregated to ultimately decide if a takeover request is necessary. In this thesis, we evaluated system-level and component-level failure prediction approaches separately. For future work, implementing a combined failure prediction framework that aggregates every individual probability would be a promising step towards further improving failure prediction for autonomous systems.

List of Figures

List of Tables

List of Publications

Journal Articles

[25] Christopher B. Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Introspective failure prediction for autonomous driving using late fusion of state and camera information". In: *IEEE Transactions on Intelligent Transportation Systems* (Dec. 2020). Early access, pp. 1–15. DOI: 10.1109/TITS.2020.3044813.

[198] Markus Hofbauer, Christopher B. Kuhn, Goran Petrovic, and Eckehard Steinbach. "Preprocessor Rate Control for Adaptive Multi-View Live Video Streaming Using a Single Encoder". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2022), pp. 1–16. DOI: 10.1109/TCSVT.2022.3142403.

Conference Proceedings

[22] Christopher B. Kuhn, Hofbauer Markus, Sungkyu Lee, Goran Petrovic, and Eckehard Steinbach. "Introspective failure prediction for semantic image segmentation". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece, Sept. 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294308.

[23] Christopher B. Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Introspective black box failure prediction for autonomous driving". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, Oct. 2020, pp. 1907–1913. DOI: 10.1109/IV47402.2020.9304844.

[24] Christopher B. Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Better look twice - Improving visual scene perception using a two-stage approach". In: *2020 IEEE International Symposium on Multimedia (ISM)*. Naples, Italy, Dec. 2020, pp. 33–40. DOI: 10.1109/ISM.2020.00013.

[26] Christopher B. Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Trajectory-based failure prediction for autonomous driving". In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. Nagoya, Japan, July 2021, pp. 980–986. DOI: 10.1109/IV48863.2021.9575937.

[27] Christopher B. Kuhn, Markus Hofbauer, Ziqin Xu, Goran Petrovic, and Eckehard Steinbach. "Pixel-wise failure prediction for semantic video segmentation". In: *2021 IEEE International Conference on Image Processing (ICIP)*. Anchorage, AK, USA, Sept. 2021, pp. 614–618. DOI: 110.1109/ICIP42928.2021.9506552.

[28] Christopher B Kuhn, Markus Hofbauer, Bowen Ma, Goran Petrovic, and Eckehard Steinbach. "Improving Multimodal Object Detection with Individual Sensor Monitoring". In: *24th IEEE International Symposium on Multimedia (ISM)*. Naples, Italy, 2022.

[29] Christopher B. Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Reverse Error Modeling for Improved Semantic Segmentation". In: *2022 IEEE International Conference on Image Processing (ICIP)*. 2022, pp. 106–110. DOI: 10.1109/ICIP46576.2022.9897331.

[100] Markus Hofbauer, Christoph Bachhuber, Christopher Kuhn, and Eckehard Steinbach. "Teaching Software Engineering As Programming Over Time". In: *IEEE/ACM 4th International Workshop on Software Engineering Education for the Next Generation*. Pittsburgh, PA, USA: Association for Computing Machinery, May 2022, pp. 1–8. ISBN: 978-1-4503-9336-2/22/05. DOI: 10.1145/3528231.3528353. URL: https://doi.org/10.1145/3528231.3528353.

[189] Shreyas Ramakrishna, Baiting Luo, Christopher B. Kuhn, Gabor Karsai, and Abhishek Dubey. "ANTI-CARLA: An adversarial testing framework for autonomous vehicles in CARLA". In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. Submitted for review. Macau, China, Oct. 2022.

[190] Markus Hofbauer, Christopher Kuhn, Goran Petrovic, and Eckehard Steinbach. "Measuring the Influence of Image Preprocessing on the Encoder Rate-Distortion Performance". In: *24th IEEE International Symposium on Multimedia (ISM)*. Naples, Italy, 2022.

[191] Markus Hofbauer, Christoph Bachhuber, Christopher Kuhn, Sebastian Schwarz, Bart Kroon, and Eckehard Steinbach. "Large-Scale Collaborative Writing: Technical Challenges and Recommendations". In: 2023, Submitted for review.

[192] Lukas Habermayr, Markus Hofbauer, Joao-Vitor Zacchi, and Christopher B. Kuhn. "Situation-aware model refinement for semantic image segmentation". In: *2021 IEEE 24th International Conference on Intelligent Transportation Systems (ITSC)*. Indianapolis, IN, USA, Sept. 2021, pp. 2696–2702. DOI: 10.1109/ITSC48978.2021.9564549.

[193] Markus Hofbauer, Christopher B. Kuhn, Goran Petrovic, and Eckehard Steinbach. "TELECARLA: An open source extension of the CARLA simulator for teleoperated driving research using off-the-shelf components". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, Oct. 2020, pp. 335–340. DOI: 10.1109/IV47402.2020.9304676.

[194] Markus Hofbauer, Christopher B. Kuhn, Jiaming Meng, Goran Petrovic, and Eckehard Steinbach. "Multi-view region of interest prediction for autonomous driving using semi-supervised labeling". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece, Sept. 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294387.

[195] Markus Hofbauer, Christopher B. Kuhn, Goran Petrovic, and Eckehard Steinbach. "Adaptive multi-view live video streaming for teledriving using a single hardware encoder". In: *2020 IEEE International Symposium on Multimedia (ISM)*. Naples, Italy, Dec. 2020, pp. 9–16. DOI: 10.1109/ISM.2020.00008.

[196] Markus Hofbauer, Christoph Bachhuber, Christopher B. Kuhn, and Eckehard Steinbach. "Teaching software engineering as programming over time". In: *44th International Conference on Software Engineering (ICSE 2022)*. Pittsburgh, PA, USA, May 2022.

[197] Markus Hofbauer, Christopher Kuhn, Mariem Khlifi, Goran Petrovic, and Eckehard Steinbach. "Traffic-Aware Multi-View Video Stream Adaptation for Teleoperated Driving". In: *2022 IEEE 95th Vehicular Technology Conference: VTC2022-Spring*. Helsinki, Finland, June 2022, pp. 1–8. DOI: 10.1109/VTC2022-Spring54318.2022.9860513.

[199] Kristian Fischer, Markus Hofbauer, Christopher Kuhn, Eckehard Steinbach, and Andre Kaup. "Evaluation of Video Coding for Machines without Ground Truth". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 1616–1620. DOI: 10.1109/ICASSP43922.2022.9747633.

Bibliography

[1] Hendrik Unger, Tobias Markert, and Egon Müller. "Evaluation of use cases of autonomous mobile robots in factory environments". In: *Procedia Manufacturing* 17 (2018), pp. 254–261.

[2] Junyao Guo, Unmesh Kurup, and Mohak Shah. "Is it Safe to Drive? An Overview of Factors, Metrics, and Datasets for Driveability Assessment in Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems (ITSC)* 21.8 (2020), pp. 3135–3151.

[3] Marco Galvani. "History and future of driver assistance". In: *IEEE Instrumentation & Measurement Magazine* 22.1 (2019), pp. 11–16.

[4] Vinayak V. Dixit, Sai Chand, and Divya J. Nair. "Autonomous vehicles: disengagements, accidents and reaction times". In: *PLoS one* 11.12 (2016), e0168054.

[5] Puneet Kohli and Anjali Chadha. "Enabling pedestrian safety using computer vision techniques: A case study of the 2018 uber inc. self-driving car crash". In: *Future of Information and Communication Conference*. Springer. 2019, pp. 261–279.

[6] Samuel Gibbs. "Google sibling waymo launches fully autonomous ride-hailing service". In: *The Guardian* 7 (2017).

[7] Natasha Merat, A. Hamish Jamson, Frank C. H. Lai, Michael Daly, and Oliver M. J. Carsten. "Transition to manual: Driver behaviour when resuming control from a highly automated vehicle". In: *Transportation Research Part F: Traffic Psychology and Behaviour*. Vehicle automation and driver behaviour 27 (Nov. 2014), pp. 274–282.

[8] Armen Der Kiureghian and Ove Ditlevsen. "Aleatory or epistemic? Does it matter?" In: *Structural Safety* 31.2 (2009), pp. 105–112.

[9] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. "Generalized out-of-distribution detection: a survey". In: *arXiv preprint arXiv:2110.11334* (2021). (Visited on 03/15/2022).

[10] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *International Conference on Machine Learning*. 2016, pp. 1050–1059.

[11] Yonatan Geifman and Ran El-Yaniv. "Selective classification for deep neural networks". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 4878–4887.

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 6402–6413.

[13] Hao Wang and Dit-Yan Yeung. "Towards Bayesian deep learning: A framework and some existing methods". In: *IEEE Transactions on Knowledge and Data Engineering* 28.12 (2016), pp. 3395–3408.

[14] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. "Weight uncertainty in neural networks". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. Vol. 37. 2015, pp. 1613–1622.

[15] Lex Fridman, Li Ding, Benedikt Jenik, and Bryan Reimer. "Arguing machines: Human supervision of black box AI systems that make life-critical decisions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[16] Rhiannon Michelmore, Marta Kwiatkowska, and Yarin Gal. “Evaluating uncertainty quantification in end-to-end autonomous driving control”. In: *arXiv preprint arXiv:1811.06817* (2018). (Visited on 03/15/2022).

[17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. “CARLA: An open urban driving simulator”. In: *Conference on Robot Learning*. 2017, pp. 1–16.

[18] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. “Semantic object classes in video: A high-definition ground truth database”. In: *Pattern Recognition Letters* 30.2 (2009), pp. 88–97.

[19] Byungju Kim, Junho Yim, and Junmo Kim. “Highway driving dataset for semantic video segmentation”. In: *29th British Machine Vision Conference (BMVC 2018)*. 2018.

[20] Charles W. Eriksen and James D. St James. “Visual attention within and around the field of focal attention: A zoom lens model”. In: *Perception & Psychophysics* 40.4 (1986), pp. 225–240.

[21] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. “Joint 3D proposal generation and object detection from view aggregation”. In: *arXiv preprint arXiv:1712.02294* (July 2018). (Visited on 03/15/2022).

[30] Guna Seetharaman, Arun Lakhotia, and Erik Philip Blasch. “Unmanned vehicles come of age: The DARPA grand challenge”. In: *Computer* 39.12 (2006), pp. 26–29.

[31] Shantanu Ingle and Madhuri Phute. “Tesla autopilot: semi autonomous driving, an uptick for future autonomy”. In: *International Research Journal of Engineering and Technology* 3.9 (2016), pp. 369–372.

[32] Shaoshan Liu, Liyun Li, Jie Tang, Shuang Wu, and Jean-Luc Gaudiot. “Creating autonomous vehicle systems”. In: *Synthesis Lectures on Computer Science* 8.2 (2020), pp. i–216.

[33] BMW Group PressClub, ed. *Die neue BMW Group High Performance D3 Plattform*. Mar. 2019. URL: www.press.bmwgroup.com/deutschland/article/detail/T0293764DE (visited on 03/15/2022).

[34] Mariusz Bojarski et al. “End to end learning for self-driving cars”. In: *arXiv preprint arXiv: 1604.07316* (2016). (Visited on 03/15/2022).

[35] Junsung Kim, Ragunathan Rajkumar, and Markus Jochim. “Towards dependable autonomous driving vehicles: a system-level approach”. In: *ACM SIGBED Review* 10.1 (2013), pp. 29–32.

[36] Rowan McAllister et al. “Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization, Aug. 2017, pp. 4745–4753.

[37] Christian Pek and Matthias Althoff. “Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 1447–1454.

[38] Kunihiko Fukushima. “Neocognitron: A hierarchical neural network capable of visual pattern recognition”. In: *Neural Networks* 1.2 (1988), pp. 119–130.

[39] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural Computation* 1.4 (1989), pp. 541–551.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90.

[41] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[42] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014). (Visited on 03/15/2022).

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[44] Andrew G. Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017). (Visited on 03/15/2022).

[45] Robert Hecht-Nielsen. “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[46] Diogo V. Carvalho, Eduardo M Pereira, and Jaime S Cardoso. “Machine learning interpretability: A survey on methods and metrics”. In: *Electronics* 8.8 (2019), p. 832.

[47] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587.

[48] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[49] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in Neural Information Processing Systems* 28 (2015).

[50] Yongri Piao, Yongyao Jiang, Miao Zhang, Jian Wang, and Huchuan Lu. “PANet: Patch-aware network for light field salient object detection”. In: *IEEE Transactions on Cybernetics* (2021).

[51] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You Only Look Once: Unified, real-time object detection”. In: *arXiv preprint arXiv:1506.02640* (May 2016). (Visited on 03/15/2022).

[52] Joseph Redmon and Ali Farhadi. “YOLOv3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (Apr. 2018). (Visited on 03/15/2022).

[53] Adam Kortylewski, Qing Liu, Angtian Wang, Yihong Sun, and Alan Yuille. “Compositional convolutional neural networks: A robust and interpretable model for object recognition under occlusion”. In: *arXiv preprint arXiv:2006.15538* (June 2020). (Visited on 03/15/2022).

[54] Angtian Wang, Yihong Sun, Adam Kortylewski, and Alan L. Yuille. “Robust object detection under occlusion with context-aware compositionalnets”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12645–12654.

[55] Zining Wang et al. “Inferring spatial uncertainty in object detection”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 5792–5799.

[56] Nir Morgulis, Alexander Kreines, Shachar Mendelowitz, and Yuval Weisglass. “Fooling a real car with adversarial traffic signs”. In: *arXiv preprint arXiv:1907.00374* (2019). (Visited on 03/15/2022).

[57] Haichao Zhang and Jianyu Wang. “Towards adversarially robust object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 421–430.

[58] Martin Simon et al. “Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[59] Jamil Fayyad, Mohammad A. Jaradat, Dominique Gruyer, and Homayoun Najjaran. “Deep learning sensor fusion for autonomous vehicle perception and localization: a review”. In: *Sensors* 20.15 (July 2020), p. 4220.

[60] Kuan Liu, Yanen Li, Ning Xu, and Prem Natarajan. “Learn to combine modalities in multimodal deep learning”. In: *arXiv preprint arXiv:1805.11730* (2018). (Visited on 03/15/2022).

[61] Konrad Gadzicki, Razieh Khamsehashari, and Christoph Zetzsche. "Early vs late fusion in multimodal convolutional neural networks". In: *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. 2020, pp. 1–6.

[62] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. "Early versus late fusion in semantic video analysis". In: *Proceedings of the 13th annual ACM international Conference on Multimedia*. 2005, pp. 399–402.

[63] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. "Deep continuous fusion for multi-sensor 3D object detection". In: *arXiv preprint arXiv:2012.10992* (Dec. 2020). (Visited on 03/15/2022).

[64] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. "Multi-view 3D object detection network for autonomous driving". In: *arXiv preprint arXiv:1611.07759* (June 2017). (Visited on 03/15/2022).

[65] Andreas Pfeuffer and Klaus Dietmayer. "Optimal sensor data fusion architecture for object detection in adverse weather conditions". In: *2018 21st International Conference on Information Fusion (FUSION)*. Cambridge: IEEE, July 2018, pp. 1–8.

[66] Oier Mees, Andreas Eitel, and Wolfram Burgard. "Choosing smartly: Adaptive multimodal fusion for object detection in changing environments". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, Oct. 2016, pp. 151–156.

[67] Taewan Kim and Joydeep Ghosh. "On single source robustness in deep fusion models". In: *arXiv preprint arXiv:1906.04691* (Oct. 2019). (Visited on 03/15/2022).

[68] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[69] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495.

[70] Fisher Yu and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122* (2015). (Visited on 03/15/2022).

[71] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2017), pp. 834–848.

[72] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 801–818.

[73] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. "Rangenet++: Fast and accurate lidar semantic segmentation". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4213–4220.

[74] Lovedeep Gondara. "Medical image denoising using convolutional denoising autoencoders". In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. 2016, pp. 241–246.

[75] Agostina J. Larrazabal, Cesar Martinez, and Enzo Ferrante. "Anatomical priors for image segmentation via post-processing with denoising autoencoders". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 585–593.

[76] Agostina J. Larrazabal, Cesar Martinez, Ben Glocker, and Enzo Ferrante. "Post-dae: Anatomically plausible segmentation via post-processing with denoising autoencoders". In: *IEEE Transactions on Medical Imaging* 39.12 (2020), pp. 3813–3820.

[77] Andreas Pfeuffer, Karina Schulz, and Klaus Dietmayer. "Semantic segmentation of video sequences with convolutional lstms". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1441–1447.

[78] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in Neural Information Processing Systems* 28 (2015).

[79] Yong-Hoon Kwon and Min-Gyu Park. "Predicting future frames using retrospective cycle gan". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1811–1820.

[80] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. "Predicting deeper into the future of semantic segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 648–657.

[81] Mrigank Rochan et al. "Future semantic segmentation with convolutional lstm". In: *arXiv preprint arXiv:1807.07946* (2018). (Visited on 03/15/2022).

[82] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Providence, RI: IEEE, June 2012, pp. 3354–3361.

[83] Jens Behley et al. "Semantickitti: A dataset for semantic scene understanding of lidar sequences". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9297–9307.

[84] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3213–3223.

[85] Fisher Yu et al. "Bdd100k: A diverse driving dataset for heterogeneous multitask learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2636–2645.

[86] Jakob Geyer et al. *A2D2: AEV autonomous driving dataset*. http://www.a2d2.audi. 2019. (Visited on 03/15/2022).

[87] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. "DeepDriving: Learning affordance for direct perception in autonomous driving". In: *arXiv preprint arXiv:1505.00256* (May 2015). (Visited on 03/15/2022).

[88] Xiangyu Yue, Bichen Wu, Sanjit A Seshia, Kurt Keutzer, and Alberto L Sangiovanni-Vincentelli. "A lidar point cloud generator: from a virtual world to autonomous driving". In: *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. 2018, pp. 458–464.

[89] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. "Playing for data: Ground truth from computer games". In: *European Conference on Computer Vision*. Springer. 2016, pp. 102–118.

[90] Pei Sun et al. "Scalability in perception for autonomous driving: Waymo open dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2446–2454.

[91] Lei Huang and Lixiang Liu. "Extended watchdog mechanism for wireless sensor networks". In: *Journal of Information and Computing Science* 3.1 (2008), pp. 39–48.

[92] Andreas Theissler, Judith Pérez-Velázquez, Marcel Kettelgerdes, and Gordon Elger. "Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry". In: *Reliability Engineering & System Safety* 215 (2021), p. 107864.

[93] Hyunsun Choi, Eric Jang, and Alexander A. Alemi. "WAIC, but why? Generative ensembles for robust anomaly detection". In: *arXiv preprint arXiv:1810.01392* (Oct. 2018).

[94] Dan Hendrycks and Kevin Gimpel. “A baseline for detecting misclassified and out-of-distribution examples in neural networks”. In: *arXiv preprint arXiv:1610.02136* (Oct. 2016). (Visited on 01/16/2019).

[95] Lei Kang, Wei Zhao, Bozhao Qi, and Suman Banerjee. “Augmenting self-driving with remote control: Challenges and directions”. In: *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*. 2018, pp. 19–24.

[96] Jean-Michael Georg and Frank Diermeyer. “An adaptable and immersive real time interface for resolving system limitations of automated vehicles with teleoperation”. In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. 2019, pp. 2659–2664.

[97] Amin Hosseini, Florian Richthammer, and Markus Lienkamp. “Predictive haptic feedback for safe lateral control of teleoperated road vehicles in urban areas”. In: *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. Nanjing, China: IEEE, May 2016, pp. 1–7.

[98] Eric T. Greenlee, Patricia R. DeLucia, and David C. Newton. “Driver vigilance in automated vehicles: Hazard detection failures are a matter of time”. In: *Human Factors* 60.4 (2018), pp. 465–476.

[99] Ye Xia, Danqing Zhang, Jinkyu Kim, Ken Nakayama, Karl Zipser, and David Whitney. “Predicting driver attention in critical situations”. In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 658–674.

[101] Mariusz Bojarski et al. “Visualbackprop: Efficient visualization of cnns for autonomous driving”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1–8.

[102] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. “Learning deep features for discriminative localization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2921–2929.

[103] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 618–626.

[104] Hamidur Rahman, Shahina Begum, and Mobyen Uddin Ahmed. “Driver Monitoring in the context of autonomous vehicle”. In: *SCAI*. 2015, pp. 108–117.

[105] Nicolas D. Herzberger, Lutz Eckstein, and Maximilian Schwalm. “Detection of missing takeover capability by the orientation reaction to a takeover request”. In: *Aachen Colloquium Automobile and Engine Technology, Aachen*. 2018, pp. 1231–1240.

[106] Fabio Arnez, Huascar Espinoza, Ansgar Radermacher, and François Terrier. “A comparison of uncertainty estimation approaches in deep learning components for autonomous vehicle applications”. In: *arXiv preprint arXiv:2006.15172* (2020). (Visited on 03/15/2022).

[107] Zoubin Ghahramani. “Probabilistic machine learning and artificial intelligence”. en. In: *Nature* 521.7553 (May 2015), pp. 452–459. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14541. URL: http://www.nature.com/articles/nature14541 (visited on 01/16/2019).

[108] Alex Kendall and Yarin Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems*. 2017, pp. 5574–5584.

[109] Pei Wang and Nuno Vasconcelos. “Towards realistic predictors”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 36–51.

[110] Donggeun Yoo and In So Kweon. “Learning loss for active learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 93–102.

[111] Shreyansh Daftry, Sam Zeng, Andrew Bagnell, and Martial Hebert. “Introspective perception: Learning to predict failures in vision systems”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1743–1750.

[112] Peng Zhang, Jiuling Wang, Ali Farhadi, Martial Hebert, and Devi Parikh. “Predicting failures of vision systems”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3566–3573.

[113] Dhruv Mauria Saxena, Vince Kurtz, and Martial Hebert. “Learning robust failure response for autonomous vision based flight”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 5824–5829.

[114] Sadegh Rabiee and Joydeep Biswas. “iVOA: Introspective vision for obstacle avoidance”. In: *arXiv preprint arXiv:1903.01028* (Mar. 2019). (Visited on 03/15/2022).

[115] Qinghua Yang, Hui Chen, Zhe Chen, and Junzhe Su. “Introspective false negative prediction for black-box object detectors in autonomous driving”. In: *Sensors* 21.8 (2021), p. 2819.

[116] Hugo Grimmett, Rudolph Triebel, Rohan Paul, and Ingmar Posner. “Introspective classification for robot perception”. In: *The International Journal of Robotics Research* 35.7 (June 2016), pp. 743–762.

[117] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. “On calibration of modern neural networks”. In: *arXiv preprint arXiv:1706.04599* (June 2017). (Visited on 03/15/2022).

[118] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. “Accurate uncertainties for deep learning using calibrated regression”. In: *arXiv preprint arXiv:1807.00263* (June 2018). (Visited on 03/15/2022).

[119] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[120] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. “Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding”. In: *28th British Machine Vision Conference 2017 (BMVC 2017)*. 2017.

[121] Meire Fortunato, Charles Blundell, and Oriol Vinyals. “Bayesian recurrent neural networks”. In: *arXiv preprint arXiv:1704.02798* (Apr. 2017). (Visited on 03/15/2022).

[122] Mattias Teye, Hossein Azizpour, and Kevin Smith. “Bayesian uncertainty estimation for batch normalized deep networks”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4907–4916.

[123] Alex Kendall, Yarin Gal, and Roberto Cipolla. “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics”. In: *arXiv preprint arXiv:1705.07115* (May 2017). (Visited on 03/15/2022).

[124] Anoop Korattikara, Vivek Rathod, Kevin Murphy, and Max Welling. “Bayesian dark knowledge”. In: *arXiv preprint arXiv:1506.04416* (June 2015).

[125] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[126] Corina Gurau, Alex Bewley, and Ingmar Posner. “Dropout distillation for efficiently estimating model confidence”. In: *arXiv preprint arXiv:1809.10562* (2018).

[127] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. “Born again neural networks”. In: *arXiv preprint arXiv:1805.04770* (2018).

[128] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. “Towards safe autonomous driving: Capture uncertainty in the deep neural network for LIDAR 3D vehicle detection”. In: *arXiv preprint arXiv:1804.05132* (Apr. 2018). (Visited on 03/15/2022).

[129] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. “Lasernet: An efficient probabilistic 3d object detector for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12677–12686.

[130] Janis Postels, Mattia Segu, Tao Sun, Luc Van Gool, Fisher Yu, and Federico Tombari. “On the practicality of deterministic epistemic uncertainty”. In: *arXiv preprint arXiv:2107.00649* (2021). (Visited on 03/15/2022).

[131] Sina Mohseni, Akshay Jagadeesh, and Zhangyang Wang. “Predicting model failure using saliency maps in autonomous driving systems”. In: *arXiv preprint arXiv:1905.07679* (2019). (Visited on 03/15/2022).

[132] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. “Deepxplore: Automated whitebox testing of deep learning systems”. In: *proceedings of the 26th Symposium on Operating Systems Principles*. 2017, pp. 1–18.

[133] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. “Addressing failure prediction by learning model confidence”. In: *arXiv preprint arXiv:1910.04851* (2019).

[134] Heinrich Jiang, Been Kim, Melody Y. Guan, and Maya Gupta. “To trust or not to trust a classifier”. In: *arXiv preprint arXiv:1805.11783* (May 2018).

[135] Manikandasriram Srinivasan Ramanagopal, Cyrus Anderson, Ram Vasudevan, and Matthew Johnson-Roberson. “Failing to learn: Autonomously identifying perception failures for self-driving cars”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3860–3867.

[136] Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. “Bias-reduced uncertainty estimation for deep neural classifiers”. In: *arXiv preprint arXiv:1805.08206* (2018).

[137] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. “Anomaly detection using one-class neural networks”. In: *arXiv preprint arXiv:1802.06360* (Feb. 2018). (Visited on 03/15/2022).

[138] Alireza Shafaei, Mark Schmidt, and James J. Little. “A less biased evaluation of out-of-distribution sample detectors”. In: *30th British Machine Vision Conference 2019 (BMVC 2019)*. 2019.

[139] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. “Do deep generative models know what they don’t know?” In: *arXiv preprint arXiv:1810.09136* (Oct. 2018). (Visited on 03/15/2022).

[140] Erik Marchi, Fabio Vesperini, Stefano Squartini, and Björn Schuller. “Deep recurrent neural network-based autoencoders for acoustic novelty detection”. In: *Computational Intelligence and Neuroscience* 2017 (2017), pp. 1–14.

[141] David Zimmerer, Fabian Isensee, Jens Petersen, Simon Kohl, and Klaus Maier-Hein. “Unsupervised anomaly localization using variational auto-encoders”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 289–297.

[142] Shiyu Liang, Yixuan Li, and R. Srikant. “Enhancing the reliability of out-of-distribution image detection in neural networks”. In: *arXiv preprint arXiv:1706.02690* (June 2017). (Visited on 03/15/2022).

[143] Angelos Filos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. “Can autonomous vehicles identify, recover from, and adapt to distribution shifts?” In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3145–3153.

[144] Charles Richter and Nicholas Roy. “Safe visual navigation via deep learning and novelty detection”. In: *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation, July 2017.

[145] Jan-Aike Bolte, Andreas Bär, Daniel Lipinski, and Tim Fingscheidt. “Towards corner case detection for autonomous driving”. In: *arXiv preprint arXiv:1902.09184* (Feb. 2019).

[146] Shreyas Ramakrishna, Zahra Rahiminasab, Gabor Karsai, Arvind Easwaran, and Abhishek Dubey. “Efficient out-of-distribution detection using latent space of β-VAE for cyber-physical systems”. In: *arXiv preprint arXiv:2108.11800* (2021). (Visited on 03/15/2022).

[147] Charles Hartsell, Shreyas Ramakrishna, Abhishek Dubey, Daniel Stojcsics, Nagabhushan Mahadevan, and Gabor Karsai. “ReSonAte: A runtime risk assessment framework for autonomous systems”. In: *arXiv preprint arXiv:2102.09419* (2021). (Visited on 03/15/2022).

[148] Alexander Amini, Wilko Schwarting, Guy Rosman, Brandon Araki, Sertac Karaman, and Daniela Rus. “Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 568–575.

[149] Petra Bevandić, Ivan Krešo, Marin Oršić, and Siniša Šegvić. “Simultaneous semantic segmentation and outlier detection in presence of domain shift”. In: *German Conference on Pattern Recognition*. Springer. 2019, pp. 33–47.

[150] Yingda Xia, Yi Zhang, Fengze Liu, Wei Shen, and Alan L Yuille. “Synthesize then compare: Detecting failures and anomalies for semantic segmentation”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 145–161.

[151] Philipp Oberdiek, Matthias Rottmann, and Gernot A Fink. “Detection and retrieval of out-of-distribution objects in semantic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 328–329.

[152] Inga Jatzkowski, Daniel Wilke, and Markus Maurer. “A deep-learning approach for the detection of overexposure in automotive camera images”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI: IEEE, Nov. 2018, pp. 2030–2035.

[153] Osama Makansi, Özgün Cicek, Yassine Marrakchi, and Thomas Brox. “On exposing the challenging long tail in future prediction of traffic actors”. In: *arXiv preprint arXiv:2103.12474* (2021). (Visited on 03/15/2022).

[154] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. “Learning spatiotemporal features with 3d convolutional networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4489–4497.

[155] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[156] Adam Houenou, Philippe Bonnifait, Véronique Cherfaoui, and Wen Yao. “Vehicle trajectory prediction based on motion model and maneuver recognition”. In: *2013 IEEE/RSJ international conference on intelligent Robots and Systems (IROS)*. 2013, pp. 4363–4369.

[157] Peng Liu, Arda Kurt, and Ümit Özgüner. “Trajectory prediction of a lane changing vehicle based on driver behavior estimation and classification”. In: *17th international IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2014, pp. 942–947.

[158] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. “Imitating driver behavior with generative adversarial networks”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 204–211.

[159] Florent Altché and Arnaud de La Fortelle. “An LSTM network for highway trajectory prediction”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 353–359.

[160] Ashesh Jain, Hema S Koppula, Shane Soh, Bharad Raghavan, Avi Singh, and Ashutosh Saxena. “Brain4cars: Car that knows before you do via sensory-fusion deep learning architecture”. In: *arXiv preprint arXiv:1601.00740* (2016). (Visited on 03/15/2022).

[161] Daxin Tian, Chuang Zhang, Xuting Duan, and Xixian Wang. "An automatic car accident detection method based on cooperative vehicle infrastructure systems". In: *IEEE Access* 7 (2019), pp. 127453–127463.

[162] Fu-Hsiang Chan, Yu-Ting Chen, Yu Xiang, and Min Sun. "Anticipating accidents in dashcam videos". In: *Asian Conference on Computer Vision*. Springer. 2016, pp. 136–153.

[163] Tomoyuki Suzuki, Hirokatsu Kataoka, Yoshimitsu Aoki, and Yutaka Satoh. "Anticipating traffic accidents with adaptive loss and large-scale incident db". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3521–3529.

[164] Xiaohui Huang, Pan He, Anand Rangarajan, and Sanjay Ranka. "Intelligent intersection: Two-stream convolutional networks for real-time near-accident detection in traffic video". In: *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 6.2 (2020), pp. 1–28.

[165] David Hallac, Suvrat Bhooshan, Michael Chen, Kacem Abida, Jure Leskovec, et al. "Drive2Vec: Multiscale state-space embedding of vehicular sensor data". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3233–3238.

[166] Xin Huang, Stephen McGill, Brian C. Williams, Luke Fletcher, and Guy Rosman. "Uncertainty-aware driver trajectory prediction at urban intersections". In: *arXiv preprint arXiv:1901.05105* (2019). (Visited on 03/15/2022).

[167] Simon Hecker, Dengxin Dai, and Luc Van Gool. "Failure prediction for autonomous driving". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1792–1799.

[168] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014). (Visited on 03/15/2022).

[169] Steven Smith. *Digital signal processing: a practical guide for engineers and scientists*. Elsevier, 2013.

[170] Mennatullah Siam, Mostafa Gamal, Moemen Abdel-Razek, Senthil Yogamani, Martin Jagersand, and Hong Zhang. "A comparative study of real-time semantic segmentation for autonomous driving". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 587–597.

[171] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. "Rethinking atrous convolution for semantic image segmentation". In: *arXiv preprint arXiv:1706.05587* (2017). (Visited on 03/15/2022).

[172] Gongfan Fang. *DeepLabv3Plus-Pytorch*. https://github.com/VainF/DeepLabV3Plus-Pytorch. (Visited on 03/15/2022).

[173] John Jonides. "Further toward a model of the mind's eye's movement". In: *Bulletin of the Psychonomic Society* 21.4 (1983), pp. 247–250.

[174] Charles W. Eriksen and Yei-yu Yeh. "Allocation of attention in the visual field". In: *Journal of Experimental Psychology: Human Perception and Performance* 11.5 (1985), p. 583.

[175] Anting Shen, Romi Phadte, and Gayatri Joshi. *Enhanced object detection for autonomous vehicles based on field view*. US Patent App. 16/703,660. June 2020.

[176] Luca Morreale, Andrea Romanoni, and Matteo Matteucci. "Predicting the next best view for 3D mesh refinement". In: *International Conference on Intelligent Autonomous Systems*. Springer. 2018, pp. 760–772.

[177] Ye Xia, Jinkyu Kim, John Canny, Karl Zipser, Teresa Canas-Bajo, and David Whitney. "Periphery-fovea multi-resolution driving model guided by human attention". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 1767–1775.

[178] James MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[179] Yue Wu, Yicong Zhou, George Saveriades, Sos Agaian, Joseph P Noonan, and Premkumar Natarajan. "Local Shannon entropy measure with statistical tests for image randomness". In: *Information Sciences* 222 (2013), pp. 323–342.

[180] Claude Elwood Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[181] Jianshu Chao and Eckehard Steinbach. "Preserving SIFT features in JPEG-encoded images". In: *2011 18th IEEE IEEE International Conference on Image Processing (ICIP)*. 2011, pp. 301–304.

[182] Hyomin Choi and Ivan V. Bajic. "High efficiency compression for object detection". In: *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Apr. 2018, pp. 1792–1796.

[183] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4510–4520.

[184] Yongxin Wang and Duminda Wijesekera. "Pixel invisibility: Detecting objects invisible in color images". In: *arXiv preprint arXiv:2006.08383* (June 2020). (Visited on 03/15/2022).

[185] Gyungin Shin, Weidi Xie, and Samuel Albanie. "All you need are a few pixels: semantic segmentation with PixelPick". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1687–1697.

[186] Philipp Krähenbühl and Vladlen Koltun. "Efficient inference in fully connected crfs with gaussian edge potentials". In: *Advances in Neural Information Processing Systems* 24 (2011), pp. 109–117.

[187] Adrian V. Dalca, John Guttag, and Mert R. Sabuncu. "Anatomical priors in convolutional networks for unsupervised biomedical segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9290–9299.

[188] Jason Ku. *AVOD - aggregate view object detection*. https://github.com/kujason/avod. (Visited on 01/18/2022).