M.Eng. Halil Beglerovic

Methodologies for Testing and Validation of Automated Driving Functions

DOCTORAL THESIS

to achieve the university degree of
Doktor der technischen Wissenschaften

submitted to
Graz University of Technology

Supervisor
Univ.-Prof. Dipl.-Ing. Dr.techn. Martin Horn

Institute of Automation and Control

Faculty of Electrical and Information Engineering

Graz, 2020

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

.....	.....
date	signature

Abstract

Automated Driving (AD) is one of the most anticipated and fast developing technologies today and it will have an enormous impact on our future. To capitalize on the various benefits of AD, a significant number of Original Equipment Manufacturers (OEMs) and research groups are trying to bring the technologies to the market as soon as possible. As it is nearly impossible to enable full autonomy directly, most of the groups focus on delivering parts of the fully automated system through Automated Driving Functions (ADFs).

As the ADFs will be incorporated into passenger vehicles, it is crucial to assure highest levels of safety and reliability. However, testing and validation of the ADFs is a very complex and time-consuming task. Currently, there is not a standardized way to test and measure performance of the ADFs. Generally, the performance of the ADFs are compared directly to human drivers, where the goal of the system is to be always at least safe as a human driver. What makes testing and validation so difficult are the various external and internal influences that the ADF can encounter on the road. To test the ADFs on all possible influences, their variations and combinations is very challenging, if not impossible.

To address some of the challenges of testing and validation, four research objectives have been considered in this thesis. The focus of the first objective is to minimize the testing efforts when the scenario and the corresponding parameter variations are known. The main idea is to minimize the number of test run while maximizing the information gained. This method is applied when testing is expensive in terms of setup time, setup cost, calibration effort, test duration, etc. We have shown that our proposed method is able to significantly reduce the number of test runs compared to other methods.

The second objective deals with cases when a model of the ADF is available, which makes it possible to simulate various influences and perform tests much faster. A benefit of this approach is that we can perform much more tests and cover the parameter space more thoroughly. This in return, makes it possible to make confident reports on the performance of the ADF.

The first and second objective deal with use-cases when a specific scenario and parameter ranges are given by a domain-expert. However, this information can be automatically obtained if labeled data is available. The main idea of the third objective is to apply Deep Learning (DL) and learn how to segment the driving scenarios. There

are two main benefits of this approach. First, if the data is segmented we can test only on the parts that are useful for the considered ADF, greatly reducing the testing effort. The second benefit is the ability to select relevant scenarios which can then be passed to the first two methods for further analysis. The relevant scenarios could be selected by defining some explicit behavior or Key Performance Indicators (KPI) we are interested in.

The main idea behind the fourth objective is to use unlabeled real data for scenario exploration. Unsupervised DL models try to group similar scenarios together in a low-dimensional latent space. By examining the latent space, is is possible to get insights on the distribution, sizes and relevance of individual clusters and their corresponding scenarios.

Keywords: Testing, Validation, Automated Driving, Automated Driving Functions, Scenario Extraction, Scenario Exploration, Deep Learning

Kurzfassung

Automatisiertes Fahren (Automated Driving, AD) ist heutzutage einer am meisten erwarteten und sich am schnellsten entwickelnden Technologien, die einen enormen Einfluss auf unsere Zukunft haben wird. Um die verschiedenen Vorteile von AD zu nutzen, versucht eine beträchtliche Anzahl von Originalgeräteherstellern (OEMs) und Forschungsgruppen, die Technologien so schnell wie möglich auf den Markt zu bringen. Da es fast unmöglich ist, eine vollständige Autonomie direkt zu ermöglichen, konzentrieren sich die meisten Forschungsgruppen auf die Lieferung von Teilen des vollautomatischen Systems durch Automatisierte Fahrfunktionen (ADFs).

Da die ADFs in Personenfahrzeuge eingebaut werden, ist es entscheidend, ein Höchstmaß an Sicherheit und Zuverlässigkeit zu gewährleisten. Das Testen und Validieren der ADFs ist jedoch eine sehr komplexe und zeitaufwändige Aufgabe. Derzeit gibt es keine standardisierte Methode zur Prüfung und Messung der Leistung der ADFs. Im Allgemeinen wird die Leistung der ADFs direkt mit menschlichen Fahrern verglichen, wobei das Ziel des Systems darin besteht, als menschlicher Fahrer immer zumindest sicher zu sein. Was Testen und Validierung der ADFs so schwierig macht, sind die verschiedenen äußeren und inneren Einflüsse, die ADF auf der Straße begegnen können. Die ADFs auf alle möglichen Einflüsse, ihre Variationen und Kombinationen zu testen, ist sehr anspruchsvoll, wenn nicht sogar unmöglich.

Um einige der Herausforderungen des Testens und der Validierung anzugehen, wurden in dieser Arbeit vier Forschungsziele berücksichtigt. Der Schwerpunkt des ersten Ziels besteht darin, den Testaufwand zu minimieren, wenn das Szenario und die entsprechenden Parametervariationen bekannt sind. Die Hauptidee besteht darin, die Anzahl der Testläufe zu minimieren und gleichzeitig die gewonnenen Informationen zu maximieren. Diese Methode wird angewendet, wenn das Testen teuer ist in Bezug auf Einrichtungszeit, Einrichtungskosten, Kalibrierungsaufwand, Testdauer, etc. Es wurde gezeigt, dass die von uns vorgeschlagene Methode in der Lage ist, die Anzahl der Testläufe im Vergleich zu anderen Methoden deutlich zu reduzieren.

Das zweite Ziel befasst sich mit Fällen, in denen ein Modell der ADF zur Verfügung gestellt wird, was ermöglicht, verschiedene Einflüsse zu simulieren als auch Tests wesentlich schneller durchzuführen. Ein Vorteil dieses Ansatzes ist, dass viel mehr Tests durchführbar waren und der Parameterraum gründlicher abzudecken war. Dies hat wiederum ermöglicht, zuverlässige Berichte über die Leistung der ADF zu erstellen.

Das erste und zweite Ziel befassen sich mit Anwendungsfällen, bei denen ein bes-

timmtes Szenario und Parameterbereiche von einem Domänenexperten vorgegeben werden. Diese Informationen können jedoch automatisch erhalten werden, wenn gekennzeichnete Daten verfügbar sind. Die Hauptidee des dritten Ziels besteht darin, Deep Learning (DL) anzuwenden und zu lernen, wie man die Fahrszenarien segmentiert. Es gibt zwei Hauptvorteile dieses Ansatzes. Erstens, wenn die Daten segmentiert sind, können nur die Teile, die für die betrachtete ADF nützlich sind getestet werden, wodurch der Testaufwand stark reduziert wird. Der zweite Vorteil ist die Möglichkeit, relevante Szenarien auszuwählen, die dann zur weiteren Analyse an die ersten beiden Methoden weitergegeben werden können. Die relevanten Szenarien könnten durch die Definition einiger expliziter Verhaltens- oder Leistungsindikatoren (Key Performance Indicators, KPI) ausgewählt werden, die von Interesse sind.

Die Hauptidee hinter dem vierten Ziel ist die Verwendung nicht unbeschriftete reale Daten für die Szenarioexploration. Unbeaufsichtigte DL-Modelle versuchen, ähnliche Szenarien in einem niedrigdimensionalen latenten Raum zusammenzufassen. Durch die Untersuchung des latenten Raums ist es möglich, Erkenntnisse über die Verteilung, Größe und Relevanz einzelner Cluster und der entsprechenden Szenarien zu gewinnen.

Schlagwörter: Testen, Validieren, Automatisiertes Fahren, Automatisierte Fahrfunktionen, Szenarioextraktion, Szenarorexploration, Deep Learning

Acknowledgment

Completing a PhD is a journey that a person does not make alone. Thank God, I was surrounded by people that motivated, encouraged and helped me to come this far.

I would like to thank my supervisor, Prof. Martin Horn, for your guidance, support, calmness and encouragement during the whole process. You have greatly influenced my work and helped me grow as a researcher.

I thank all colleagues from AVL List GmbH, starting with our department head Dr. Michael Paulweber, who has given me the opportunity to work and conduct research on the ITEAM project. Then, my supervisor Steffen Metzner. Thank you for your support and advices. Thank you for all the brainstorming and research we did together. Thank you Jonas Reubsam, Hans-Michael Koegeler, Juergen Holzinger, Thomas Schloemicher, Leitner Andrea, Abishek Ravi and Niklas Wikstroem for your help and the research collaboration.

I would like to express my gratitude for everybody involved in the ITEAM project. Thank you Prof. Valentin Ivanov for the support and amazing job you did as the project coordinator. Thanks to all the early stage researchers who worked alongside me in the project. I especially want to thank Zlatan Ajanovic for encouraging me to apply for this position, for the research and the wonderful time we spent together during the project. Thank you Michael Stolz, for your help and support during the beginning of the project and for the research supervision. Thank you Prof. Alessandro C. Victorino and Shriram Jugade for hosting me on my secondment at UTC (France). Thank you Prof. Bakir Lacevic for inviting me to conduct research at UNSA (BiH). I am very thankful for all the workshops, summers schools and events we had during the ITEAM project.

I would like to thank my wonderful family for their support, understanding and encouragement. Especially, my wife Selma. Thank you for the unconditional love and support. Thank you for all the weekends we spent home working. Thank you for all the sleepless nights proofreading and correcting everything I wrote. Thank you for taking care of our daughter so that I could focus on the thesis.

I would like to express my appreciation and gratitude for the financial support this project has received. Without it, this research would not be possible. The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 675999.

Graz, June 2020

Contents

1

Introduction

This chapter summarizes the motivation, objectives and scope of the conducted research. First, it lists some of the benefits of AD explaining why it is such an important and anticipated technology. Further, it shortly introduces challenges for developing reliable ADF from which the objectives of this thesis are drawn. Finally, it outlines the research steps, indicating which topic will be covered in which chapter.

1.1 Motivation

With the current development rate of ADFs, fully autonomous vehicles do not seem to be that far away. All of the major OEMs are investing a considerable amount of resources for the development of ADF, indicating its importance. The OEMs see a huge potential in the market that AD will bring. It is estimated that AD will bring more than 600 billion euros of revenue by 2025 [1]. Other than sheer economical gain, there are even more important social and environmental benefits where AD will have a huge impact.

Maybe one of the most important social impacts of AD is that it will make driving much safer, both for the passengers and other traffic participants. According to the European Road Safety Observatory [2], the number of casualties in traffic accidents is around 1 million in the European Union. Most of these accidents are caused by human error. These types of errors occur when the human driver is either tired, sleepy, distracted or not able to responsibly drive the car. Most, if not all, of these errors would be eliminated by Autonomous Vehicle (AV), as machines don't get tired, sleepy or distracted. An AV is evaluating the current traffic situations several times per second and it is also able to react on unexpected situations more reliably than a human.

Another prominent benefit of AD is a significant reduction in traffic congestion. As traffic congestions are directly linked to traffic accidents [3], AV would already decrease the congestions by lowering the number of accidents. However, AVs will be able to reduce the congestions even further. Being able to communicate with other vehicles, Vehicle to Vehicle (V2V), and with different infrastructures, Vehicle to X (V2X), makes

it possible to optimize velocity, starting and stopping times, etc. [4], greatly impacting the flow of traffic.

Under normal conditions, human drivers tend to create stop-and-go traffic. Creating the effect called "phantom traffic jam". Basically, by accelerating more than necessary and then breaking, human drivers create a phantom obstacle which extends to all subsequent vehicles. It has been shown that, by using a few AVs it is possible to totally eliminate the undesired effect [5], [6]. The AVs can recognize that a phantom traffic jam is occurring and optimize its motion in such a way to eliminate the effect.

The reduction of traffic congestions has further beneficial impacts. The first one is reduced emissions. Research done by Bart *et al.* [7] states that by using different congestion mitigation and velocity adjusting strategies it is possible to reduce $C0_2$ emission by 20%. These strategies consists of the aforementioned "phantom traffic jam" elimination and optimal velocity control to lower higher velocity traffic to a more efficient state. Similarly, Lee *et al.* [8] have shown that by using V2X and V2V it is possible to design smart intersection control, cutting down $C0_2$ emission and fuel consumption by 40%.

The second benefit of reduced traffic congestion is the reduction in travel time and transportation costs. It is estimated that AVs could achieve a 40% reduction in overall travel time leading to recovery of a staggering 80 billion lost hours in the US alone [9]. The reduced travel time will potentially also lower the price of consumer goods as the fuel consumption and man-hours go down.

In addition to the emission reduction through decreased traffic congestion, AVs can also use various optimization techniques to reduce the overall fuel consumption and consequently the CO_2 emission. Some of these strategies include optimized acceleration and deceleration on traffic lights or dense city driving. If V2X is used, then the impact is even bigger as the vehicles can plan exactly how much and when to start moving depending on the current light situation. Another strategy could include online acceleration and velocity optimization depending on the road grade and current traffic situation. It has been shown that, by using online optimization, the AV can reduce the overall fuel consumption and emission up to 40% in city driving [10].

Using fast reacting times, fast sensing and V2V, AVs have a great potential to increase the traffic throughput or lane capacity. By using platooning, AVs are able to decrease the space they are occupying and also save fuel as the vehicles behind the leading vehicle don't have to fight air resistance [11]. Platooning is proving to be extremely useful for transportation over long distances done by trucks.

The next benefits of AVs are related to mostly city scenarios. AVs will offer a much cheaper taxi alternative, as there will be no man-hour costs. In addition, it is anticipated that these vehicles will be running on electricity where they will have dedicated charging stations. The AV will drive itself to recharge when necessary. Another benefit of AV driven taxis is that the waiting time can be significantly reduced as optimization techniques could distribute the vehicles where they are most needed for a given time. Similarly to charging, the AVs will be able to park outside the urban areas or even outside cities making space available for more important infrastructure. An example could be a person arriving at work and then the AV drives itself to the best parking

space. Similarly, the AV would pick-up the person at the end of the day.

In the future, we anticipate that AVs will not need any input from the user. This means that mobility will be available for more people. Most impacted groups are people with special needs, elderly people and children.

1.2 Objectives

After mentioning some of the most important benefits AD will bring, one could ask: What needs to be done for AV to finally arrive?

There are many challenges that the development of AVs faces, but among the most important ones is testing and validation of not only the individual ADFs but also the complete system. The second chapter will introduce in more detail the specific challenges. Here, we will just briefly point-out some of them in order to introduce the objectives of the thesis.

Currently, there are no standardized tests that an AV should complete. The performance is measured against humans, with the goal that the AV should be at least as safe as a human driver. In addition, it is not possible to do testing that would cover all possible situations that an AV could encounter because of the huge amount of time this would take, and the huge number of external and internal influences.

However, even when execution time would not be a problem, we still would not know on which scenarios or situations should we test the AV as we don't know beforehand where the system is going to fail.

Driven by these challenges, we devised four objectives for the thesis.

- **O1: Reduction of test runs** - Research and develop methodologies that can reduce the number of test runs or parameter space exploration for testing of ADFs
- **O2: Validation using Model-based approaches** - Research and develop methodologies that are able to leverage simulation models in order to accelerate testing and validation.
- **O3: Scenario Classification** - Research and develop methodologies that can use labeled data in order to classify and extract scenarios for testing.
- **O4: Scenario Exploration** - Research and develop methodologies that can use the large amount of unlabeled data to understand distribution, groups and relevance of scenarios for testing.

1.2.1 O1: Reduction of test runs

Objective 1. focuses on the reduction of test runs needed for testing or falsification of ADFs. For example, let's consider an Automatic Emergency Braking (AEB) on a specific scenario. In the scenario, an unwanted obstacle is found on the highway. First, we need to define the number of lanes, highway curvature and other static parameters. Then, we define the varying parameters. For, example, vehicle velocity,

obstacle height and radius, maximum allowed deceleration, etc. The objective of the developed method is to find out if the ADF is working without errors whilst minimizing the number of actual test runs. Minimizing test runs is important for testing where, the pipeline setup, initial calibration and running the test is very expensive or very time-consuming. The number of runs could also be limited by various constraints like deadlines or budget so we need to be sure that every test run we execute provides as much information as possible.

1.2.2 O2: Validation using Model-based approaches

Even though we tried to find faulty behavior and maximize the information gain from every test run in Objective 1., we can not provide any guarantees on the performance of the ADF under test. That is why in Objective 2. we explore methodologies which can use simulation models in order to perform extensive testing and validation. This is possible because of the benefits that simulation provides. First, we are not bound to run test at real time. Second, setup and reproduction of test is trivial and test execution can be done in parallel making it possible to execute significantly more test runs. After executing a large number of test runs, it is possible to verify, with a certain confidence, that the ADF is working without errors in a specific parameter range.

1.2.3 O3: Scenario Classification

Methods developed in Objective 1 and 2 need as input a specific scenario and parameter ranges. The required information is typically provided by the user. However, in Objective 3. we try to provide the necessary scenarios in a more automated way. A trivial way would be to use recorded data and process all available scenarios; however, this is very inefficient and time-consuming. The core idea of the proposed methods is to use Machine Learning (ML) and learn how to segment the recorded data. The scenario segments make it possible that only the relevant scenarios are considered for testing. For example, let's look at driving on a highway. Most of the driving is actually free driving or following another vehicle in a specific lane. If we already know that our system is behaving correctly on this type of scenario, we could omit it partially, or completely, from our next test and focus mostly on the new scenarios or parts where we know the ADF is not behaving optimally. A real-world example could be a last minute cut-in scenarios before a highway exit. If we want to test our system only on those type of scenarios then we need to be able to identify and extract them from our data.

1.2.4 O4: Scenario Exploration

In Objective 3 we were using labeled data in order to train ML models used for scenario classification. The limitation of this approach is that we are able to classify only scenarios that were labeled and present in the training data.

The aim of Objective 4 is to extend this approach and make it possible to explore scenarios in an unsupervised way. The DL models group scenarios by similarity creating

scenario clusters. The user can explore these clusters and get information regarding distribution, sizes and possibly find clusters where unknown or unconsidered scenarios are grouped. Furthermore, if the user is considering a specific scenario, he can find out if similar scenarios exist in the data and if so, he is be able to find out which ones they are.

In addition, this method can be used to explore the behavior of pre-trained models. Specifically, we can search for misclassifications or false positives. After the interesting clusters are found and labeled, we can retrain the model, improving its performance.

The developed research objectives on testing methodologies and scenario selection create a positive feedback loop. The testing can lead the classification and extraction to interesting scenarios by checking where the ADF is having problems. Similarly, the scenario classification and extraction methodologies can be used to search for similar scenarios and feed them back to the testing methodologies.

1.3 Scope of the Research and Contributions

The scope of the research and contributions can be divided into the corresponding objectives.

- **O1** - The methodology developed for the first thesis objective "Reduction of test runs" is focusing on ADF falsification. The proposed approach is used when we are testing on real hardware, or execute very complex and time-consuming simulations, where the setup time, run time, cost, etc., are very high. In other words, we are limited by the number of tests we can actually execute. The idea is to maximize the amount of information we can gain from these test runs. The proposed method is not only limited to ADFs but can be used on any System Under Test (SUT). The main contribution of this research is a novel algorithm that is able to reduce the number of executed test runs but still find faulty behavior. In addition, by using a model of the system behavior we are able to perform optimization task much faster than on the real system.

  This research was introduced in the conference paper: *"Testing of Autonomous Vehicles using Surrogate Models and Stochastic Optimization"* [12].
- **O2** - The Model-based Validation approach is used when we are not limited by the number of test runs. An example could be when we have a simulated model of the ADF. In this case, we can execute more thorough testing to assure that in some specific parameter bounds the system is working without errors. The main contribution of this research is the adaptation and extension of tuning methodologies in order to facilitate model-based validation. In addition, a use case where an Adaptive Cruise Control (ACC) was firstly tuned and then validated is shown.

  This research was introduced in the book-chapter: *"Model-based Safety Validation of the Automated Driving Function Highway Pilot"* [13].
- **O3** - For the third thesis objective "Scenario Classification", two research contributions were made:

1. Scenario classification related to the Ego vehicle
2. Scenario classification with dynamic traffic participants

The first contribution on scenario classification focuses only on the ego vehicle states and road information. Information coming from various sensors like acceleration, velocity, steering angle, road curvature etc., are given to a Neural Network (NN) which estimates the current scenario. The main contribution of this research is the successful usage of DL to classify scenarios related to a Lane Keep Assist (LKA). Furthermore, several classification models were investigated, and an offline and online classification approach was introduced.

This research is described in the article paper *"Deep Learning Applied to Scenario Classification for Lane-Keep-Assist systems"* [14].

The second contribution focuses on methods which make it possible to classify also scenarios with other traffic participants. The main contribution of the proposed method is a very efficient traffic representation that can be used by NN to distinguish between various dynamic scenarios. The scenarios can later be segmented and testing can be done only on the desired parts.

This research was introduced in the conference paper *"Polar Occupancy Map - A Compact Traffic Representation for Deep Learning Scenario Classification"* [15].

- **O4** - The methodology developed for the fourth objective deals with exploring scenarios from recorded data. The main contribution is the use of DL to create a scenario latent space which can be used to find interesting scenario clusters, ease labeling, explore behavior of pre-trained models, etc. This work has not been published yet and it will be introduced in this thesis.

Additionally, the first contribution that was made for the thesis was an overview of the current challenges for testing of AVs.

The corresponding book chapter is called *"Challenges for the validation and testing of Automated Driving Functions"* [16].

1.4 Thesis Outline

The outline of the thesis in shown in Figure 1.1. At the top we see **Chapter 2**. which is dedicated to the challenges of testing and validation of ADFs.

Below **Chapter 2**, two groups are shown. On the left side, we have the developed testing and validation methodologies related to Objective 1. and 2. **Chapter 3** introduces the ADF test reduction through surrogate modeling and stochastic optimization. **Chapter 4** introduces the model-based validation approach.

On the right side, we have the methodologies which make it possible to extract relevant scenario related to Objective 3. and 4. **Chapter 5** introduces the DL classification approaches regarding the ego vehicle and road information. **Chapter 6** introduces the classification approaches for dynamic traffic participants. Finally, **Chapter 7** introduces the scenario exploration methodologies.

The arrows in the diagram indicate the synergy between the approaches. In order

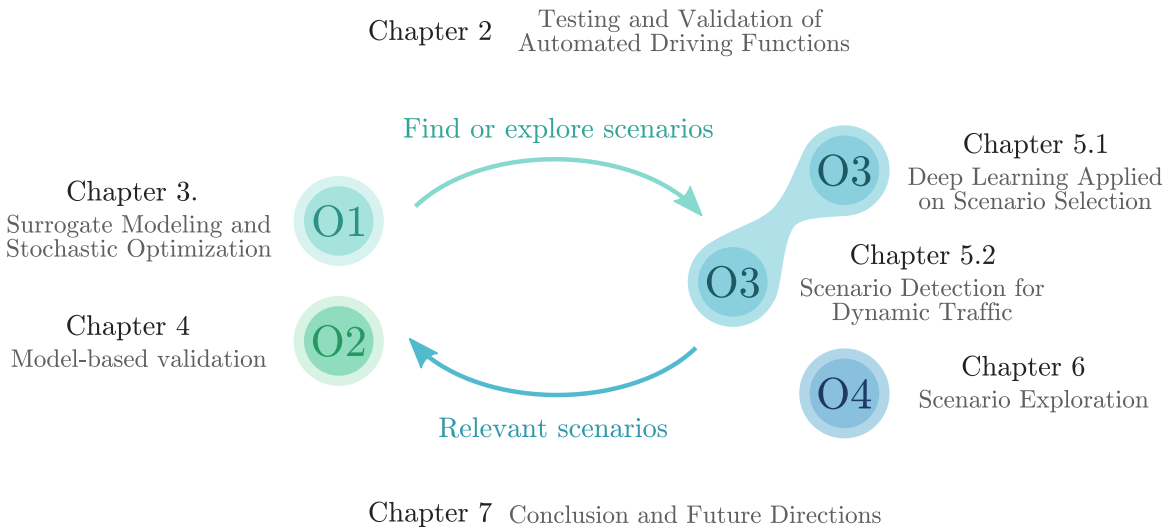

Figure 1.1: *Thesis outline with objectives and research linked to the corresponding chapters.*

to efficiently use the test reduction and validation methodology, we should provide scenarios that are significant to the ADF that is currently under test. On the other side, information of the current behavior and test success, gained from the reduction of test cases and validation methodology, can be used in order to find the most relevant scenarios for further testing.

Finally, in **Chapter 8** conclusion and the possible future directions are presented.

2

Testing and Validation of Automated Driving Functions

This chapter provides an overview of the challenges for validation and testing of ADF. We provide an overview of current methodologies used for validation and testing, focusing on the missing parts. Furthermore, we give an insight into promising methodologies, frameworks, and research areas which aim to reduce current testing and validation efforts.

Most of the content presented in this chapter is adopted from

H. Beglerovic, S. Metzner, M. Horn, Challenges for the validation and testing of automated driving functions, in: C. Zachäus, B. Müller, G. Meyer (Eds.), Advanced Microsystems for Automotive Applications 2017, Springer International Publishing, Cham, 2018, pp. 179–187.

2.1 Introduction

In order to satisfy the increasing demand for safety, reliability and comfort of commercial vehicles, manufacturers and research groups put great effort in the development of new and sophisticated driving functionalities throughout the decades. This development has undergone several phases. It started with Driver Assistance Systems which required constant driver monitoring, e.g., Cruise Control (CC), where some were active only in certain situations, e.g., Electronic Stability Program (ESP).

With the increase in the processing power of computers, researchers and OEM were able to include more sophisticated algorithms and sensors into the vehicle, what led to the development of Advanced Driver Assistance System (ADAS). The new technologies allowed the development of ACC, LKA, Lane Change Assist (LCA), traffic signs, pedestrian and vehicle detection, etc. Even though these systems are more complex and sophisticated, the driver still needs to monitor them continuously in order to compensate unforeseen conditions on the road or misbehavior of the algorithm. Nevertheless, the safety benefits of ADAS systems have led governments to make laws which obliges manufactures to include some of them into commercial vehicles (e.g., Anti-Lock Braking System (ABS) and ESP), showing that these systems have become

an irreplaceable part of the driving experience.

In the recent years, we observe a significant rise in the set of functionalities ADF provide. ADFs present the next technological step as they aim to provide a driving experience where constant monitoring of the system is not needed. In its core, ADF combine several ADAS functions in a comprehensive and complex system. For example, a Highway Pilot is combining ACC, LKA, LCA, traffic sign and vehicle recognition, path planning, etc., in order to successfully drive the vehicle on a highway. Many manufactures like Tesla, Daimler, etc. have already demonstrated various capabilities of Highway Pilots.

Current technology allows limited automated driving on specific scenarios and the aim for the future, as shown in Figure 2.1, is to enable full autonomy where human intervention is not needed at all. These systems should be able to perceive and understand the environment in order to act accordingly to all possible situations encountered in real traffic. Such systems would open the door to new types of transportation where fleet of automated vehicles could be shared between people, effectively lowering the overall number of used vehicles. In addition, the mobility of elderly people or people with disabilities would increase drastically.

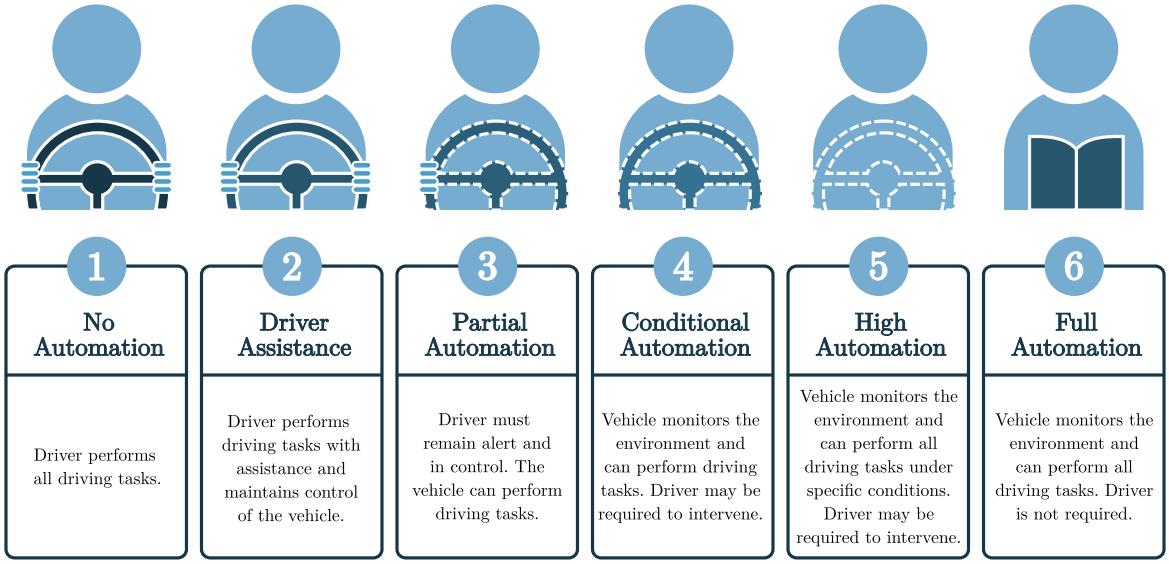

Figure 2.1: *Levels of automation - Society of Automotive Engineers (SAE)*

2.2 Challanges for Testing and Validation

The challenges for testing and validation of ADFs are encountered on both the tools and methods side. On the tools side, there is an increasing need to develop and test very complex ADFs consisting of many sub-systems. Furthermore, new signal types need to be incorporated in a standardized way making it possible to resolve the problem of transferring huge amounts of data between the control units, which is out of scope for the classic transfer protocols used in today's vehicles. In addition,

development environments need to be able to incorporate these new signal types in the development process and deal with the increasing complexity of ADFs.

On the method side, the Safety of the Intended Functionality (SOTIF) [17] needs to be ensured while remaining economically feasible. Economic feasibility is very important as the ADF could be tested on a theoretically infinite variety of scenarios. Furthermore, the scenarios include a huge number of parameters that are used to faithfully reflect the reality. In order to keep up with the high commercialization demand, rapid developments and advancements in technology, testing and validation procedures need to lower the current testing effort, and in addition, new innovative methods need to be developed. Watzenig and Horn [18] state that there is an increasing demand for methodologies and validation procedures that will enable the development of ADF which allow greater safety, traffic flow optimization, reduced emission and enhanced mobility.

The following sections will introduce the tools and methods challenges into more details.

2.2.1 Complexity of Automated Driving Functions

A general structure of an ADF can be seen on Figure 2.2. The task of an ADF is to perceive and understand the environment and take adequate actions depending on the current situation. In order to accomplish this task, ADF use an increasing number of heterogeneous sensor systems and complex algorithms, fusing and interpreting the data of the dynamic environment. The information coming from the sensors are combined with existing knowledge from maps, V2V or V2X communication. All of these inputs, together with the internal states of the vehicle, e.g., velocity, engine speed, etc., are needed by the decision-making algorithms in order to predict and plan adequate trajectories.

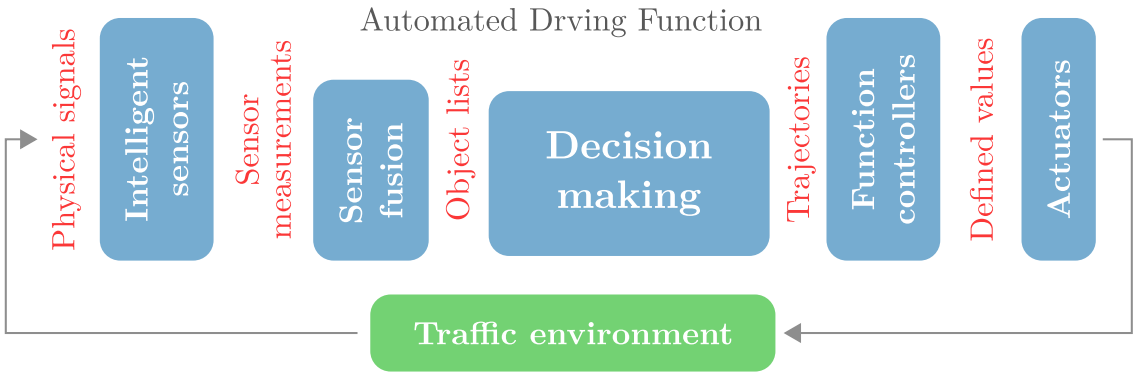

Figure 2.2: *Structure of an Automated Driving Function.*

Each of the individual sub-systems of the ADF has to be tested and validated separately before being used in the complete system. Additionally, the testing and validation procedure can vary between the building blocks which brings in additional complexity.

After testing each individual building block, testing of the integrated system can provide additional information and show where improvements can be made. That is why an iterative approach is needed, where continuous improvement and integration is possible. In general, this development stage relies heavily on the use of simulations where test and corrections can be done quickly. After satisfactory behavior is achieved in this step, the testing and validation is continued on proving grounds or real roads.

However, some of the main challenges in the development and testing of complex ADFs are:

- Reducing the number of test runs for specific scenarios.
- Finding the appropriate scenarios for testing and validation.
- Data interface standardization between sensors, hardware components and simulation. On the simulation side, some standardization approaches like Open Simulation Interface (OSI) [19] are being developed and show good progress in this regard but still some datatypes are not supported.
- Easy access and debugging of the specific datatypes in the programing environments used for development.
- With the advancement of technologies like DL, dedicated hardware is introduced and developers need to be able to include both the models and hardware into their development environment.

2.2.2 Variation of Scenarios and Parameters

Winner *et al.* [20] state that an ADF should be driven for more than 240 million kilometers without fatalities to prove that they are, at least, not performing worse than human drivers. Performing such an extensive validation, using classical validation methods on public roads or proving grounds, it is not possible in a sustainable frame. If we take into consideration the development process introduced in the previous section, with several iterations of testing and validation, it becomes clear that such extensive test runs are not economically feasible.

However, if we consider possible scenarios and respective parameter variations, the ADF could theoretically be tested on around 10^{12} different test cases. This type of full factorial testing is also not feasible and new advanced methods are needed for appropriate scenario selection and parameter variation. The parameter variation and test run reduction are the main research focus of the first thesis objective.

On the other hand, simulation has been proven to be a very promising tool for successful development and testing of ADF. The most important advantage is the ability to easily vary various scenarios and parameters, ensure reproducibility and, in the case of pure Software in the Loop (SIL), even run test cases in parallel and faster than real-time. This advantage is being explored in the second thesis objective.

2.2.3 Scenario Selection and Test Generation

In order to successfully validate an ADF, an appropriate selection of scenario subsets from all possible scenario variations is needed. Furthermore, the scenario subsets have to provide a sufficient scenario coverage because only a small portion of all available scenarios presents a challenge for the ADF and could potentially lead to faulty behavior. It is important to reduce the number of considered scenarios in order to save time, costs and resources needed for testing and validation.

The main task is the systematic selection of scenarios and corresponding parameters for the variation. A possible approach for the scenario selection is to group all scenarios into different categories, e.g., highway, left turn, right turn, country road, etc. With this type of grouping methodology, it is possible to conduct validation on certain test runs and exclude test runs with similar characteristics from further consideration. This idea is explored in the third thesis objective.

An opposite approach could be applied by identifying scenarios which are most relevant for the ADF. These scenario could be selected either by the engineers, or by using some predefined KPIs or looking directly at scenarios that lead to most critical behavior. Then by knowing these relevant scenarios we could expand the testing to similar scenarios and stop only after the desired criterion is not fulfilled anymore. This methodology is explored in the fourth thesis objective.

Additionally, a subset of scenarios could be selected depending on specific regulations, imposed by the laws from different countries. After appropriate scenario selection, an automated test generation procedure is needed. This procedure should be able to generate appropriate test cases taking into consideration the capabilities and level of autonomy of the ADF.

It is important to mention that no unified metrics, references nor testing criteria exist up to today. The test generation is usually carried out by domain-experts tailored to meet their specific needs. Better objective evaluation methods are needed which can derive, with some certainty, the overall behavior of the model from just a subset of test cases.

2.3 Current Methodologies overview

Today's ADF functions are developed using environment simulation systems like VTD from Vires, PreScan from Tass, IPG CarMaker, Carla [21], AirSim [22], SUMO [23], etc. All available environment simulation systems have strengths and weaknesses depending on the origin domain (vehicle dynamics, driver simulator, etc.) and allow manual design and simulation of a wide range of scenarios. Dependent on the complexity of the respective scenario, its setup and design can take a lot of time.

As for usual control units, the functionality and the electronics are integrated and tested on Hardware in the Loop (HIL) systems. The HIL systems are either used for testing one single control unit or testing several units in compound on an integration HIL. Usually, setup and maintenance need a lot of effort. Variability and the possibility

to influence the inputs of the control unit independent from the environment of the vehicle overcome these disadvantages.

After the integration of the functionality into the vehicle, testing and validation is nowadays executed on test tracks and public roads. Especially on public road, the reproducibility of scenarios is very difficult and can be very time consuming.

Recently more and more ground truth measurement systems are being developed and used. Such systems are able to collect the data of the environment with higher precision than the mass production sensors inside the vehicle. These systems enable the engineers to compare the sensor output of both the vehicle sensors and the ground truth measurement system, facilitating easier debugging and development. Reproducing such measured scenarios in simulation environments, in order to debug the source code in detail, still means a lot of manual work with limited tool support.

However, these ground truth measurements are also used to develop and train new perception and planning algorithms using DL. Some of the most popular datasets are: KITTI [24], Berkeley Deep Drive (BDD100K) [25], HighD [26], INTERACTION [27], Waymo Open Dataset [28], Lyft Level 5 [29], Comma 2k19 [30], etc. Testing, validation and improvement of the trained DL methods are very challenging tasks. As mentioned before, continuous improvement and integration is used to incrementally increase the performance of the algorithms. Some OEMs went even further; they use the available fleet of vehicles and the so called "shadow mode" to test the algorithm performance directly on the vehicle during operation. In "shadow mode", the ADFs are running and generating decisions. However, these decisions are not executed directly. Instead, they are compared to the behavior of the driver and if a strong deviation between the two behaviors exist, the occurrence is reported and further analysis and testing is done. By leveraging a large fleet with over-the-air software updates, it is possible to expose the ADFs to a huge variety of real world scenarios and events. This leads to very short and effective improvement cycles early on in the development.

In addition to this type of testing and validation, the OEMs also provide more mature ADFs to the customers during the development stage. In this case, the customer is liable and is required to monitor the system during operation. If the fleet of available vehicles is large enough then the OEMs are able to quickly test and validate their improvements. In addition, the OEMs record the data in order to further improve the functions. However, this type of testing heavily depends on the number of vehicles available in the fleet, and interesting scenarios and events still need to be extracted as most of the driving already has satisfactory behavior.

2.3.1 Supporting tools in the validation task

The environment simulation tool is the central tool on all development and validation levels involving simulation. Such environment simulation systems are already available from several suppliers and usually include sensor models that can be directly connected to the ADF controllers. Dependent on the complexity of the models, a huge amount of processing power may be required. This includes models involving ray-tracing methods necessary for detailed camera or radar models. However, for a lot of cases this accuracy

is not necessary or even obstructive. If very accurate sensor models are used and the robustness regarding special effects of the sensor shall be tested, it is also necessary to model the environment in the same level of detail to trigger these effects. In such cases, a phenomenological model may be a better choice as the modeling of the scenario is made much easier. This is also valid for other parts of the model, e.g., the vehicle dynamic model complexity may be reduced, if the test case does not imply stability critical maneuvers. Therefore, simulation platforms need to support the exchange of models dependent on the parametrization of the test case.

Another area where high accuracy sensor models are very important is Vehicle in the Loop (VIL) testing. The outputs of the sensor models can be directly injected into the Electronic Control Unit (ECU) of the vehicles if an adequate interface is present. Similarly, the injection can also be done on the sensor itself through adequate signals. Sensor stimulation has already been successfully applied on camera and radar sensors [31]. The aim of VIL testing is to further reduce real world testing as the setup time and repeatability of test is easier.

Even if methods are found to reduce the number of test case, there are still millions of kilometers which need to be driven either virtually or on real roads. In addition, test cases need to be handled in a structured manner and if no real hardware is involved it is possible to execute these test cases in parallel on the cloud. This coordinated distribution of the test cases and collection of the results is also an important requirement to the simulation platform.

However, it is still not possible to execute all test cases virtually or with only partial hardware. Some of the test cases or scenarios still need to be executed in real environment or test tracks. The data collected should be as close as possible to the data collected in simulation to enable usage of the same evaluation metrics. In order to accomplish this metrics transfer, ground truth measurement systems which record the environment with similar sensors as the vehicle but with higher precision, are necessary.

2.3.2 Standardization

For development and validation of ADF, both standards for the tooling and for the methodology need to be extended. On methodology side ISO26262 [32] is a well-accepted and applied functional safety standard for the development and validation of functionalities, realized with electronic control systems. In the concept phase of development, this standard demands, among others, an item definition and a hazard and risk analysis.

Both require the knowledge of the item (which may be available) and its environment, as well as the dependencies of the item to the environment. However, the fact that the number of scenarios and the variation of the environment parameters (daylight, weather, temperature etc.) is large restricts the direct application of ISO26262 on automated functions with automation level 3 and above. However, there needs to be a definition of a finite testing space regardless which method and which tools will be necessary for validation. Finding this testing space is even harder considering

that all implementations of the ADF functionalities will have different strengths and weaknesses. A good example is the development of the vehicle dynamics functionality. In all test cases defined by authorities for, e.g., brake distance or stability during a lane change maneuver, systems from different manufacturers react in a similar manner. But comparing those systems outside of this test space leads to significant differences as the developers are focusing on the defined test cases.

Therefore, the test case space for autonomous driving functionalities needs to be chosen wisely to avoid over- and under-optimized areas. In the worst case, this may lead to a product passing all tests but still not being useful for the customers.

ISO26262 has been published at a time where electronic control units have been already used for many years also for safety critical applications. Hence, it contains a lot of experience gained through the development of those control units. The same is to be expected for publication of the standard covering automated driving.

On the tools side the same problems as on control unit site occur as all bus systems, especially the CAN-bus, are optimized for signal-based controls-communication. This kind of communication will still play a big role in the future, but with new functions considering the objects of the environment, also object-based communication becomes necessary. For small objects containing few data, the CAN-bus may still be used, although the CAN-drivers need to be adapted. But the CAN-communication description and well known DBC-file format cannot be applied. Here it is necessary to establish new formats describing the exchanged objects and bus-systems able to handle this kind of communication properly. This adaption needs to be applied to simulation, measurement and evaluation tools and on all related file-types.

For the simulation environment several steps for the standardization of interfaces and file formats have been made. The OSI interface tries to bridge the description of vehicles and attributes between the different simulators and the real world. The OSI interface is an object oriented format where at each time-step all traffic participants are recorded together with their attributes. The OpenDRIVE [33] file format provides a unified way to analytically express road networks. The OpenCRG [34] file format provides a standardized way to describe the road surfaces. OpenScenario [35] defines a file format for the description of dynamic traffic participants.

In the end, it is still unclear how validation, homologation and certification of automated driving functions above SAE automation level 2 shall be done. However, there are few points that seem to be accepted among research groups and OEMs:

- The solution uses simulation for a major part of scenario testing and validation.
- A clever combination of methods and validation environments (SIL, HIL, VIL, test-track, public road etc.) is necessary.
- The number of test-cases/scenarios will still be quite big so methodologies that can reduce them are necessary.

As it can be seen, testing and validation of ADF is a challenging problem that will take some time to be properly solved. In the thesis, we choose four objectives that will hopefully bring us closer to that goal. In the following chapters we will discuss those objectives and their corresponding research in detail.

3

Reducing the Number of Test Runs

This chapter introduces an innovative method for the reduction of test runs based on surrogate models and stochastic optimization. The approach presents an iterative zooming-in algorithm aiming to minimize a given cost function and to identify faulty behavior regions of an ADF within a specific parameter range. The surrogate model of the ADF is updated in each iteration and is further used for intensive evaluation tasks, such as exploration and optimization.

Most of the content of this chapter is adopted from

H. Beglerovic, M. Stolz, M. Horn, Testing of autonomous vehicles using surrogate models and stochastic optimization, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 1–6. doi:10.1109/ITSC.2017.8317768. © IEEE 2017

3.1 Introduction

As it is not possible to test an ADF on all possible scenarios and parameter variations, we need to find a way to reduce testing to a specific number of test runs whilst extracting the most information out of it. The specific number of test runs is usually determined by various deadlines, fixed budgets, etc. making it very challenging to plan adequate test runs which maximize information gain.

In this chapter, we introduce an iterative approach that guides the testing towards faulty behavior regions in the parameter space, as shown in Figure 3.1. This type of testing is also known as falsification.

In order to identify the faulty behavior regions, appropriate cost functions, which can be minimized by various optimization methods, must be defined. As we are interested in the region around the worst behavior, and we want to avoid false positives in the form of local minima, global optimization methods are needed. One drawback of global optimization algorithms is that they require many function evaluations (test runs) to find the global optimum. To overcome this limitation in the proposed approach, we create a surrogate model with inexpensive evaluations on which the optimization algorithms can run. With each new iteration a better model of the system is built

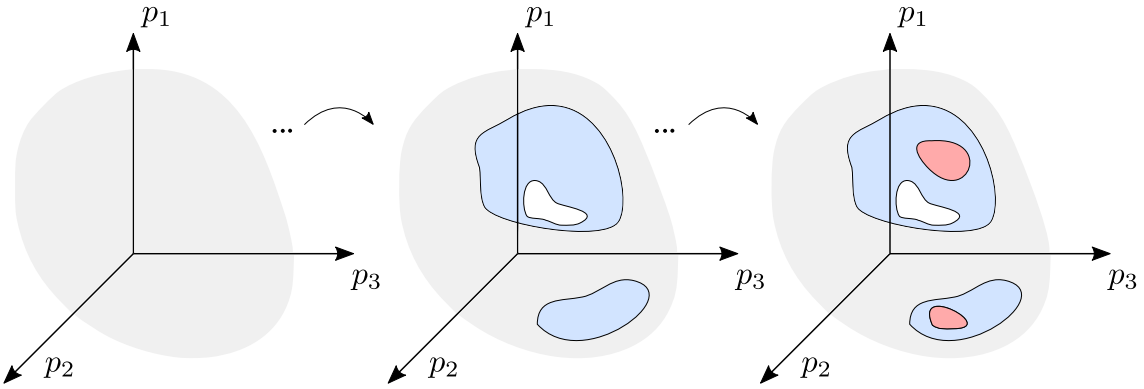

Figure 3.1: *Iteratively locating a faulty behavior region inside the parameter space © IEEE 2017*

around the faulty region. The advantage of this method is that it can handle black box systems when an appropriate feedback cost function is present.

In summary, the main contribution of this research is the development of a methodology based on surrogate modeling and stochastic optimization, used for falsification of ADFs.

3.2 Related Work

Several research groups have addressed the state-of-the-art topics on autonomous vehicle testing using simulation or mixed virtual and real setup. Stellet *et al.* [36] focus on the taxonomy and high level approach, problem statement and requirements regarding the testing of autonomous vehicles. They propose a systematic methodology for testing, elaborating on the need for properly defined metrics, references and scenarios.

Huang *et al.* [37] present an overview on current methodologies, tools, platforms and proving grounds used for testing of autonomous vehicles. Zofka *et al.* [38], [39], [40] and Jemma *et al.* [41] focus on developing new simulation frameworks and validation methodologies for ADAS. Sieber *et al.* [42] conducted research on driver perception and reactions when avoiding collisions, and discussed the implications on the development of ADAS systems.

Tuncali *et al.* [43] have proposed an approach based on stochastic optimization techniques to automatically generate test cases that lead to collisions. They start by sampling the parameter and input space and generate an initial state configuration. After selecting a proper robustness criterion and using the parameter space as an input, the framework is able to iteratively find faulty behavior regions and minimize a given cost function by applying various optimization techniques. The proposed framework is called S-TaLiRO [44]. Kapinski *et al.* [45] have given a great overview and comparison of other similar tools: Breach [46], RRT and S3CAM, that utilize similar optimization methodologies for falsification of embedded control system. In the article, they give an in depth state-of-the-art overview and problem statement for testing and verification of ECUs, focusing on current and emerging approaches.

Another method has been proposed by Abdessalem *et al.* [47]. They conducted research on a visual emergency breaking ADAS system detecting pedestrians. The main idea was to use neural networks to model the ADAS behavior and use the model instead of the real simulation to get an evaluation and confidence level output. By using genetic optimization methods, they were driving the system to faulty behavior. In addition, by utilizing the neural network model, they were able to reduce the total number of simulation evaluations leading to faster discovery of faulty behavior. However, one limitation of this method is that expert knowledge in neural networks is necessary to build a satisfactory model. Also, the network needs to be trained and validated beforehand with a significant amount of real simulation data, which presents a time-consuming and sometimes unfeasible task.

The main difference between the method proposed in this chapter and the research described above is that we build a surrogate model during the simulation runs by applying an iterative zooming-in approach. No training, nor a priori knowledge of the system, nor data preparation is needed. In addition, since we are using the surrogate model as the input for the optimization algorithms, the complexity of the real simulation will not directly impact the optimization speed.

3.3 Problem Statement

For the problem statement we are using the notation presented in Kapinski *et al.* [45]. Some parts are modified to better fit the proposed algorithm. Firstly, we denote the model we want to test with M. The model M is not necessarily a simulation model; it can also represent a real system running on the vehicle or on a HIL/VIL setup. Furthermore, it is also not limited in complexity as it can represent the whole autonomous driving system or any part of it. Next, we define the parameter space P which contains all the environmental (external) or model (internal) parameters and it represents an infinite search space. Usually, a new set of inputs U is introduced; however, we are going to assume that the input signals can be parameterized and the shape can be varied by changing the adequate parameters, in which case the parameter space P contains U, i.e., $P \supseteq U$. Finally, the testing criterion is denoted with ψ.

In general, each model M exhibits certain behavior during the simulation or real world trial. This behavior is denoted as $\Phi(M, p)$ of the model M with respect to the set of parameters $p \in P$. $\Phi(M, P)$ represents the behavior of M in respect to all possible variations of parameters in the parameter space P. If some behavior $\Phi(M, p)$ satisfies the criterion ψ, the system is working correctly, and we can write $\Phi(M, p) \models \psi$. In contrast, if ψ is not satisfied, we write $\Phi(M, p) \not\models \psi$.

As we are not able to test the whole parameter space, either because we do not know all the parameters or because of the complexity of the problem, we need to define a subset of the parameter space $\hat{P} \subseteq P$, on which the test is going to be carried on. To test a system, we need to determine whether $\Phi(M, p) \models \psi$ holds. However, this task turns out to be very challenging. The main problems are the curse of dimensionality - testing becomes exponentially more complex with each new introduced parameter, the evaluation complexity and evaluation time. One possible solution, proposed by [45],

[43], [47], is to find a set of parameters $p \in \hat{P}$ such that the $\Phi(M, p) \nvDash \psi$, i.e., a testing instance where the evaluation criterion is not satisfied. By limiting our search for a specific set of parameters $p \in \hat{P}$, we can vastly improve the speed and avoid exploring regions of the search space $\hat{P}$ that are of no interest.

In order to evaluate the behavior $\Phi(M, p)$, we need to introduce some kind of cost function based on the evaluation criterion ψ, which is going to generate numerical evaluations based on the behavioral performance. We can denote such a cost function with $c_\psi(\Phi(M, p))$. By selecting an appropriate cost function c_ψ, it is possible to guide the testing towards regions where the behavior is not satisfactory and where the evaluation criterion ψ is not satisfied.

It is also important to note that the selection of an appropriate evaluation criterion ψ, cost function $c_\psi(\Phi(M, p))$, parameter space $\hat{P}$ and initial state $p_0 \in \hat{P}$ is a non-trivial task and all of them come with their own challenges. Nevertheless, the outcome of the testing will heavily depend on the quality of the chosen values.

3.4 Concept Overview

In the previous section, we gave an overall problem statement and mentioned that a cost function $c_\psi(\Phi(M, p))$ is needed in order to evaluate the behavior of the model M. However, because of the complexity of the model or long simulation duration, it is not always feasible to run the simulation and evaluate the cost function directly. This limitation is a big problem if we want to find a global minimum of the cost function, as all the methods searching for a global minimum require many function evaluations. In this research, we propose an approach that relies on a surrogate model which approximates the cost function $c_\psi(\Phi(M, p))$ with a new function $\hat{c}_\psi(p)$. The new function $\hat{c}_\psi(p)$ takes a parameter set p as input and outputs an approximated numerical value $\hat{c} \approx c_\psi(\Phi(M, p))$ which is improved iteratively, giving a good representation of the real cost function in the region of interest.

Figure 3.2 gives an overview of the proposed method. As an input to the algorithm, we need to provide a search space $\hat{P}$ and the Zoom-in sampler is going to make an initial sampling of the space and invoke the *Simulation Engine*. The number of initial samples n_s, zooming factor z_f, zooming iteration number z_{it} and sample randomness factor r_f present the only input parameters for the algorithm.

The *Zoom-in Sampler* will make an initial n_s x n_s grid on the parameter borders and scale it down using the z_f for each iteration. The randomness factor r_f moves the sampling points in the vicinity of the original position, if the simulation is run for more than one time, leading to a better overall approximation.

The parameter selection for the proposed algorithm, surrogate modeling and optimization will be discussed later in more detail. The cost function $c_\psi(\Phi(M, p))$ is going to be evaluated for each parameter set $[p_1...p_n] \in \hat{P}$ generated by the sampler and an output vector of numerical values $[c_1...c_n]$ is going to be computed. For each iteration, we can evaluate the numerical values $min\{[c_1...c_n]\} < c_{thresh}$ and decide if a faulty region has been reached. Usually, the cost function $c_\psi(\Phi(M, p))$ can be modeled in

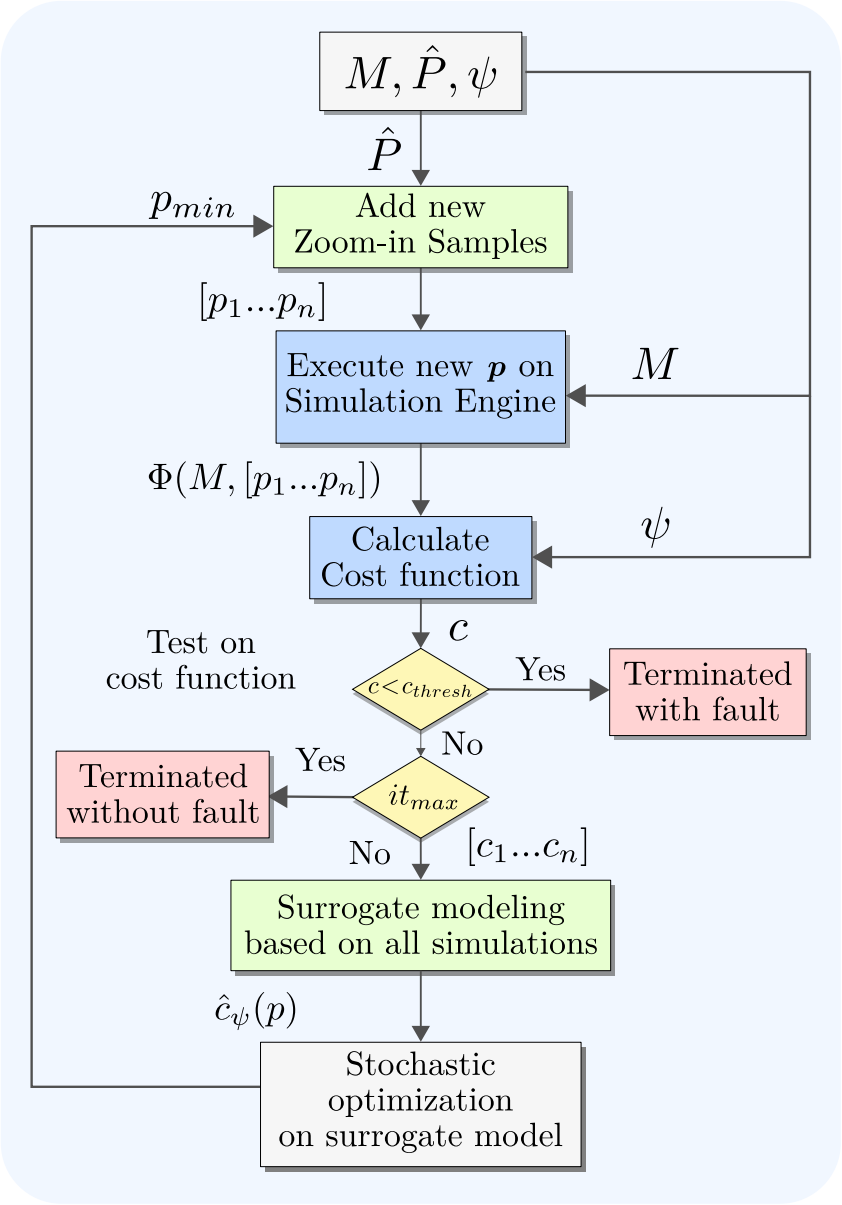

Figure 3.2: *Workflow of the proposed method © IEEE 2017*

such a way that a negative value represents a faulty behavior, i.e., $c_{thresh} = 0$; however, that is not mandatory and any kind of value for c_{thresh} can be used.

If the faulty behavior has not been found and if the maximum number of iterations has not been reached, the numerical evaluations of the cost functions $[c_1 ... c_n]$ are passed to the surrogate modeling block. The surrogate model is iteratively extended using the values of $[c_1 ... c_n]$ and provides the approximation function $\hat{c}_\psi(p)$ to the stochastic optimization block where various optimization algorithms can be used. The evaluation of the approximated function $\hat{c}_\psi(p)$ is computationally much cheaper than the original function and is suitable for the extensive evaluations done by the optimization algorithms. Stochastic optimization block outputs p_{min} representing the most likely location for the global minimum of the approximated function $\hat{c}_\psi(p)$.

In the next iteration, the *Zoom-in Sampler* reduces the size of the new sampling set $[p_1...p_n]$ using the r_f coefficient and moves the center to the value of p_{min}, effectively zooming into the region with the lowest value of the cost function. A new evaluation is done with the new parameters, and a better model of the approximated function $\hat{c}_\psi(p)$ is built until the algorithm reaches a faulty behavior or the maximum number of iterations. After several zooming-in iterations, the size of the sampling set n_s x n_s is significantly reduced and the model accuracy does not increase. In order to further minimize the number of function evaluations, only the most likely location for the global minimum is considered for iterations higher than z_{it} without using the randomness factor r_f.

3.5 Surrogate Modeling

For the surrogate modeling, we decided to use the Radial Basis Function (RBF) approximation. The main idea of RBF is to find an estimation $\hat{f}(x)$ of a real system or process $f(x)$ by sampling the function at samples $x_i \in X_s$ and assigning a radial basis symmetrical kernel function ϕ for each sample. By adding the influences of each kernel, the approximation can be constructed. The equation of the estimation is given as:

$$\hat{f}(x) = W^T \Phi = \sum_{i=1}^{n_s} w_i \phi(\|x - x_i\|) \tag{3.1}$$

where $w_i \in W$ represent the weights corresponding to each kernel function. The approximated function has the following properties: $\hat{f}(x)$ is equal to $f(x)$ for $x_i \in X_s$ and the approximation is worse as the distance from the sample increases. Weights w_i can be obtained by solving the system of linear equations

$$W = \Phi^{-1} Y \tag{3.2}$$

where $Y = f(x), \forall x \in X_s$ and $\Phi = \phi(\|x_i - x_j\|), i, j = 1...n_s$ where Φ is also called the Gram matrix. For the introduction of RBF, we used notation consistent with literature and the Gram matrix Φ should not be confused with the system behavior $\Phi(M, p)$ from the problem statement.

There are many possibilities when choosing the kernel function for the RBF surrogate modeling. However, in our use case, we tested the Gaussian (ϕ_g) and Multiquadric (ϕ_{mq}) kernel.

$$\phi_g = e^{\frac{-\|x_i - x\|^2}{2\sigma^2}}, \ \phi_{mq} = \sqrt{\|x_i - x\|^2 + h} \tag{3.3}$$

When building a model with a Gaussian or Multiquadric kernel, the user needs to manually assign the coefficients σ or h. The selection of those coefficients is a non-trivial task, especially when new zoomed-in sampling points are added in each iteration.

In order to overcome this limitation, we decided to use Kriging models [48] as they provide a way to use optimization tools in order to find the appropriate coefficients.

The Kriging model also uses the Gaussian kernel function ϕ_g but a different coefficient $\gamma = \frac{1}{2\sigma^2}$ and norm p (usually $p \in [1, 2]$) are computed for each parameter. Even though Kriging models are more complex and use more resources for construction, the effort clearly pays off since no tuning is needed. Furthermore, the coefficients γ show which parameter has the higher influence on the cost function. The higher the value of γ for a particular parameter, the higher the influence on the cost function. This can be useful in cases with high number of parameters where the parameters with lower influence could be fixed and excluded from the search.

An additional benefit of using Kriging models is that, instead of searching for the global minimum of the model, we can use a probabilistic approach in order to find the most likely location of the global minimum [48].

Beside the most likely global minimum search, the Kriging models require another optimization step for finding adequate parameters γ and p. The value of p was fixed to $p = 2$, as proposed in [48], leading to a simpler optimization task for finding γ. It is important to state that the limitation of all surrogate models is that the cost function needs to be smooth in order to achieve the best modeling results and save computation time.

3.6 Stochastic optimization

Stochastic optimization represents a family of optimization methods where some form of randomness is present. This randomness can either stem from the objective function, constraints or the optimization algorithm itself. In this research, we focus on the optimization algorithms that use randomness in order to accelerate the search progress [49]. These methods are extensively used in global optimization where they are able to escape local minima and provide good results in a wide range of problems.

As we don't have any knowledge of the shape of the cost function c_ψ and there is no guarantee that is doesn't contain local minima, we have to apply global optimization algorithms in order to minimize the cost function and locate the faulty behavior.

For the optimization tasks, we will use two well known search algorithms. Differential Evolution (DE) [50] and Particle Swarm Optimization (PSO) [51].

3.6.1 Differential Evolution

Differential Evolution is a nature inspired search algorithm that does not rely on gradients or any related information from the cost function, which makes it suitable for a wide variety of cost functions as they don't have to be continuous, differentiable or even deterministic. The only input the algorithm takes are the evaluations of the cost function at specific points. These points are determined by the so called agents or candidate solutions.

The algorithm starts by randomly placing $n \in N$ agents in the parameters space $\hat{P}$. After evaluating the cost function at each agent's initial position, the optimization algorithm starts. For each agent n_i, three random and distinct agents (k, l, m) are

chosen. Then, a sample is taken from an uniform distribution $r_i \sim U(0,1)$. If $r_i > c$, where c is the crossover coefficient, then the agent position is calculated as $n_i = k + f(l - m)$ where f is the crossover coefficient. Next, the cost function c_ψ is evaluated on the new position n_i. If the new position is better then the previous one then the agent is updated, else it is discarded. The number of agents N and coefficients c, f are chosen by the user and they will impact the performance of the algorithm.

3.6.2 Particle Swarm Optimization

Similarly to DE the Particle Swarm Optimization algorithm is also based on behavior that appears in nature. More precisely, it represents a simplified representation of the movement of a bird flock or fish school. The main idea behind the algorithm is that each particle will track the best position it has visited, but also take in the consideration the best position from the whole swarm. If a particle has found a better location then the current best, all other particles will be aware of this fact. This effect is also called swarm intelligence. By using both the best local position and the best position of the swarm the particle is able to balance exploration and exploitation. Also it is common that the velocity of the particles is decreased with the number of iterations which makes the particles group at the minimums.

The same as for DE, PSO also does not depend on any information from the cost function and can be applied on a wide variety of problems.

The algorithm starts by randomly placing an initial population (swarm) of particles, also known as candidate solutions. Each particle is also assigned a random initial velocity. For each particle $n_i \in N$ pick random numbers $r_1, r_2 \sim U(0,1)$ and update the particle's velocity $v_i = \omega v_i + c_1 r_1(n_{ib} - n_i) + c_2 r_2(s_b - n_i)$. ω is used to lower the velocity with the number of iterations. c_1 is the cognitive coefficient, and it will determine how much will the particle focus on it's own best position n_{ib}. c_2 is the social coefficient and it determines the effect of the best position that the swarm has found s_b. After updating the velocity, the particle position is updated with $n_i = n_i + v_i$. The cost function is evaluated on the new position. If the new position is better than the current best it will take it's place. Also, if the new position is better than the current best for the whole swarm it will take it's place. The algorithms continues until some termination criterion is met.

It is important to mention that neither of these two algorithms guaranties to find the global minimum. Moreover, overall execution time of the proposed algorithm depends strongly on the parameters, *i.e.*, number of agents and number of iterations, as the optimizers are used several times for each iteration.

3.7 Case Study and scenario description

In order to validate the proposed method, a simple highway scenario was used. The testing is done for an Emergency Brake Assistant (EBA) system. The scenario consists of a passenger car driving on the highway and encountering an obstacle in its path. The goal of the EBA is to avoid collisions by braking, i.e., no evading maneuvers are

used. To prove that our system is able to detect faulty behavior inside the parameter space, we introduce an error in the sensor's field of vision. These types of errors can occur from external mechanical damage, internal circuit failure or occlusion.

A detailed overview of the scenario can be seen in Figure 3.3. The vehicle starts from a still stand and accelerates with constant acceleration until it reaches a maximum velocity of $100\frac{km}{h}$, moving along the x axis. Simulation duration is 10 seconds at which the car reaches 128 m. The parameter space $\hat{P}$ is represented by the obstacle's position (p_1, p_2), leading to a 2D search space. Static obstacles are placed on the vehicle's path within the following boundaries: $\hat{P}_{min} = [25, -12]$, $\hat{P}_{max} = [165, 12]$. The distance at which an obstacle is detected is fixed to $d_{sensor} = 20\ m$ and the sensor's detection angle is $\alpha = 30°$ ranging from $[-15°, 15°]$. An error $\theta = 1,5°$ is introduced starting at 1°, and a search for the worst case crash is going to be conducted.

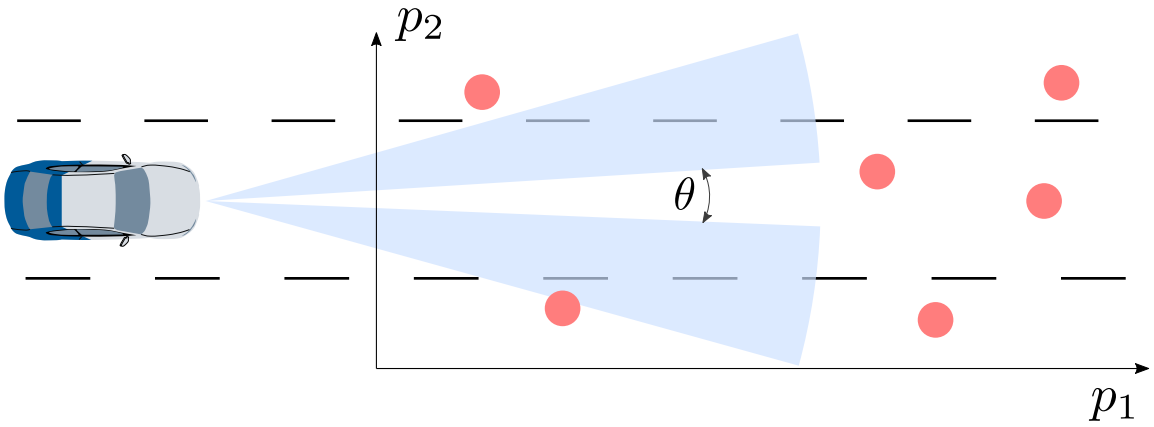

Figure 3.3: *Scenario setup © IEEE 2017*

3.7.1 Simulation setup

For the simulation setup Matlab and Simulink [52] are used. The vehicle dynamics and ADAS model represent the model M from the problem statement. The simulation setup overview can be seen in Figure 3.4. In order to use the proposed approach, only a helper function needs to be available to run the simulation model with parameters $p \in \hat{P}$ and receive back the evaluated cost function $c : c_\psi(\Phi(M, [p_1...p_n])$.

3.7.2 Cost function selection

As discussed before, the selection of the cost function c_ψ is not a trivial task. Nicolao *et al.* [53] have introduced an approach where they propose a risk assessment evaluation based on the probability that a collision with a pedestrian will occur. Ferrara *et al.* [54] have used a collision cone approach where they explore necessary and sufficient conditions for a collision to occur. Some good practices that can be considered when constructing a cost function are: smoothness - as it will enable a better approximation when using surrogate modeling and convexity - this will ensure quicker convergence to the global minimum.

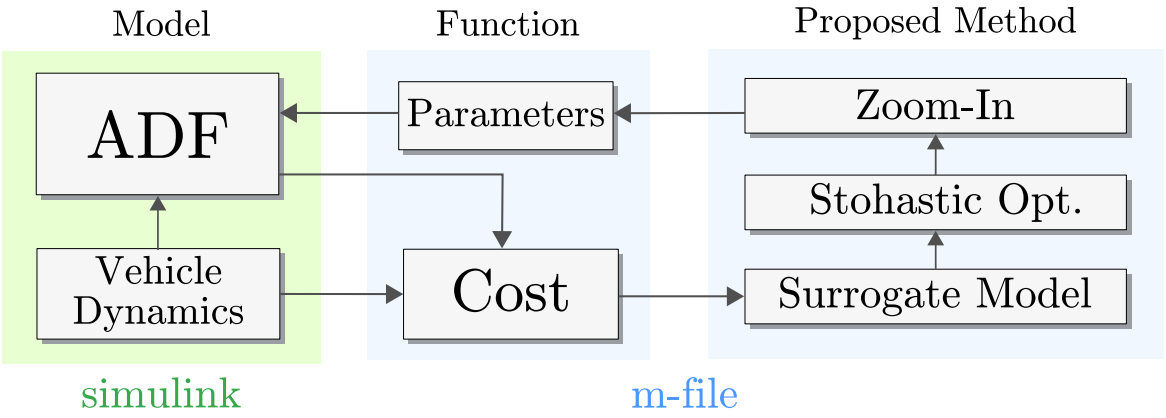

Figure 3.4: *Simulation setup using MATLAB © IEEE 2017*

For our case study, we are going to use a cost function based on the Time to Collision (TTC) between the vehicle and the obstacle. The scenario duration is set to 10 seconds an the lowest value of the cost function is taken. However, defining the cost function only with the TTC can lead to test runs were the cost function is misleading. An obstacle can be close to the vehicle (on the side) but as it is not on the path of the vehicle it is not critical and a higher cost can be assigned. To avoid this behavior, the vehicle velocity v_{veh} is added in order to give a higher cost value to the obstacle that is not in the vehicle path or it is out of the sensor range.

Finally, we also define that crashes that occur at higher velocity have a lower evaluation value then crashes at lower velocities. The complete cost function is defined as:

$$c_\psi = \begin{cases} min(TTC + v_{veh}),\ TTC = \frac{d - d_{min}}{v_{veh} + \varepsilon} & , no\ collision \\ -v_{veh} & , collision \end{cases} \tag{3.4}$$

where ε is a small value used to evaluate TTC when the velocity is equal to zero. By minimizing the cost function, we aim to find the most severe crash conditions. The complete cost function on the parameter space $\hat{P}$ is shown in Figure 3.5. In the figure we can distinguish 3 regions. The outside red shaded region represents the obstacles that were outside the sensor range. They do not pose a threat to the vehicle so the cost function is high in this region. The second region is in a light blue color. This region corresponds to obstacles that were detected and the vehicle was able to decelerate and stop. And finally, the dark blue region represents the fault in the senor. The obstacles were not detected and a crash occurred. Furthermore, crashes with higher velocities have lower cost values.

3.8 Results

In order to evaluate the results obtained from our algorithm, we have simulated the same scenarios using the optimization algorithms DE and PSO directly on the sim-

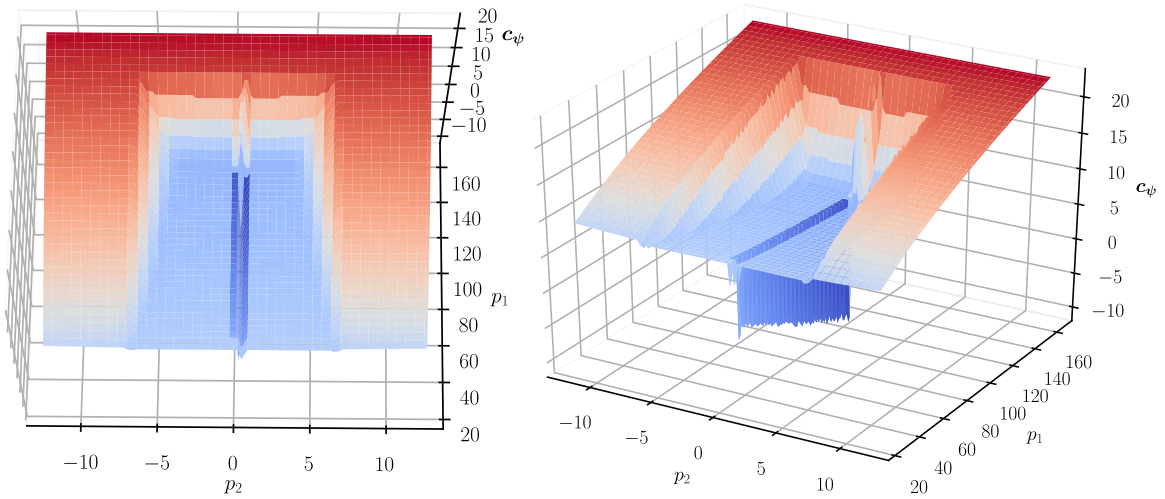

Figure 3.5: *Cost function defined in equation* (3.4)

ulation model, without building the surrogate. The aim of this research has been to reduce the overall number of function calls as they can have a long execution time or could be expensive regarding computation or other resources. Since we are dealing with stochastic optimization algorithms, we can not be certain on the number of function calls needed to find a global minimum. Therefore, we have conducted our experiment with 100 test runs and gathered the average values for comparison. In addition, the functions and parameter spaces are equal for all methods used.

Because the optimization algorithms do not save states between evaluations, we built the surrogate model for every test run, *i.e.*, the samples were not saved between the test runs, even though that would be beneficial in a real scenario. The number of function evaluations was limited for all algorithms and was set to 100 evaluations per test run. The optimization algorithms were limited to use 5 agents and 19 iterations with an additional 5 evaluations for the initialization. Coefficients in the DE algorithm were set to $c = 0.5$ for the mutation and $f = 0.5$ for the crossover. For the PSO algorithm, the coefficients were set to $c_1 = 1.05$ for cognitive, $c_2 = 1.05$ for social and $w = 1 \rightarrow 0.1$ for the particle speeds. The parameters were chosen taking in account advice given in [55] and [51], respectively.

Our proposed method was running using $n_s = 3$ leading to 9 samples per zooming iteration for iterations below $z_{it} = 4$ and was limited to maximum 100 function evaluations. The zooming-in factor was set to $z_f = 0.35$ and the random factor was set to $r_f = 0.1$. In general, a higher value of n_s leads to better surrogate model but it will also lead to higher number of required real function evaluations per iteration. The zooming-in factor z_f should be chosen depending on the cost function. If the cost function is convex then a lover value is better as it will lead to faster convergence and a good interpolation around the faulty behavior. However, if the cost function is unknown and may have many local minima, a higher value is recommended as the model will have better overall approximation of the cost function. If several test runs are possible, it is better not to sample the model at the same location and reasonable values for the random factor are $r_f \in [0, 0.5]$.

For the surrogate model optimization tasks, we used the PSO optimization algorithm. However, because we are not calling the objective function and the computation is reasonably inexpensive, we used a higher number of agents (15) and iterations (15).

The experiment was repeated for 100 times within the parameter space $\hat{P}_{min} = [25, -12]$, $\hat{P}_{max} = [165, 12]$. The testing criterion $\psi : TTC \leqslant 0$ means that a collision has occurred.

Table 3.1: *Comparison of different algorithms for test run reduction © IEEE 2017*

	f-avrg	f-best	g-best	found	a-f-call	time
Kri	-5.27	-11.43	111.2, 0.24	97%	42.67	1785s
R_g	-2.70	-13.78	106.7, 0.20	70%	57.70	689s
R_{mq}	-3.29	-12.04	112.7, 0.23	84%	51.24	901s
PSO	-5.08	-12.19	99.18, 0.21	86%	53.70	623s
DE	-2.37	-13.03	92.20, 0.20	67%	59.90	345s

Table 3.1 presents the obtained results. The columns of the table are respectively: average evaluated global minimum; the best global minimum; position of the best global minimum; percentage of successfully found crashes from all test runs, i.e., Krigin model found crashes in 97 out of 100 runs; average number of simulation evaluations per test run; and computation time for all test runs. The rows show the results for the Kriging, RBF with Gaussian and Multiquadric kernel, DE and PSO optimization algorithms.

We can see that the Kriging model managed to find test runs leading to crashes with the highest probability while using less simulation evaluations. The higher computation time of the Kriging model is directly related to the two optimization steps needed in each iteration; however, in our setup, one simulation run lasted on average 0.127s, and the benefits of a reduced number of function calls could be seen on simulations with longer execution times.

3.9 Chapter Conclusion

In this chapter we introduced an iterative falsification approach for ADF, focusing on finding faulty behavior inside a given parameter space. Because of the simulation duration or complexity, it is not always feasible to run global optimization algorithms directly on the system. We proposed an approach where a computationally inexpensive surrogate model of the system behavior is built, and optimization algorithms are then applied on the surrogate and not the real system.

For the surrogate modeling we used the RBF approximation, and we have explored models with different kernel functions. For the optimization tasks the DE and PSO were implemented. The testing evaluation was conducted on a highway scenario with an EBA. The scenario consisted of a passenger car driving in a straight line and the

obstacle position was varied. An error in the sensor's field of vision was introduced and the task of the algorithms was to find the test case with the worst crash evaluation.

The aim of the research was to show that we can reduce the number of test runs if we first build a surrogate model and then run the optimization algorithms on the surrogate and not on the real system. In the evaluation, the maximum number of real system evaluations was fixed to 100 for all algorithms and the average outcomes were compared. In the end, we have shown that the Kriging model produced the best results leading to a lower number of test runs and a good approximation inside the faulty region.

4

Model-based safety validation

In the previous chapter we introduced a methodology which is used to lower the number of test runs and lead the testing towards faulty behavior regions. However, we are not able to guarantee that the system is working without errors if none were found. Therefore we can make use of simulated models, where the number of test runs is not an issue, to perform more thorough testing. This in return makes it possible to provide guarantees that the system is behaving correctly in a given parameter range.

Most of the content of this chapter is adopted from

H. Beglerovic, A. Ravi, N. Wikström, H.-M. Koegeler, A. Leitner, J. Holzinger, Model-based safety validation of the automated driving function highway pilot, in: 8th International Munich Chassis Symposium 2017, Springer, 2017, pp. 309–329.

4.1 Introduction

In this chapter, we will show a validation approach applied to a specific scenario chosen among 1000 relevant traffic scenarios for a Highway Pilot. By selectively and systematically varying scenario parameters, it is expected that a better coverage is achievable compared to classical full-factorial validation. However, this approach is only feasible if adequate virtualization tools and Model in the Loop (MIL)/SIL environments are available.

According to accident databases like German In-Depth Accident Study (GIDAS) [56], most highway accidents in recent years have occurred when driving in the same direction on the same lane, at sunny weather and dry road conditions.

Consequently, the first automated driving use cases focus on Highway Pilot functionality [57] – allowing a SAE level 4 automation [58]. In general, the Highway pilot is a combination of several complex ADAS functions such as: ACC, LKA, and LCA. Each of these ADAS functions individually, as well as their integration, presents a challenge for validation and verification. In the scope of this research we will focus on the ACC part of the Highway Pilot in order to focus on the "following behind a car in the same lane" type of scenario, which have the highest accident probability. To keep the confidentiality of manufacturers, we put ourselves in the position of an

ADAS-Function supplier and develop it from the ground up all the way to the first tuning and validation.

4.2 Adaptive Cruise Control

In this section we will go through the implementation of the ACC function. The objective of the investigated control strategy is to allow fuel-efficient, comfortable and safe longitudinal speed profiles for the ego vehicle on a highway road, presenting the first and fundamental part of a Highway Pilot.

A Model Predictive Control (MPC) based strategy is employed, which adaptively controls the acceleration of the vehicle based on information from the prediction models. These models recurrently predict the motion of the ego vehicle as well as a preceding target vehicle over a prediction horizon of 20 seconds. This information is then used as inputs to a quadratic programming optimization problem, which takes the form

$$min_x \;\; x^T Q x + q^T x \tag{4.1}$$

subjected to the vehicle constraints, traffic speed limits.

$$\begin{aligned} 0 \leqslant v &\leqslant min(v_{max}^{defined}, v_{max}^{legal}, v_{max}^{curve}) \\ a_{min} &\leqslant a \leqslant a_{max} \\ j_{min} &\leqslant j \leqslant j_{min} \end{aligned} \tag{4.2}$$

where x denotes the state of the system, which includes: v - velocity, a - acceleration and j - mechanical jerk. The cost matrix Q and cost vector q^T describe the weightings between the different state variables. The cost matrix Q is positive definite diagonal matrix defined as

$$Q = diag(c_{v2}, c_{a2}, c_{j2}), \;\; c_{v2}, c_{a2}, c_{j2} > 0 \tag{4.3}$$

and q^T is defined as

$$q^T = [c_{v1}, c_{a1}, c_{j1}] \tag{4.4}$$

where the symbol c denotes the coefficients for the respective state variables. The optimization problem is solved every $500ms$ to obtain desired vehicle accelerations. The parameters were tuned in an initial phase for a single route with fixed traffic and environment conditions and will be discussed in detail in the next section.

In addition to the cost given in equation (4.1), two headway costs are added to influence the following behavior of the ego vehicle. The ego vehicle is allowed to adapt its headway distance to the preceding vehicle within a predefined, velocity-dependent range. This is referred to as a flexible spacing policy. The policy is defined with the minimum and maximum headway distance.

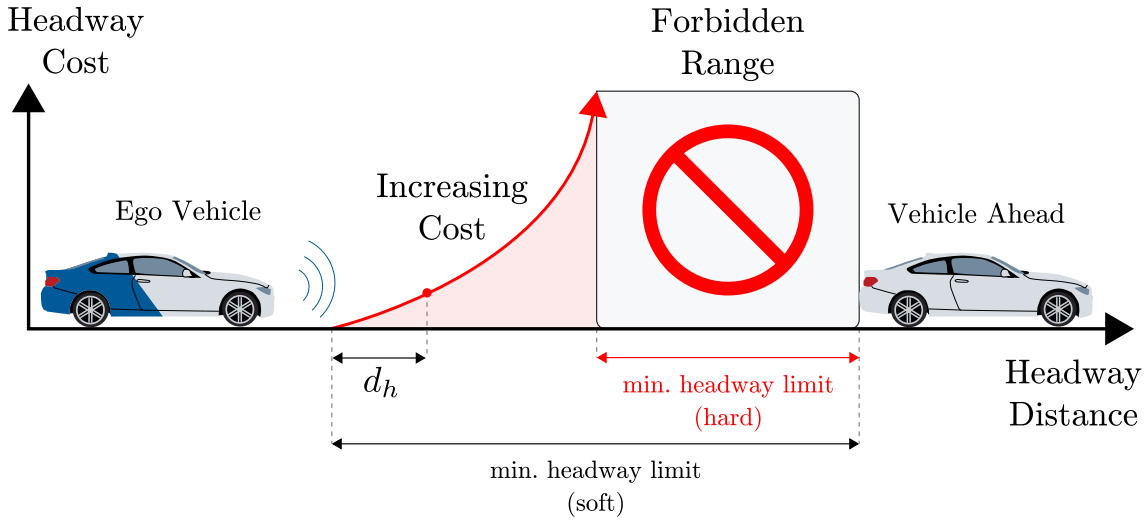

Figure 4.1: *The minimum inter-vehicle distance is given by two headway time limits: the first (soft) limit may be violated at a cost whereas the second (hard) limit should never be violated.*

Figure 4.1 shows how the minimum distance is defined by two headway limits: a hard and a soft limit. The hard limit must never be violated and it is added to the constraints, whereas the soft limit can be violated but at a given cost. This cost grows quadratically approaching the hard limit and it is defined as

$$cost_{headway} = c_{h2} d_h^2 + c_{h1} d_h \tag{4.5}$$

The maximum headway distance d_{MAX}, illustrated in Figure 4.2, is also composed of two components. The first component, d_{MAX}^{hr}, is a predefined limit given either as a fixed distance or a velocity-dependent distance. The second component, $d_{MAX}^{CatchUp}$, is proportional to the preceding vehicle's current velocity, scaled by a 'catch-up' factor defined to be greater than one. The first component dominates when the ego vehicle is within the desired headway range, which corresponds to a following maneuver. The second component dominates when the vehicle is far behind and it should close the gap to the preceding vehicle. Neither of these constraints are allowed to conflict with the speed limit on the route, and lagging behind the limits leads to a higher travel time cost.

$$cost_{travel} = c_{tr} t \tag{4.6}$$

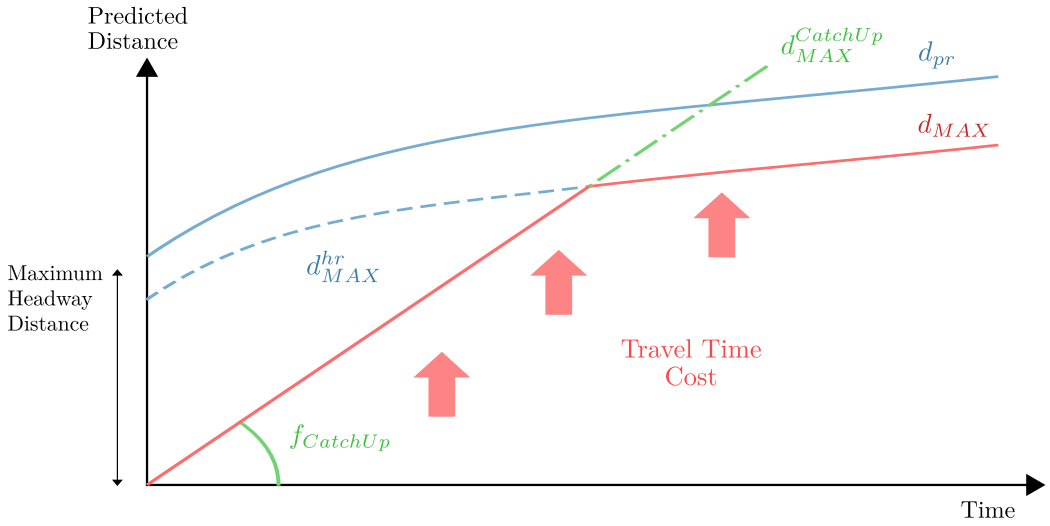

Figure 4.2: *The maximum inter-vehicle distance d_{MAX} is defined by two limits: one limit dominates when the ego vehicle is located far behind the preceding vehicle ($d_{MAX}^{CatchUp}$), while the other dominates after a predefined headway range has been reached (d_{MAX}^{hr}).*

4.3 Model-based tuning

In order to reach a target behavior of the ACC, or in more general terms the SUT, the relevant influencing parameters have to be determined and varied systematically. Typically the first development environment in the development process is a MIL, where the ACC function is simulated together with the environment.

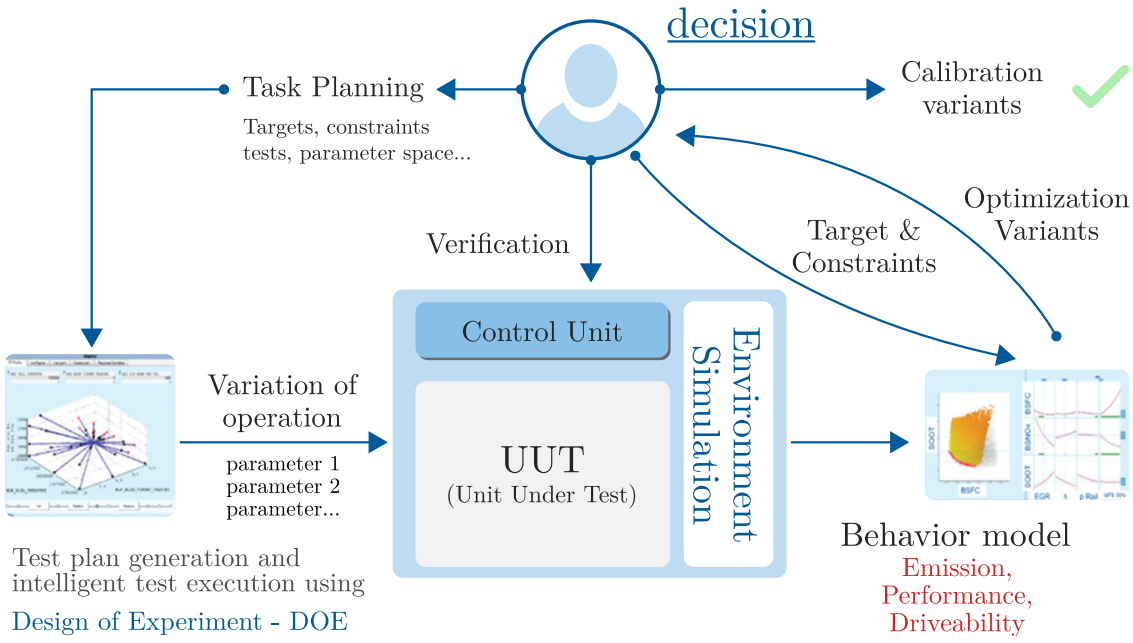

Figure 4.3: *Model-based tuning of a SUT*

In general, this tuning task is carried out by an expert operator, who, based on his experience, varies the most relevant parameters until the target behavior is met. The results from such a subjective tuning process depend on: start values, step size, number of considered parameters, etc. Without a good understanding of the underlying correlations between the parameters, a perceived optimum behavior could easily represent a local optimum.

To improve that, a statistical Design of Experiments (DOE) based tuning methodology is presented in Figure 4.3, which tries to provide a correlation of the input parameters and the KPIs using the least possible number of test runs. Using an automated test procedure, all the relevant input parameters are varied simultaneously to identify the system behavior. The observed behavior of defined KPIs is used to build models, which allow predictions in the whole space defined by the variation parameters and some limited extrapolation depending on the model quality. The method assures that a global optimum is found, the parameter space is optimally covered, and the efficiency as well as traceability of the tuning decisions are ensured. This statistical DOE approach helps to understand the parameter influence and correlations, which is very beneficial in cases with a large number of input parameters. In such cases, the parameters with lower influence can be kept constant or excluded, which reduces the tuning effort. Furthermore, the behavioral models built between the inputs and KPIs can be used for optimization tasks and tradeoff analysis, providing an additional tool for the adequate tuning of parameters.

The development environment for the SUT consists of the dynamic vehicle simulation program IPG CarMaker, Matlab-Simulink for the development of the ACC and AVL Cameo for the test automation, tuning and optimization.

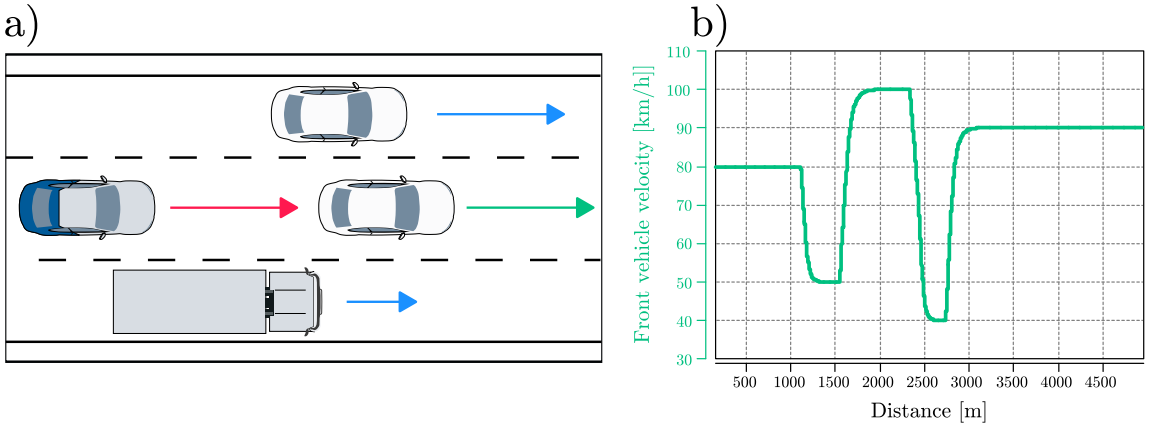

Figure 4.4: *a) Selected scenario for tuning; b) Reference velocity of the preceding vehicle*

Figure 4.4 shows the scenario selected for the tuning of the ACC function. It consists of an ego vehicle (red arrow) following a preceding vehicle (green arrow) using the model predictive ACC. The preceding vehicle has a predefined behavior as shown in Figure 4.4 b) and the controller is tuned to satisfy several KPIs with respect to the preceding vehicle behavior. The tuning task represents the first effort conducted for the new controller and it is placed early in the development cycle.

4.3.1 Tuning Targets KPIs

For the tuning task of the ACC controller the following KPIs have been chosen:

- Safety – The controller should not violate the hard constraints regardless of the maneuver. As shown in Section 1 the controller has a hard constraint for the headway time. This constraint must not be validated in order to leave enough time for the vehicle to come to a complete stop in case of an accident.
- Fuel Performance - The controller should minimize the fuel consumption measured in liters per 100km, and the influence of sudden changes in the preceding vehicle's velocity or maneuvers should be kept low. The fuel consumption when the ego vehicle perfectly follows the preceding vehicle with a fixed headway time is considered as a baseline.
- Driver Comfort – The controllers should maximize the driving comfort by reducing jerk. Similarly as above, the behavior of the preceding vehicle will be smoothed out and the controller will try to minimize braking while maintaining a safe and smooth ride.
- Travel Time – The controllers should minimize travel time taking in consideration all above mentioned KPIs.

4.3.2 Design Variables Sensitivity Analysis

As we are dealing with a new controller function without a priory knowledge of the behavior, we have taken the available 10 parameters (c_{v2}, c_{a2}, c_{j2}, c_{v1}, c_{a1}, c_{j1}, c_{h2}, c_{h1}, c_{tr}) and conducted a sensitivity analysis. Figure 4.5 shows the Relative Significance Indicator (RSI) for the four KPIs with respect to each of the ten investigated tuning parameters.

To calculate the RSI, the sensitivity algorithm analyzed the contribution of each individual parameter on the overall model quality. At the beginning we start without any parameters and calculate the Root Mean Square Error (RMSE) with respect to the mean, which is denoted by $RMSE_{max}$. Then, we calculate the RMSE by considering all parameters denoted as $RMSE_min$. Afterwards, the contribution of all individual parameters on the reduction of the RMSE is calculated and normalized using $RMSE_{max}$ and $RMSE_{min}$ leading to the RSI score. The parameter with the highest RSI score gets selected, its value gets fixed and the process starts over with the remaining parameters. These steps are repeated iteratively until all parameters are sorted by their RSI score for each KPI.

A Robust Neural Network (RNN) [59], which also considers nonlinear influences and interactions correspondingly, is used as modeling base. We conclude that only the first five parameters, (c_{j1}, c_{a2}, c_{j2}, c_{v1}, c_{a1}), have a considerable influence on the overall behavior of the system. Therefore, for further tuning the rest of the parameters were kept constant.

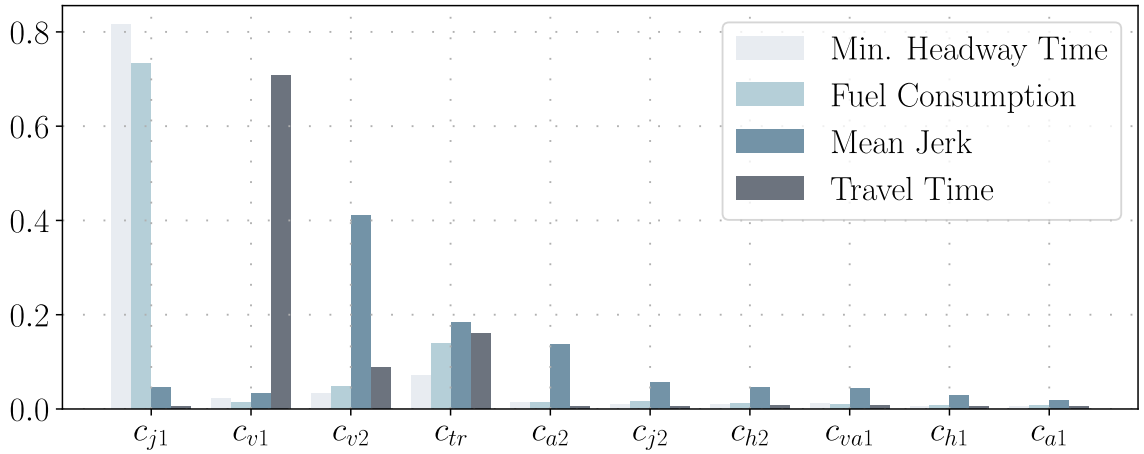

Figure 4.5: *Parameter sensitivity analysis*

4.3.3 Tuning Results

The tuning of the controller has been conducted with an interactive DOE procedure. The start design of 40 test runs for the 5 parameters was automatically extended to 520 test runs in order to get reliable models in regions where the KPI show interesting behavior [60], [61]. Figure 4.6 shows the resulting tuned ACC-behavior: The upper two signals represent the velocities of the preceding and ego vehicle respectively. The preceding vehicle's velocity profile was fixed for the whole tuning task and the ego vehicle's ACC controller was tuned in order to assure that the safety, fuel consumption, comfort and travel time KPIs are met. The next two signals represent the road elevation profile and the clearance between the ego and the preceding vehicle. We can see that the controller settings are chosen in such a way that robustness is assured and outside disturbances coming from the road elevation are successfully handled. Finally, the two remaining signals represent the accelerator and brake pedal position percentages, and we notice that the controller settings minimize the braking energy and jerk, leading to more efficient and comfortable ride.

For the evaluation of the tuning, we compared the behavior and consumption of the ego vehicle with respect to the preceding vehicle's velocity profile. The controller was able to achieve approximately 18% decrease in fuel consumption whilst still maintaining a smooth and safe ride. In the next section, a validation of the tuned control function will be performed in order to determine the robustness of the chosen parameters with respect to different behaviors of the preceding vehicle.

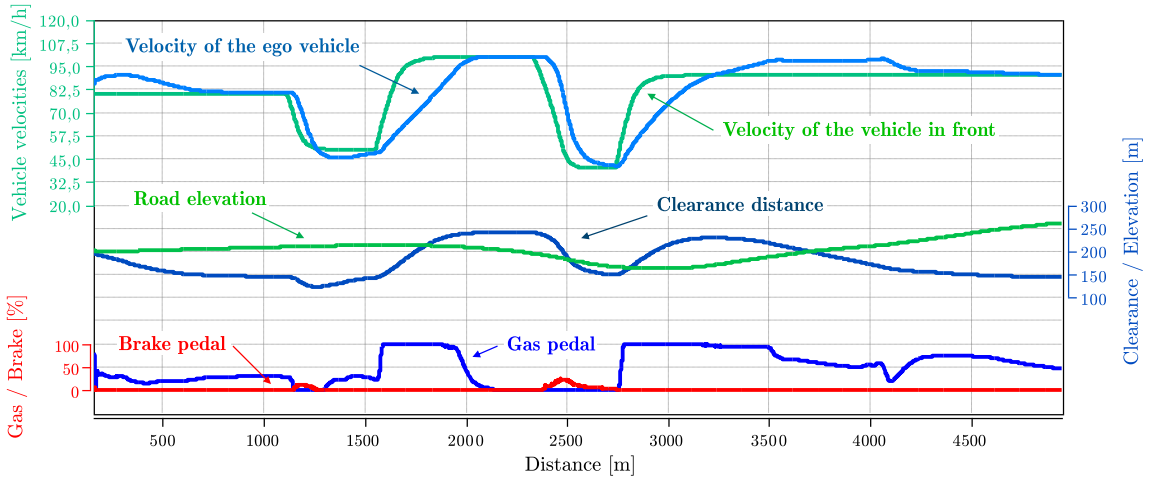

Figure 4.6: *Tuning results of the reference scenario*

4.4 Model-based validation

After successfully tuning the SUT, a validation procedure is needed in order to prove that satisfactory behavior and robustness is assured in a wider range of maneuvers. Figure 4.7 shows an overview of the validation task and we notice a considerable similarity to the tuning task discussed previously. In this research, we aim to prove that the same principles of model-based tuning can be transferred to the model-based validation of ADFs. The main difference between the tuning and validation is that now the external or environmental parameters are varied in order to validate the internal or controller parameters. Similar to the tuning task, a parameter space coverage is determined using DOE after which behavior models are built for all KPIs. The most important benefits of this approach are:

- Compared to the "full factorial grid approach" the behavior of the safety-critical KPIs can be predicted with an order of magnitude less simulation runs, giving complete results in the investigated ranges.
- Interactions and significance of the parameter changes are fully considered when searching for safety-critical situations.
- The highly automated process is much faster than a manual one, and the human interaction happens on the level where decisions are made. In addition, a new interactive DOE procedure [59]is used which fills in points automatically after an initial sampling in order to improve the accuracy of the models in regions of high interest.

By selecting appropriate KPIs, it is possible to locate safety-critical behavior regions in-side the parameter space by optimizing the models built in the previous step. Faulty behavior of an ADF in a test run can be defined as the behavior occurring when a critical criteria is not satisfied, e.g., the minimum clearance distance was breached. For a function developer it is of high interest to investigate the region in more detail and to get a better understanding of the faulty behavior. This can be done automatically using the interactive DOE procedure.

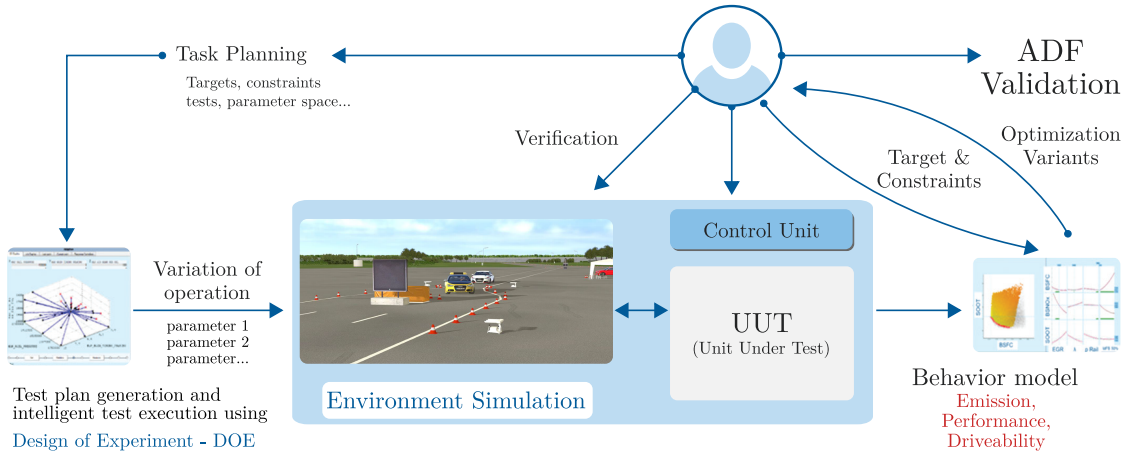

Figure 4.7: *Model-based validation of a SUT*

4.4.1 Scenario Selection for Validation

The scenario selected for the validation of the Highway Pilot can be seen in Figure 4.8. In this scenario the ego vehicle is driving with the desired cruising velocity and a heavy truck is cutting in from the right side with a much lower speed. After cutting in, the preceding vehicle's velocity is changing following a sinusoidal trajectory where the mean velocity, amplitude, and period are varied as validation parameters. The length of the test run is fixed to 3km. In addition, this type of scenario represents the second most likely accident type defined by the GIDAS [56].

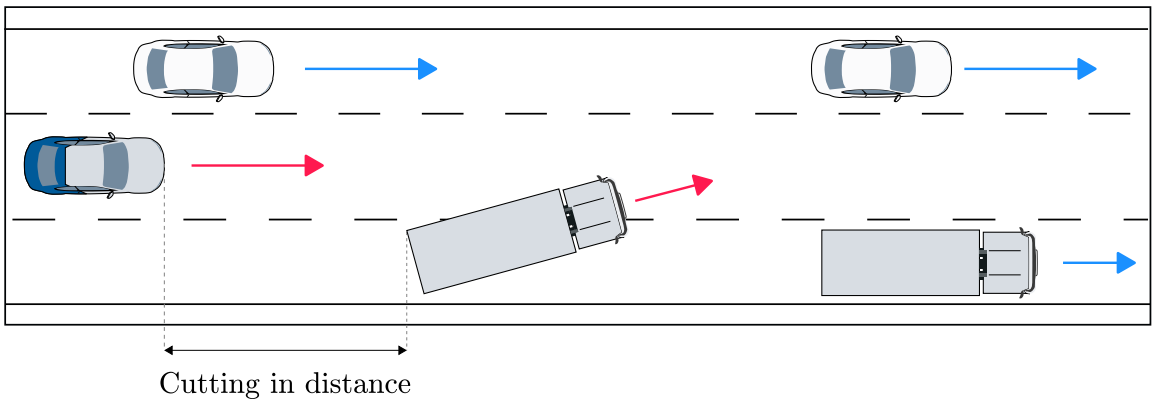

Figure 4.8: *Selected scenario for validation*

4.4.2 Validation parameters and KPIs

The parameters of the scenario can be split in two groups. First, the parameters set by the driver:

- ACC target/crousing velocity – If no preceding vehicle is detected, the ACC will accelerate to the target velocity.

- Maximum headway time to next vehicle – If a preceding vehicle is detected the ACC is going to keep the headway time below the maximum value, i.e., keeping a steady but safe distance from the vehicle in front.

Second, the environment parameters shaping the test case to a defined test run:

- Distance to the preceding vehicle – Distance between the ego and preceding vehicle at the cutting-in time instant;
- Road friction coefficient;
- Amplitude, frequency and mean velocity of preceding vehicle – In order to validate the ACC under a wide range of behaviors, the preceding vehicle's velocity was changed following a sinusoidal trajectory where the amplitude, frequency and mean are defined as input parameters for the validation.

Table 4.1 gives an overview of the validation parameters and their corresponding ranges for the selected example. The scenario was simplified for a better understanding in the following way: The cutting-in distance was fixed to a reasonable borderline value (for shorter distances the truck driver would be responsible for an accident). We did not consider variations in the road friction, and the road gradients turned out to have no influence as long as the road friction is sufficient.

Table 4.1: *Scenario parameters for model-based validation*

		min	max	Initial value
Selected by the Driver	Start velocity of the maneuver	fixed values		$150\frac{km}{h}$
	ACC target velocity			$150\frac{km}{h}$
	Maximum time gap			2s
Maneuver parameters	Distance to the incoming vehicle	fixed values		80m
	Road friction			1
	Road gradient			0%
	Average traffic velocity	$40\frac{km}{h}$	$140\frac{km}{h}$	$80\frac{km}{h}$
	Sine velocity amplitude	-$20\frac{km}{h}$	$20\frac{km}{h}$	$0\frac{km}{h}$
	Sine velocity time period	10s	40s	10s

Figure 4.9 shows one example taken from the test runs. The ACC set velocity was $150\frac{km}{h}$ and at time instance of 40s the ego vehicle's sensor detects a vehicle cutting in with a velocity of $65\frac{km}{h}$. Finally, the velocity of the preceding vehicle is changing following a sinusoidal trajectory with an amplitude of $20\frac{km}{h}$ and a period of 25 seconds.

For the validation, the following KPIs were considered:

- Minimum clearance distance between ego and preceding vehicle
- Minimum headway time
- Maximum braking energy

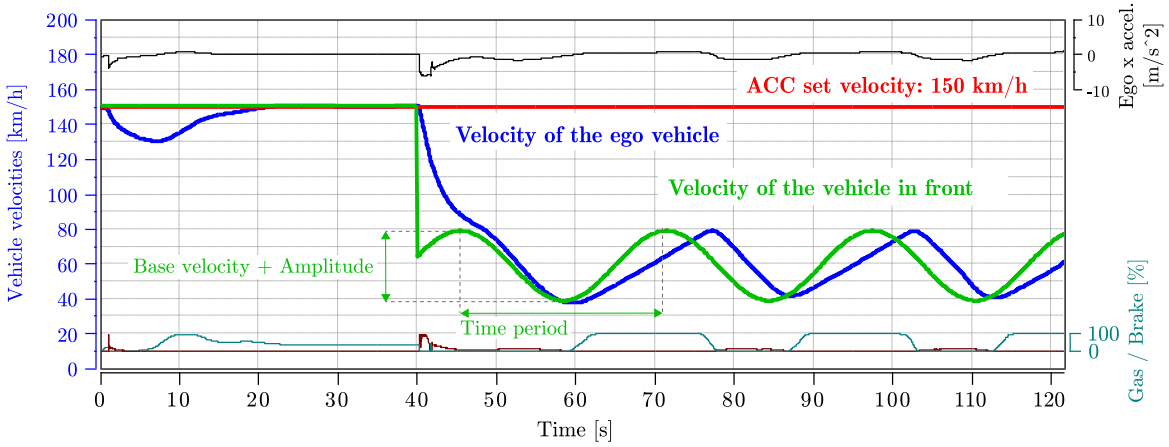

Figure 4.9: *Example validation test run*

The KPI values for the example test run are shown in Figure 4.10.

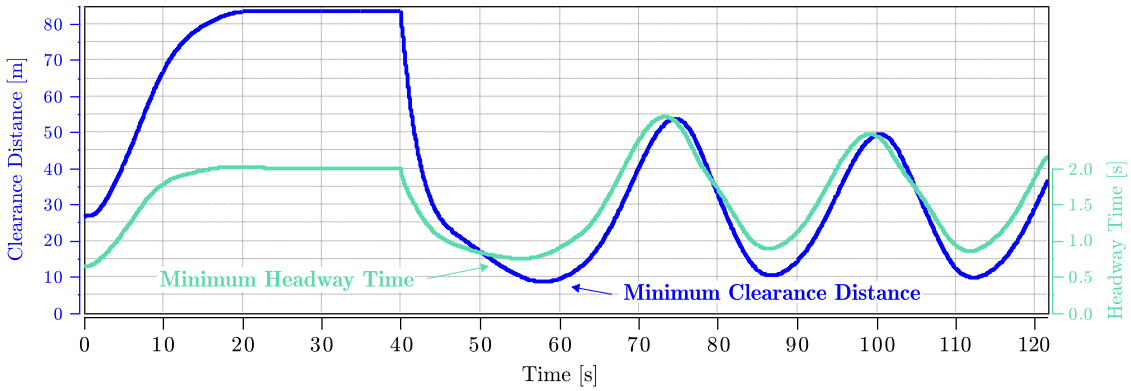

Figure 4.10: *Clearance distance and time gap KPIs for the example run*

4.5 Results

To validate the ACC for this specific scenario we first need to define a criterion that denotes a critical situation. The "Safe distance between vehicles" published by the Conference of European Directors of Roads [62] gives an overview of laws regarding the minimum clearance distance and headway time. In the publication, we find that the minimum headway time defined by the Austrian law is 0.4 seconds, regardless of the scenario.

Following the regulations as guidelines, we define that a dangerous test run ranges between 0.4s – 0.8s headway time and clearance distance ranging from 3m to 15m. Scenarios with a clearance distance lower that 3m are considered crashes to take in consideration the behavior model's standard deviation which can vary between ±1m. Furthermore, this borderline also takes into account the small simulation errors that are introduced by the imperfect vehicle and sensor models.

In this regard, we performed a multi objective optimization on the KPI models in the dangerous ranges defined above, in order to find the most critical environmental parameters for the specified scenario.

Figure 4.11 shows the results of the optimization. The optimization variables are the minimum clearance distance and the maximum breaking energy. The optimization leads to a test run with the strongest braking effort, while the minimum clearance distance is still considered safe. From Figure 4.11 a) we can see that the selected pareto-optimum solution dominates all other dangerous test runs and that a more critical solution does not exist except those with a lower predicted minimum clearance distance of 3 m. Figure 4.11 b) shows the two behavior models for minimum headway time and minimum clearance distance on the pareto-optimum solution from Figure 4.11 a). We can see that among the three parameters the average traffic velocity has the most impact on both models.

The optimization results obtained in Figure 4.11 are taken from the behavioral models; however to obtain the real values of the minimum headway and clearance distance for this test run we have to run the simulation with the corresponding parameters. After running the simulation, we determine that the minimum clearance distance and minimum headway time for the worst, bus still safe, test run are 2.2 m and 0.45s respectively.

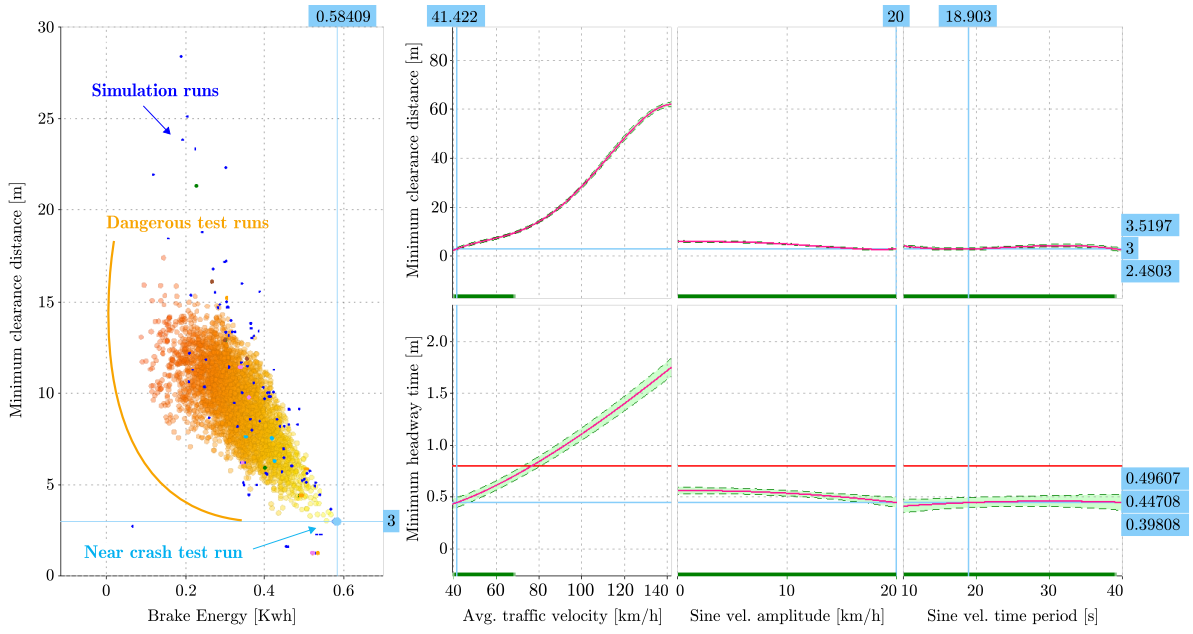

Figure 4.11: *a) Pareto front for the critical scenario; b) Parameter influence on safety*

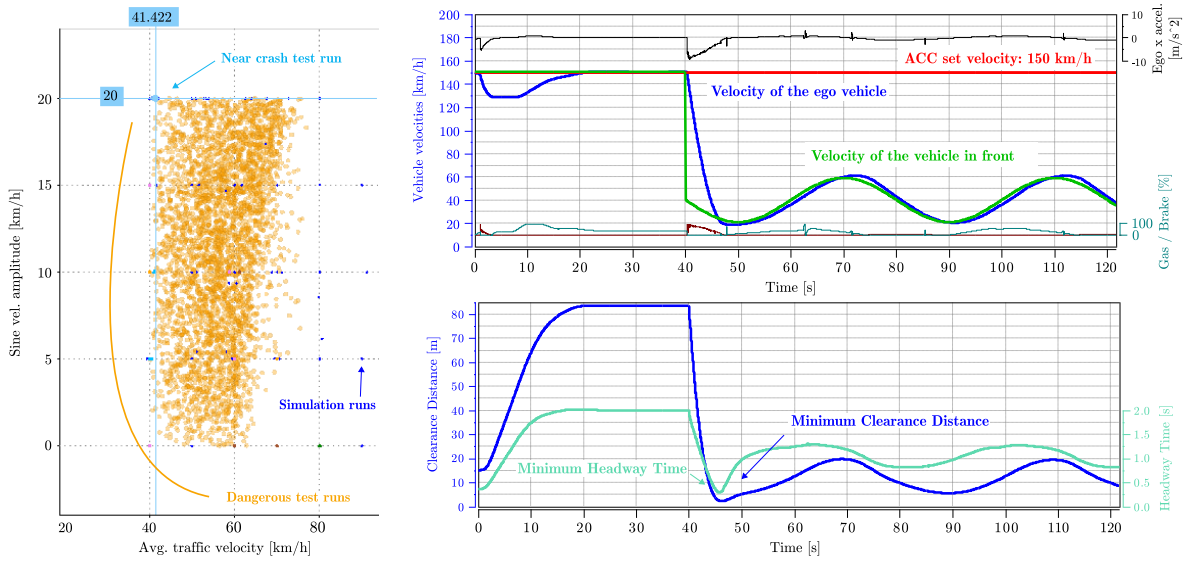

Figure 4.12: *a) Dangerous test run regarding the Sine parameters; b) Most critical test run*

In addition, Figure 4.12 a) shows all the dangerous test runs with respect to the sine velocity amplitude and the average traffic velocity. Figure 4.12 b) shows the most critical "near crash test run". We can see that in these test runs there was a combination of the sine parameters such that the velocity of the preceding vehicle was very low. This in return resulted in very harsh braking from the controller. However, the vehicle was able to decrease the velocity in time to avoid a crash.

Finally, we can state that for the test run considering any combination of the parameters in 4.1 with an ACC target velocity of $150\frac{km}{h}$ under:

- dry conditions, on flat road, with a cutting in distance of 80m,
- a sinus time period between 10 and 40 seconds,
- with a velocity amplitude within $\pm 20\frac{km}{h}$,
- a base velocity of the "cutting in vehicle" of more than $41\frac{km}{h}$,

the system will react safely in terms of "crash avoidance" (minimum clearance distance less than 3 ± 1m is not reached). Furthermore, as long as the base velocity of the cutting in vehicle is above $78\frac{km}{h}$(no dangerous test runs exist with higher velocity values) the legal limit of at least 0.8 seconds headway time is kept under all conditions.

One difference to the conventional testing is that we can draw these conclusions based on 210 simulation runs while considering three parameters. However, conventional testing methods, carried out by an engineer, would still need around 700 simulation runs, where the three parameters could be changed in steps of 11 average traffic velocity values, 9 velocity amplitude values and 7 velocity time period values.

The information of these borderline test runs are useful in later testing and validation steps which can be performed on various development environments and proving ground, where a high integration between the function and vehicle is possible.

4.6 Chapter Conclusion

In this chapter, we presented a comprehensive introduction for the development, tuning and validation of a simple ACC of a Highway pilot. The emphasis was put on the validation methodologies using as an example one test scenario. We have shown how principles of the well understood tuning task could be carried onto the validation. An ACC function was developed from the ground up, presenting the first part of a Highway pilot. The tuning of the controller was done on a fixed scenario and the ACC was then validated on a defined test case. The resulting behavior models allow us to specify:

- Areas of robust and satisfactory response of the controller
- Areas where legal limits are violated
- Borderline performance - Near crash test run
- Areas were crashes are occurring

5

Classification of Ego Related Scenarios

This chapter focuses on methods that can be used to extract desired scenario out of recorded data. The selection of specific classes can significantly reduce development and testing effort for ADF. We apply DL methods on data coming from the ego vehicle's sensors to classify scenarios relevant for a LKA system.

Most of the content of this chapter is adopted from

H. Beglerovic, T. Schloemicher, S. Metzner, M. Horn, Deep learning applied to scenario classification for lane-keep-assist systems, Applied Sciences 8 (12) (2018) 2590. doi:10.3390/app8122590.

5.1 Introduction

In order to ease and accelerate the development of an ADF, we need to be able to continuously assess and monitor its behavior throughout the whole development cycle. The data used for the development should, ideally, be classified by scenario type and severity level of safety and/or comfort issues. In recent years, scenario classification was based on heuristic algorithms [63], [64], where engineering experts spent a lot of time and effort to create the necessary rules. For each detected driving scenario, various KPI are calculated to reflect the driving quality.

However, some signal patterns are very challenging to be detected by heuristic algorithms only. There are no distinct triggers where a threshold could be set. Instead, the signals usually show decreasing and increasing slopes containing noise that could lead to misclassification. Furthermore, due to sensor noise or false target detections some information gets lost and therefore some interesting driving scenario cannot be detected correctly by the heuristics. This is where ML is applied to increase the detection rate when the NNs, along with a well labeled data-set, have learned to deal with false positives and false negatives. The NNs are not immune to data inaccuracy, but they are able to learn, to some degree, when the sensor measurements are inconsistent.

In this chapter we explore end-to-end DL methods to classify driving scenarios related to a LKA using time series data coming from various sensors. We focused on Convolutional Neural Networks (CNN) as they have shown better performance compared

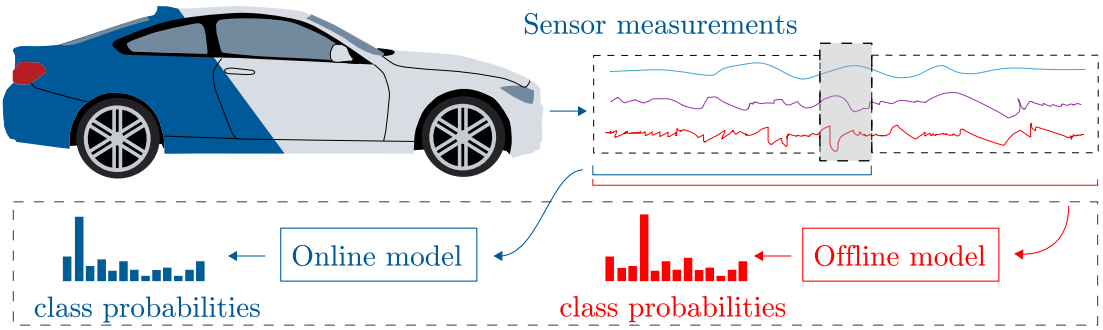

Figure 5.1: *Concept overview. Scenario detection from sensor data using an online and offline model.*

to Recurrent Neural Network (RNN). Finally, we propose two classification models as shown in Figure 5.1. The online model is used during driving, when a real time classification is necessary. This model uses the past values to make the class predictions. The offline model is used in post-processing. As this model is using the recorded data, it can look what the vehicle was doing and then trace back and make the prediction. In other words, the model makes the prediction in the center of the considered time window taking in consideration what the vehicle did before and after. This approach leads to improved accuracy and can, to some degree, eliminate wrong detections.

5.2 Related Work

In recent years, natural language processing and machine translation [65], [66], [67], had a great impact on the development of deep learning models for processing sequential data. The main idea behind the approach is to translate words, or phonemes if speech is used, into a fixed length vector embeddings. These vector embeddings can then be used as a time series sequence. The research resulting from the natural language processing is widely adopted to other fields that deal with time series data. Until recently, RNNs have been the *de facto* choice for time series modeling. RNNs are networks that use hidden states that get propagated during time to compress valuable information from the past. Especially successful RNNs are the Long Short-Term Memory (LSTM) [68] or Gated Recurrent Unit (GRU) [69]. Both use several gates to control which information should be stored and which information should be forgotten.

Outside the natural language processing, the LSTM and GRU have been applied in various tasks such as general time series classification [70], classification of Electrocardiography (ECG) data, [71], time series forecasting [72], etc.

Other researchers have also combined convolutional neural networks CNNs with RNNs to increase model performance. The main idea is to use the CNN in the first layers as feature extractors and later use RNNs for sequence classification. This method has been applied to trajectory clustering [73], human activity classification [74], sentiment classification [75], and others.

However, RNNs have several limitations that can impact model accuracy and training

duration. The training data for RNNs need to be sequential and divided in such a way that the hidden outputs from one batch match the inputs of the next batch. This batching procedure could lead to deficient performance as some classes are rarely seen by the network. Normally, shuffling of the data could greatly improve the training. However, shuffling of the data is not possible because we need to follow the constraint that each previous batch needs to lead to the current batch to preserve the propagation of the hidden states. In addition, RNNs process only one set of features at a time. After each step new hidden states and outputs are calculated which are then used in the next step. This sequential calculation procedure leads to increased training duration. CNNs do not suffer from these limitations. The data can be shuffled as individual batches do not have to be sequential. Furthermore, as the CNN is a feed-forward network the whole batch is processed directly.

The advantages of CNNs over RNNs have led researchers to apply only CNNs for time series classification [76]; however, their performance has been inferior until recently. Bai *et al.*[77] have shown that convolutional neural networks can produce even better results than RNNs if new architectural improvements like dilatation and residual networks are used. Because of the recent breakthrough and their superior performance, we will focus on the usage of CNNs for the task of time series classification and compare their performance to RNNs.

Outside the scope of time series classification, CNNs have also been used for classification of driving scenarios. Gruner *et al.*[78] proposed a spatiotemporal representation (Grid Map) that is able to capture dynamic traffic behavior. The representation is constructed using object information obtained by fusing data from several sensors. The representations are then used as inputs for a CNN which classifies the data into five classes: *Validity of frame*, *Ego vehicle speed larger than 1m/s*, *Leading vehicle ahead in lane*, *Other vehicle overtaking the ego vehicle on adjacent lane*, *Cross-traffic in front of the ego vehicle*.

Our research differs from the mentioned methods as we perform time series classification using only sensor data. This data is used as input for a CNN augmented with the best practices obtained from various recent studies, i.e., separable and dilated convolutions, residual connections and self-normalizing layers. In addition, we apply the network on a different classification problem, classifying challenging driving scenarios relevant for a LKA.

5.3 Time series classification using CNN

This section will give an overview of the proposed model that consists of a CNN acting as a feature extractor which is followed by a dense multi-layer perceptron for classification. The convolutional feature extractor uses several enhancements compared to an ordinary CNN, which will be explained in the following subsections.

5.3.1 Depthwise Separable Convolutions

Given an input tensor T_i with shape $ch_i \times h_i \times w_i$, where ch_i is the input depth or number of channels, h_i and w_i are the height and width respectively, an ordinary convolution layer applies the kernel $k \times k$ over all channels of the input tensor simultaneously and the whole kernel is applied for ch_o times, leading to the output tensor T_o with dimensions $ch_o \times h_o \times w_o$. Thus, the dimension of the kernel K is defined as $k \times k \times ch_i \times ch_o$, leading to a computational cost of $h_i \cdot w_i \cdot ch_i \cdot ch_o \cdot k \cdot k$. Generally, the convolutional layer filters input features and computes new representations at the same time.

On the contrary, the depthwise separable convolutions [79], [80] separate the feature filtering and feature computation into two parts. For the filtering part, a separate convolutional kernel K_i is applied for each of the input channels ch_i. As all the channels have their individual kernels, an additional layer is needed to combine these filtered features. Thus, the depthwise convolution is followed by a 1×1 convolution generating the new representation. The 1×1 layer is a linear combination of the filtered features generated in the depthwise convolution. Each kernel K_i has a dimension of $k \times k \times 1$, and combined with the 1×1 convolution leads to a computational complexity of $h_i \cdot w_i \cdot ch_i \cdot (ch_o + k \cdot k)$. The number of parameters compared to the ordinary convolutional layer is reduced by a factor of:

$$\frac{ch_o \cdot k \cdot k}{k \cdot k + ch_o} \tag{5.1}$$

5.3.2 Dilated Convolutions

One drawback of ordinary CNNs is that in order to achieve a wide receptive field the network needs to be very deep. For example, if we have a 1D time series input and network with two convolutional layers, each with a 1×3 kernel and stride $(1, 1)$ (the kernel moves by 1 feature each step), the receptive field of the second layer will be 5. In other words, a perceptron in the second layer will be able to "perceive" 5 values from the input. In general, the receptive field grows linearly with the depth of the network and is given as:

$$R_i = R_{i-1} + \prod_{j=1}^{i} s_j \cdot (k - 1) \tag{5.2}$$

where R_{i-1} and R_i are the receptive fields of the previous and current layer, s_j is the product over all previous strides and k is the kernel size.

However, when working with time series data, it is often necessary to consider a very wide view of the input to make adequate predictions. What if we could increase the receptive field of the network without adding more layers and increasing the size of the network? The solution is to use dilated convolutions [81], as shown in Figure 5.2. The main idea is that during the convolution the kernel is not considering every adjacent input from the previous layer. Instead, there is some fixed dilation step d_i

between them. With this approach, it is possible to achieve an exponentially increasing receptive field formulated as:

$$R_i = R_{i-1} + d_i \cdot \prod_{j=1}^{i} s_j \cdot (k-1) \tag{5.3}$$

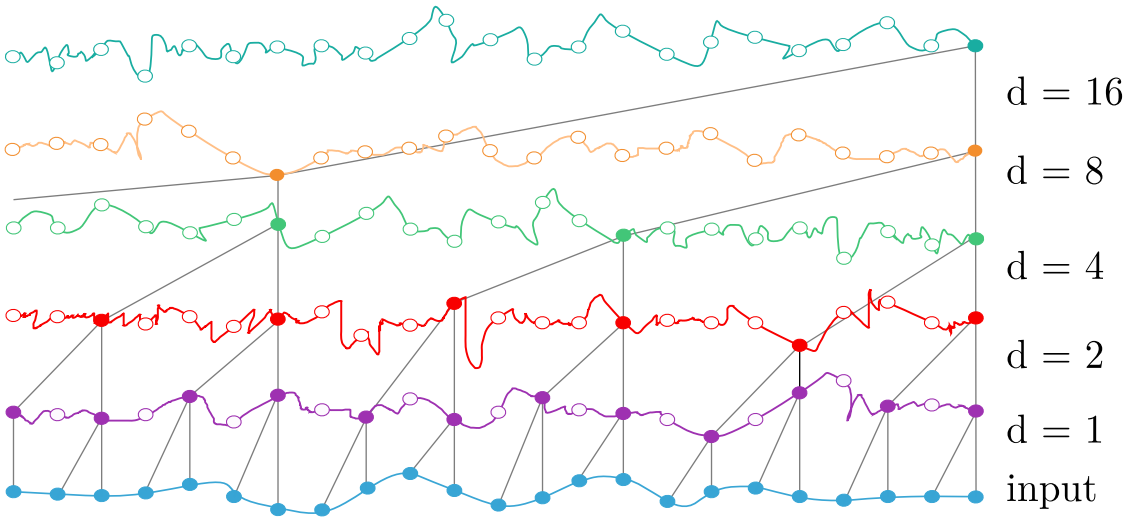

Figure 5.2: *Visual representation of the dilated convolutional layers with different dilation factors d. It can be seen that the last layer has a wide receptive field.*

It can be seen that the receptive field of the ordinary CNN (5.2) is actually a special case of the dilated convolutional network (5.3) where the dilation factor d_i equals 1. In practice, it is common to increase the dilation factor d_i exponentially for each layer i, i.e., $d_i = 2^i$.

5.3.3 Residual Connections

In order to produce models that generalize well on highly complex data, the number of layers needs to increase. By creating deeper models, and by using good regularization, we assure that the models learn the necessary features to produce appropriate outputs. However, training deep networks presents a challenge because of the vanishing gradient problem. The gradients from the output layers need to be propagated through the whole network, which makes the gradients closer to the input part of the network infinitely small, saturating and degrading the model.

He *et al.*[82] have proposed skip connections, as shown in the residual block in Figure 5.3. The skip connection allows the network to more easily learn a residual mapping instead of the original underlying mapping, circumventing the gradient vanishing problem, as gradients can more easily flow through the network. However, Veit *et al.*[83] have done a further study on the residual connections and argue that the skip connections allow the network to create better prediction as they effectively allow information to pass through different pathways.

The skip connection introduced in the residual block is basically an identity connection added to the convolutional layers and can be expressed with the following equation:

$$res_x = x + F(x), \tag{5.4}$$

where x is the layer's input and $F(x)$ presents an ensemble of operations, in our case convolutional layers followed by an activation function.

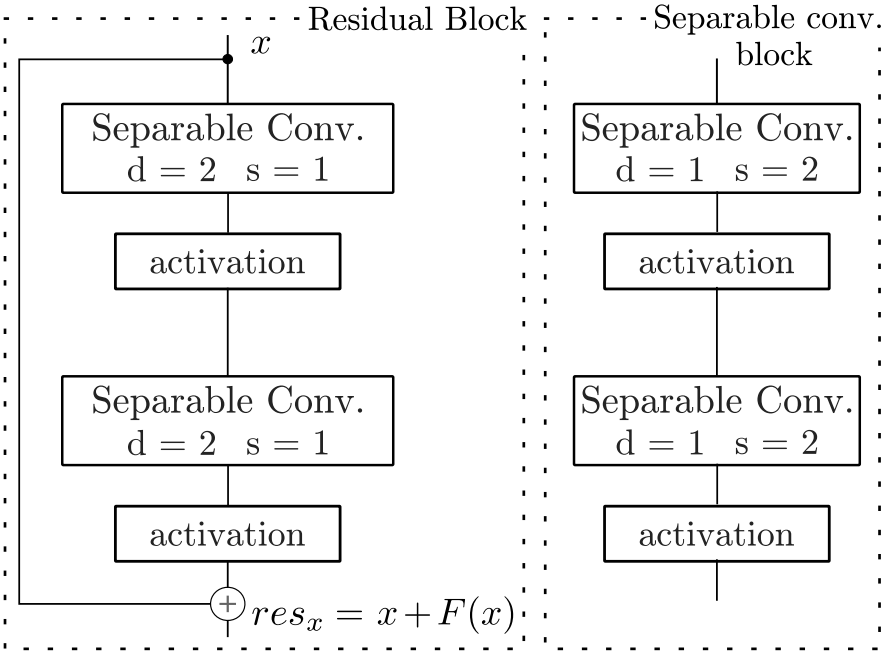

Figure 5.3: *Building blocks of the CNN network. The residual block with dilated separable convolutions is shown on the left. The separable convolutional layers with stride $s = 2$ for dimensionality reduction are shown on the right.*

5.3.4 Self-normalizing Layers

It has become a widespread practice to use Batch Normalization [84] between layers in CNNs to assure that layer outputs have zero mean and unit variance. This approach leads to faster training and serves as a regularizer for the network. However, a drawback of batch normalization is that it increases the number of parameters of the network as they are used after each layer.

Klambauer *et al.*[85] have introduced a new activation function called Scaled Exponential Linear Unit (SELU) that makes the layer's outputs converge to zero mean and unit variance naturally. This means that the usual batch normalization layer followed by nonlinear activation can be directly replaced by the SELU activation. The SELU activation is defined as:

$$selu(x) = \lambda \begin{cases} x, & \text{if } x > 0 \\ \alpha e^x - \alpha, & \text{if } x \leqslant 0 \end{cases}, \tag{5.5}$$

where λ and α are coefficients that determine the mean and variance of the layer's output. Usually, their values are selected as $\lambda = 1.6733$ and $\alpha = 1.0507$, which leads to zero mean and unit variance outputs.

If dropout [86] is applied in the network than a special type of dropout called alpha dropout is needed to preserve the zero mean and unit variance between the layers. Together the SELU activation and alpha dropout lead to self-normalizing layers in the model.

Table 5.1: *Model architecture for scenario classification*

layer	kernel	stride	dilation	activation	input on. model	input off. model
conv1	1×3	1×1	1×1	selu	10×60	10×80
conv2	1×3	1×1	1×2	linear	10×60	10×80
conv3	1×3	1×2	1×1	selu	18×30	18×40
conv4	1×3	1×2	1×1	linear	18×15	18×20
conv5	1×3	1×1	1×2	selu	18×15	18×20
conv6	1×3	1×1	1×4	linear	18×15	18×20
conv7	1×3	1×2	1×1	selu	20×8	20×10
conv8	1×3	1×2	1×1	selu	20×4	20×5

	output size conv8	dense1	dense2	dense3	dense4
online	80	64	32	20	13
offline	100	64	32	20	13

5.3.5 Model Architecture

By applying the modifications to the ordinary convolutional network, we construct two models. One model for online and one for offline classification of scenarios. Both models share the same architecture but differ on how the input data is processed. The model architecture consists of a combination of two residual and two separable convolutional blocks, which are shown in Figure 5.3. The eight convolutional layers are then followed by four dense layers. The parameters for each layer can be seen in Table 5.1.

The input to the models is split into windows of equal length and move, with a fixed stride, through the time series data, as shown in Figure 5.4. The online model is not able to "look" into the future and its input window consists of past measurements followed by the current stride. The model then uses this data to predict the classes associated with the current stride. When applied to real measurements on a vehicle, this scheme would result in a prediction for each stride duration, e.g., every half a second. The offline model has all the data available and thus can consider behavior

of the vehicle before and after the current stride which leads to higher prediction accuracy.

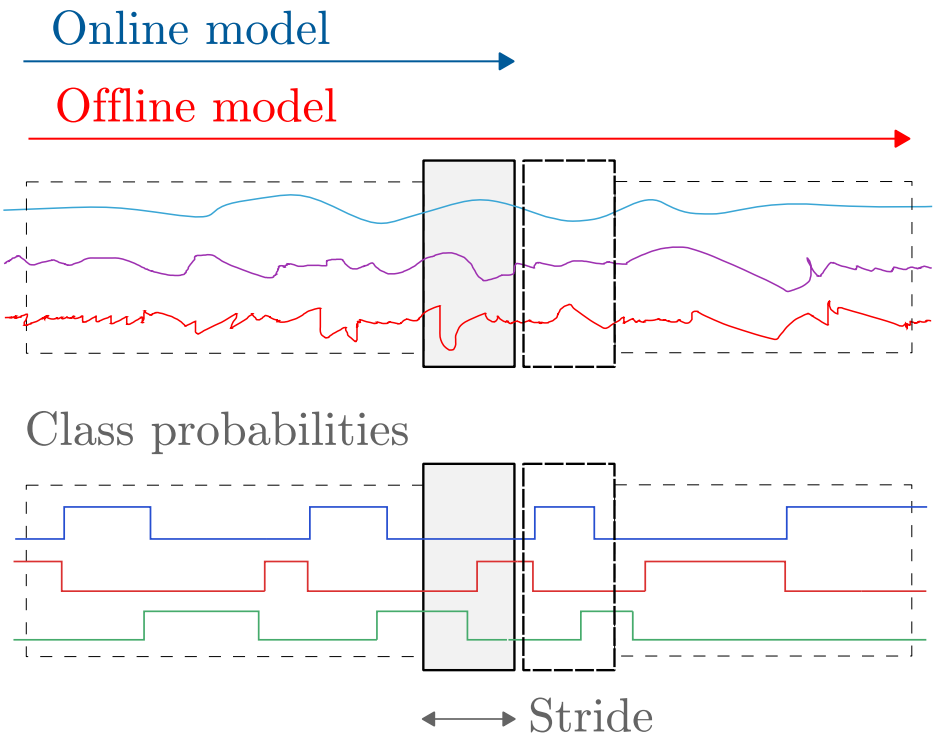

Figure 5.4: *Window partitioning for class predictions. In the online model, the inputs are only the current stride and an aggregated past. No information of what happened after the prediction is available. In the offline model, the information what the vehicle did after the detection is available, enabling the model to make a prediction based on past and future behavior of the vehicle. To process the whole data, the windows are moved with a fixed stride length, generating corresponding class probabilities.*

5.4 Driving Data

The full dataset used to train the neural networks consists of real-world recordings obtained in seven highway and four motorway runs in the area around Graz, Austria. The dataset contains 183 channels and 304 thousand samples per channel, with a sampling rate of ten samples per second. The channels are raw sensor measurements coming from various sensors like the lane detection camera, inertial measurement unit, encoders, etc., recorded directly from the CAN-Bus interface. As the focus is on classifying the scenarios relevant for a LKA, 10 relevant input channels were chosen in order to learn the distinct features.

The used channels are:

- *Lane Assist Lateral Deviation*
- *Lane Assist Lateral Velocity*

- *Lane Assist Lateral Velocity SMO20* - Smoothed with a 20 step window
- *Lane Curvature*
- *Lane Width*
- *Steering Wheel Angle*
- *Longitudinal Acceleration*
- *Lateral Acceleration*
- *Velocity*
- *Yaw Rate*

The model should learn to segment the data into 13 classes, as shown in Figure 5.5, which were hand labeled using a custom labeling tool.

The labels are:

- *TR: Turn Right* - Low velocity sharp turn
- *TL: Turn Left* - Low velocity sharp turn
- *LCLR: Lane Change Left to Right*
- *LCRL: Lane Change Right to Left*
- *S: Straight* - Straight driving
- *CL: Curve Left* - Curve with fixed radius
- *CR: Curve Right* - Curve with fixed radius
- *TOL: Turn Out Left* - Curve with increasing radius
- *TOR: Turn Out Right* - Curve with increasing radius
- *TIL: Turn In Left* - Curve with decreasing radius
- *TIR: Turn In Right* - Curve with decreasing radius
- *R: Roundabout*
- *SS: Still Stand.*

The labeled data contain class probabilities for each sample from the training data. When a sample could not be assigned to one of the thirteen classes, a uniform distribution between the classes was introduced to represent the unknown state. In other words, the model should not be able to tell which class the sample belongs to by generating low probabilities for all classes.

The distribution of samples per class is shown in Figure 5.6. A strong imbalance between the classes exists, which will limit the capacity to train the models. Classes TR, TL and SS appear very rarely in the dataset, whereas S, CL, CR appear with a very high frequency. There was no possibility to balance the classes more evenly during the recording. In addition, various time series augmentation techniques were applied but they did not show any improvements on the accuracy of the models.

Furthermore, time series data like this contain classes which repeat for a long time, i.e., for a certain amount of time during training, the network gets as an input only one class which hinders generalization. A remedy for this problem is to shuffle the data. However, shuffling of the data is not possible if RNNs are used. The requirement of these networks is that all data is introduced sequentially. CNNs on the other hand

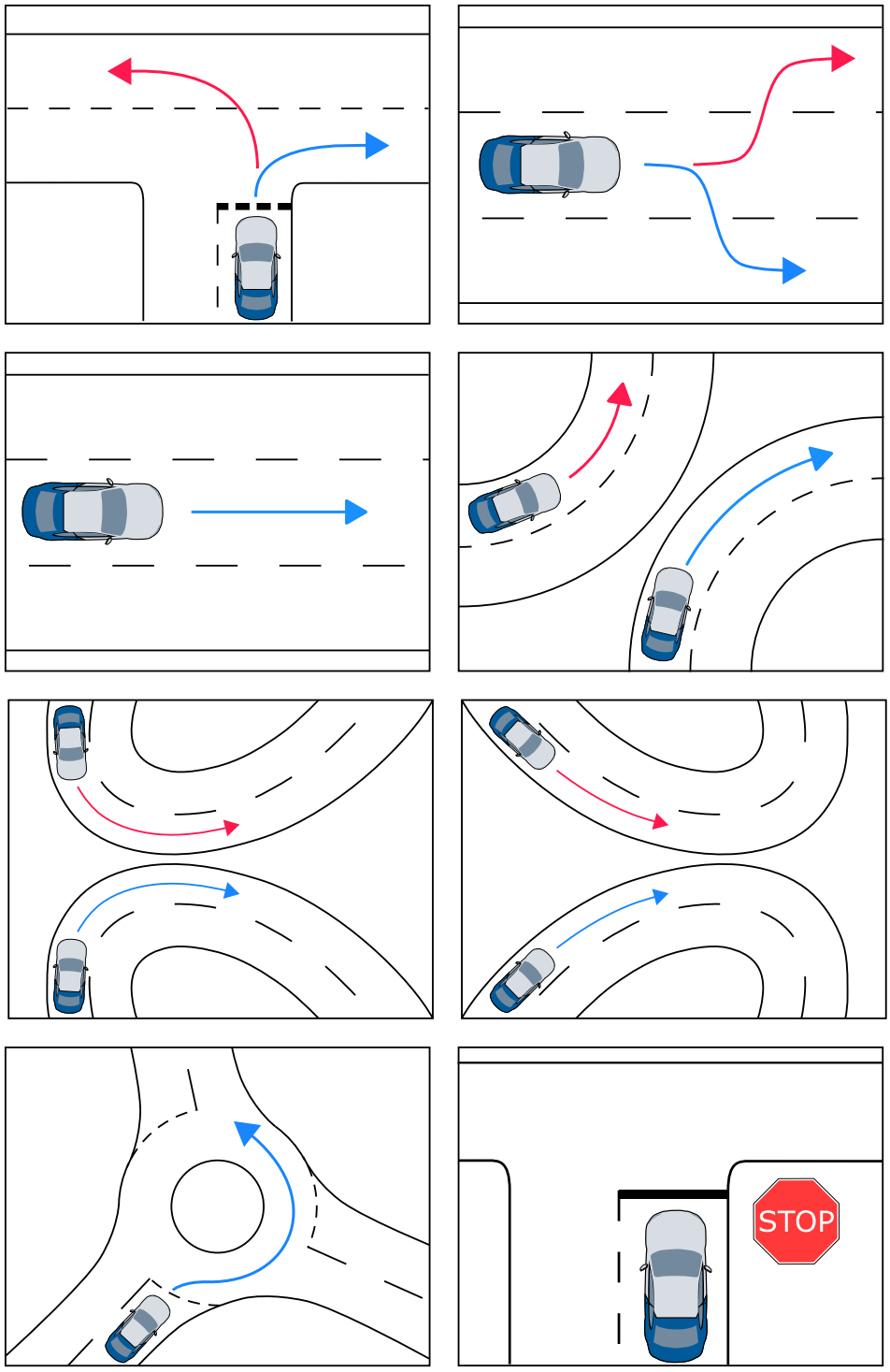

Figure 5.5: *Example scenario classes. From left to right, top to bottom the labels are: TR: Turn Right, TL: Turn Left, LCLR: Lane Change Left to Right, LCRL: Lane Change Right to Left, S: Straight, CL: Curve Left, CR: Curve Right, TOL: Turn Out Left, TOR: Turn Out Right, TIL: Turn In Left, TIR: Turn In Right, R: Roundabout and SS: Still Stand*

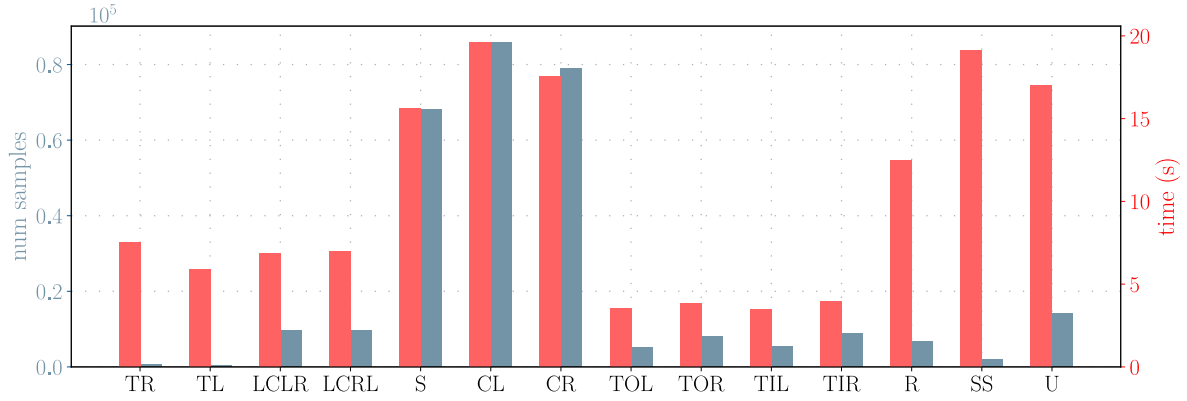

Figure 5.6: *Distribution and average duration of scenario classes.*

are applied on segments of the data, i.e., windows. Inside the windows the data is sequential but individual windows can be shuffled which increases the accuracy.

The average duration per class is shown in Figure 5.6. This information is used in order to determine the size of the windows used for the CNNs. By increasing the window size, we increase the complexity, size and accuracy of the models. However, if we increase the window size top much the accuracy will decrease because we are not able to distinguish classes with shorter duration. On the other hand, if we use a small window size the network will not be able to learn the classification triggers for the classes with longer duration.

After an experimental analysis, we decided to use a window size of 8 seconds (80 samples) for the offline, and 6 seconds (60 samples) for the online model, providing good balance between accuracy and complexity. In addition, both models use the same stride length of 5 seconds (5 samples) to traverse the windows over the whole data.

5.5 Results

The models were trained and evaluated using both the highway and motorway data. The first dataset contains only the highway records. Whereas, the second dataset contains all recordings. In both cases, a test set including 10% of the recorded data is left for the evaluation of the models after the training.

We compared the performance of the proposed CNN networks with two LSTM networks using the same window size and stride length. The first LSTM network is a multi-layer network with 3 layers and a hidden state size of 512 neurons. The second LSTM network is a 3 layer bi-directional network with the same hidden state size. The bi-directional LSTM has twice as many neurons compared to the first LSTM network. One part of the biLSTM network starts at the beginning and traverses the input until the end, and the second part starts at the end and traverses the input to the beginning. This approach leads to better results as the network combines both the forward and backward pass to make a prediction.

All models were trained for 100 epoch using a decaying learning rate, with a batch size of 256 samples. The CNNs use the *Adam* optimizer, whereas the LSTM networks use the *Adagrad* optimizer. To increase generalization, a dropout rate of 15% was used. For the LSTM networks the dropout was applied on the 3 layers, and for the CNN networks the dropout was applied on the dense layers. In addition, the CNN networks use L2 regularization for the kernel weights with a regularization factor of $2e^{-3}$.

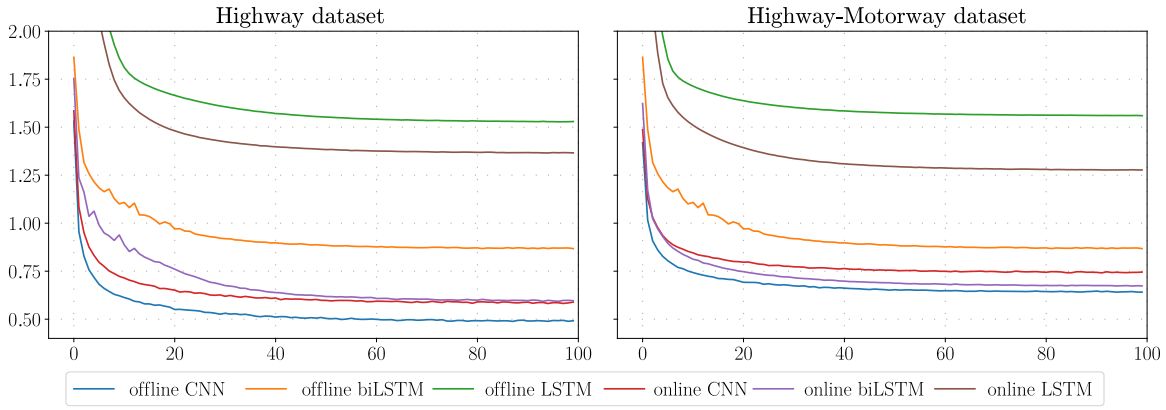

Figure 5.7: *Training losses for the online and offline models trained on the highway and highway-motorway data sets respectively*

The training losses are shown in Figure 5.7. The offline CNN model has the lowest training loss compared to all other networks. The LSTM network has the worst performance and the network is not able to model the data well. The bi-directional LSTM has a much better performance, and in the highway-motorway dataset the online model performs better than the online CNN model.

However, when comparing the offline and online bi-directional LSTM we can see that the offline model performs worse than the online one. A possible explanation could be that the offline model is not able to memorize the "future" part of the data well when making a prediction in the middle of the window.

The accuracy of the models on the test sets can be seen in Table 5.2. The columns present the evaluated models and the rows present the accuracy and loss for the highway and highway-motorway datasets. We can see that the offline CNN has the best accuracy on both test sets. The online bi-directional LSTM has the lowest loss in both cases, but still a lower accuracy compared to the CNN. This is possible because the loss is calculated as the mean cross entropy loss, meaning that the bi-direction LSTM makes more mistakes in general, but the class probabilities are closer to the desired ones.

The decrease in accuracy between the highway and highway-motorway data set is due to the fact that the motorway dataset is more complex, and that the outputs from the sensors were not always consistent. This presents a challenge as we are directly using outputs from the LKA function even though sometimes those outputs are not valid. In addition, because some classes are appearing less frequently than others, the models are not able to fully generalize.

Table 5.2: *Test accuracy and loss on the Highway and Highway-Motorway dataset*

	High. acc.	High. loss	High. & Mot. acc.	High. & Mot. loss
off. CNN	86.4%	0.725	81.6%	0.821
off. biLSTM	76.8%	0.775	73.3%	0.833
off. LSTM	51.5%	1.614	45.3%	1.486
on. CNN	84.4%	0.787	79.6%	0.824
on. biLSTM	77.6%	0.663	78.9%	0.640
on. LSTM	57.4%	1.469	69.0%	1.192

Figure 5.8. shows a part of the highway test set. We confirm that the model is able to distinguish the desired classes providing satisfactory results. However, some misclassifications are still present. The black ellipses in the figure highlight some of them. For example, the model confuses the CL class with TIL, TOR or LCRL. These classes are all very similar with subtle differences in the curvature radius or duration of the maneuver. In general, the accuracy and generalization of the model could be greatly increased if more balanced data were available.

5.6 Chapter Conclusion

In this chapter, we applied end-to-end DL architectures for classification of driving scenarios used for evaluation of ADFs. We focused on scenarios relevant for LKA systems and considered both convolutional and recurrent neural networks. Through evaluation on two datasets, we concluded that CNNs provide better accuracy. Even though the performance of the proposed method is satisfactory, due to the imbalances of classes in the data, similarities between classes and sensor inconsistencies the networks could not fully generalize. However, we believe that deep neural networks represent a great tool for scenario classification as the complex classification rules can be learned directly by the networks. In addition, the accuracy of the networks can be increased incrementally as more data is recorded.

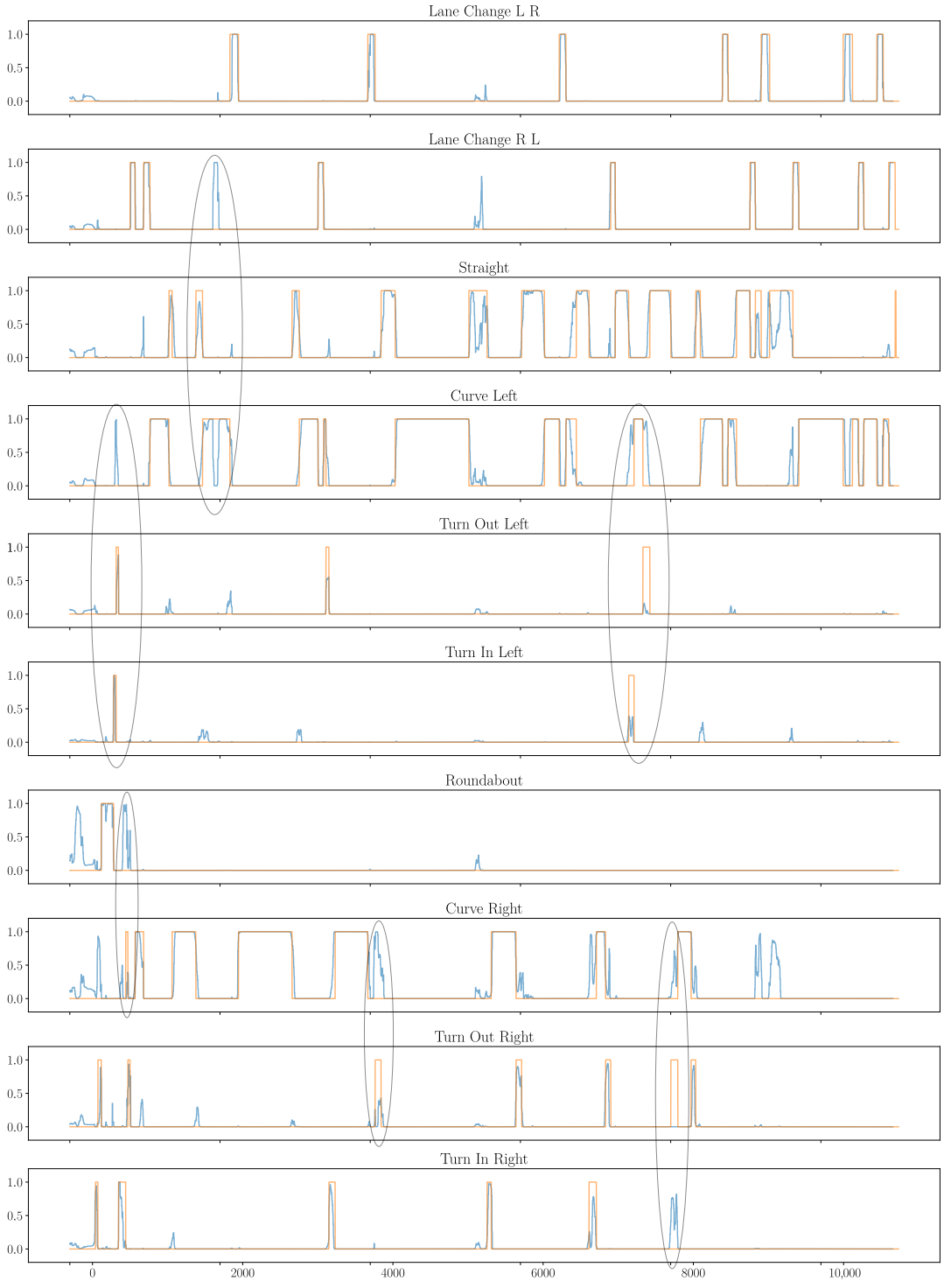

Figure 5.8: *Part of the highway test set with a duration of 20 minutes showing that the model is able to learn the desired class labels. Predictions from the offline CNN model are shown in blue and the class labels are shown in orange. Classes TR, TL, SS were not shown as they did not occur on the highway. The black ellipses highlight some misclassifications made by the network.*

6

Classification of Dynamic Traffic Scenarios

In this chapter, we introduce a new methodology to represent and classify scenarios with dynamic traffic participats. The representation is based on the field of views that the traffic participants are occupying, leading to a significant size reduction and increased accuracy.

Most of the content presented in this chapter is adopted from

H. Beglerovic, J. Ruebsam, S. Metzner, M. Horn, Polar occupancy map - a compact traffic representation for deep learning scenario classification, in: 2019 IEEE 22nd International Conference on Intelligent Transportation Systems (ITSC), 2019. © IEEE 2019

6.1 Introduction

As discussed in the previous chapter, scenario classification is mostly done manually or using rule-based algorithms. The manual extraction is done by engineers during the test drive or directly on the recorded data. Even though manual extraction of relevant scenarios provides high accuracy, it requires a lot of effort and time. On the other hand, rule-based methods use predefined triggers and patterns that are matched against the data. Complicated scenarios require great effort to define all the necessary rules.

As a lot of the recorded driving data is already available, it is possible to use NNs to learn the required features and underlying scenario rules. This approach is highly scalable as new scenarios can be added by retraining, and once the scenarios are learned, the network can be easily deployed and used on new data.

In this chapter, we try to extend our previous research and include also information of dynamic traffic participants in order to create more complex scenarios. The overall idea, shown in Figure 6.1, is to first create a representation that captures the dynamic traffic behavior from recorded data and then apply DL to extract relevant scenarios.

The task of the NN is to look at specific test runs from the recordings, and learn how to group them in the correct scenario classes. By focusing only on the relevant scenarios

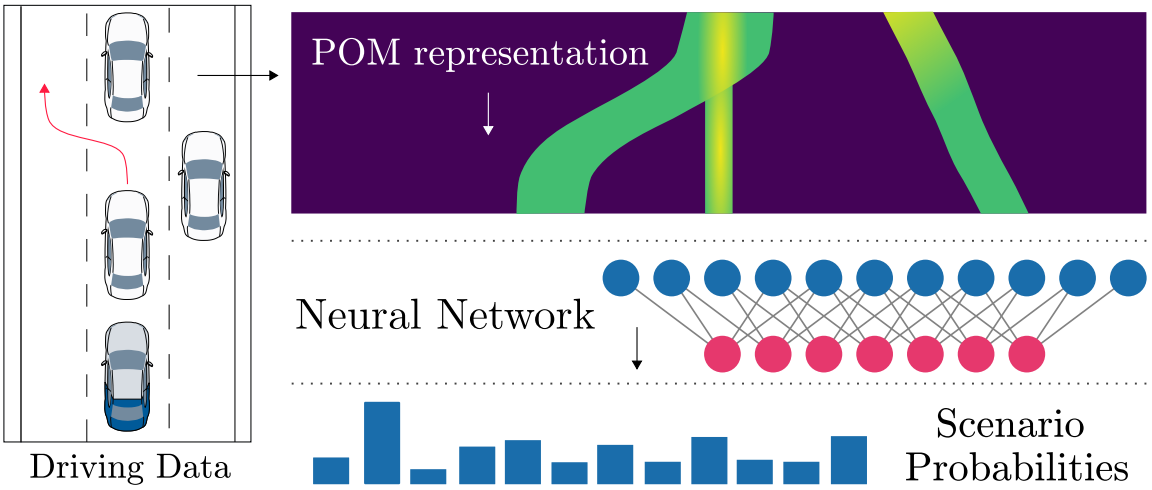

Figure 6.1: *Concept overview of the proposed scenario classification framework using the Polar Occupancy Map (POM) representation. The example scenario is a Target ahead switch cut out. © IEEE 2019*

and eliminating the known and unimportant ones, it is possible to significantly reduce current testing and validation times of ADFs. In addition, this method could be used to find and add new scenarios to a catalog of relevant scenarios.

However, there is an intrinsic challenge with NNs that needs to be addressed. The feature input size of a NN has to be constant and independent of the current scenario and number of participants. Furthermore, to create a robust classifier the input should be independent from the order of the participants to avoid generating different scenarios when the motion of the participants is the same. In this chapter, we focus on a novel compact representation that satisfies the requirements and relies on sensor fusion information, i.e., object lists. The proposed method creates a POM representation of the scenario. The benefits of such an approach are a huge reduction in the input size and an abstract representation that can be used on a variety of traffic scenarios.

As a proof of concept, we focus on detecting and classifying most common highway scenarios. The scenarios are categorized in 9 classes: Driving in lane, Lane change, Ego free lane cut in, Ego free lane cut out, Target Ahead (TA) free lane cut in, TA free lane cut out, TA switch cut in, TA switch cut out.

6.2 Related Work

Scenario classification has been addressed by several research groups in the recent years. The methodologies used for scenario classification can be divided into two major groups: model-based classification and classification using ML. Regarding the model-based approach, Elrofai *et al.* [87] have used a vehicle model to estimate the yaw rate using a minimal number of basic sensors. The estimated yaw rate is filtered and together with other parameters it is used to classify turn and lane change scenarios. The difference of the proposed POM approach is that it considers other traffic participants and does not rely solely on the internal measurements of the ego vehicle.

The ML approach can be further divided into image and time-series based classification. Image-based classification was addressed by Kastner *et al.* [88] and Bernini *et al.* [89]. The two research groups proposed similar methods in which they use static images to classify the current type of surrounding the vehicle is in. Both groups divide the image into 16 equal parts, transforming each sub-part into the frequency domain and use ML methods for the classification. Kastner *et al.* proposed a Hierarchical Principal Component Classification (HPCC) that uses a 400×300 image to classify whether the vehicle is currently located on a highway, country road or inner city. Similarly, Bernini *et al.* used a 256×256 image to classify urban, highway and rural driving. They did a comparison between Principal Component Analysis (PCA), NN and Support Vector Machines (SVM). The limit of this approach is that it cannot capture the dynamic behavior of the traffic and it mainly focuses on the surrounding to make the prediction of the scenario type.

In our previous work [14], we used time-series data recorded from the ego vehicle's sensors and NNs in order to classify scenarios relevant for a Lane Keep Assist System. However, other traffic participants were not taken into consideration.

Roesener *et al.* [90] worked also on time-series classification of scenarios and applied it on the assessment of automated driving. They used extracted features, e.g., TTC, Time to Next Cut In (TTNCI), TTC and TTNCI derivatives etc., to classify four classes: Lane Change, Vehicle Following, Free driving and Cut In. They compared the performance of the following methods: Naive Byes, ADABoost Simple Tree and Median Tree.

Cara *et al.* [91] used time-series data to classify critical car-cyclist scenarios. The input for their proposed method was the cyclist trajectory and ego vehicle velocity and acceleration. They used 99 recordings that were filtered to ensure that a cyclist was included.

Even though Roesener and Cara *et al.* consider dynamic behavior of other participants, the proposed approaches are limited to only one participant per recording. In addition, they use the distinct information of the participant to create the features needed for the classification. In our approach, we explore how the information of an arbitrary number of participants can be represented and used as input to a NN.

A representation of driving scenarios that can capture the dynamic behavior of an arbitrary number of participants was introduced by Gruner *et al.* [78]. They have used a top view Grid Map (GM) approach and proposed three types of maps: Velocity Grid (VeG), Stacked Velocity Grid (SVeG) and History Grid (HiG). The VeG is a 3 channel map where the first channel represents a binary occupancy, i.e., whether an object is present or not, and the other two channels capture the object's v_x and v_y velocity components. Both the SVeG and HiG are based on the VeG. The SVeG is a stacked representation of two VeGs separated by some Δt leading to a 6 channel representation. The HiG representation fuses several VeGs by fading the occupancy channel of older time-steps and uses the last VeG for the two remaining channels. All maps use 50×50 pixels per channel, with a resolution of 1 pixel per meter in the longitudinal and 2 pixels per meter in the lateral direction. In other words, to capture a single time-step, the VeG requires $3 \times 50 \times 50 = 7500$ inputs. The HiG requires the

same amount for an arbitrary number of time-steps; however, because of the fusion of several VeGs it is not possible to avoid information loss. For example, if a vehicle moves right (*motion-1*) and then left again (*motion-2*) the whole motion would not be visible as *motion-2* overwrites *motion-1*. The SVeG requires double the amount of inputs as VeG and introduces information loss depending on the size of Δt.

We propose a completely different method for encoding the dynamic behavior of traffic participants called Polar Occupancy Map. The POM requires only 270 inputs per time step to capture the closest traffic participants without loss of relevant information. This leads to a size reduction of factor 27 per time-step compared to the VeG representation.

Our work shares some resemblance with the Vector Field Histogram (VFH) proposed by Borenstein *et al.* [92]. They have used the field of view to create a polar histogram density function for robot navigation. Even though the proposed representation also uses the field of view approach, we do not use the histogram representation for the horizontal axis, instead we propose a new function that is used to manipulate the attention of the neural network. In addition, the representations are stacked to track the traffic motion through time. The next section will give an in-depth explanation of the proposed approach.

6.3 Polar Occupancy Map

The POM approach is based on the idea that the dynamic behavior of traffic participants can be tracked by measuring the field of view that each vehicle is occupying. This process is shown in Figure 6.2.

The design of the POM representation starts with the ego vehicle's local coordinate system (x, y) as shown in Figure 6.2 a). This scenario contains two vehicles. The first vehicle is performing a left lane change entering the lane of the ego vehicle. The second vehicle is moving forward on a free lane left to the ego vehicle. In order to track the movements of the vehicles, three time-steps t_0, t_1, t_2 are shown. The field of view of the ego vehicle starts at the x-axis, or zero radians, and changes to π in the clockwise and $-\pi$ in the counter clockwise direction. For each time-step, the slices of the field of view φ_i that each vehicle occupies are recorded. This is denoted with light red color in the figures.

The recorded fields of view for each vehicle can be visualized in a 2D plot as shown in Figure 6.2 b). In this subfigure, three plots that correspond to the three time-steps t_0, t_1, t_2 are shown. The horizontal axis is the field of view φ, given in radians, and the vertical axis is a generic function f. This function is used to manipulate the attention of the NN and is chosen such that it aids the network in finding the extraction features. Considering the classes we want to recognize in this research, the minimal distance to the vehicle and the lateral velocity of the ego vehicle are used as arguments.

The focus of the network should be on vehicles that are closest to the ego vehicle and are in the same lane. Furthermore, the NN needs to distinguish if the motion in the representation is coming from the traffic or from the ego vehicle. In other words, because the representation is based on the ego vehicle's local frame, without

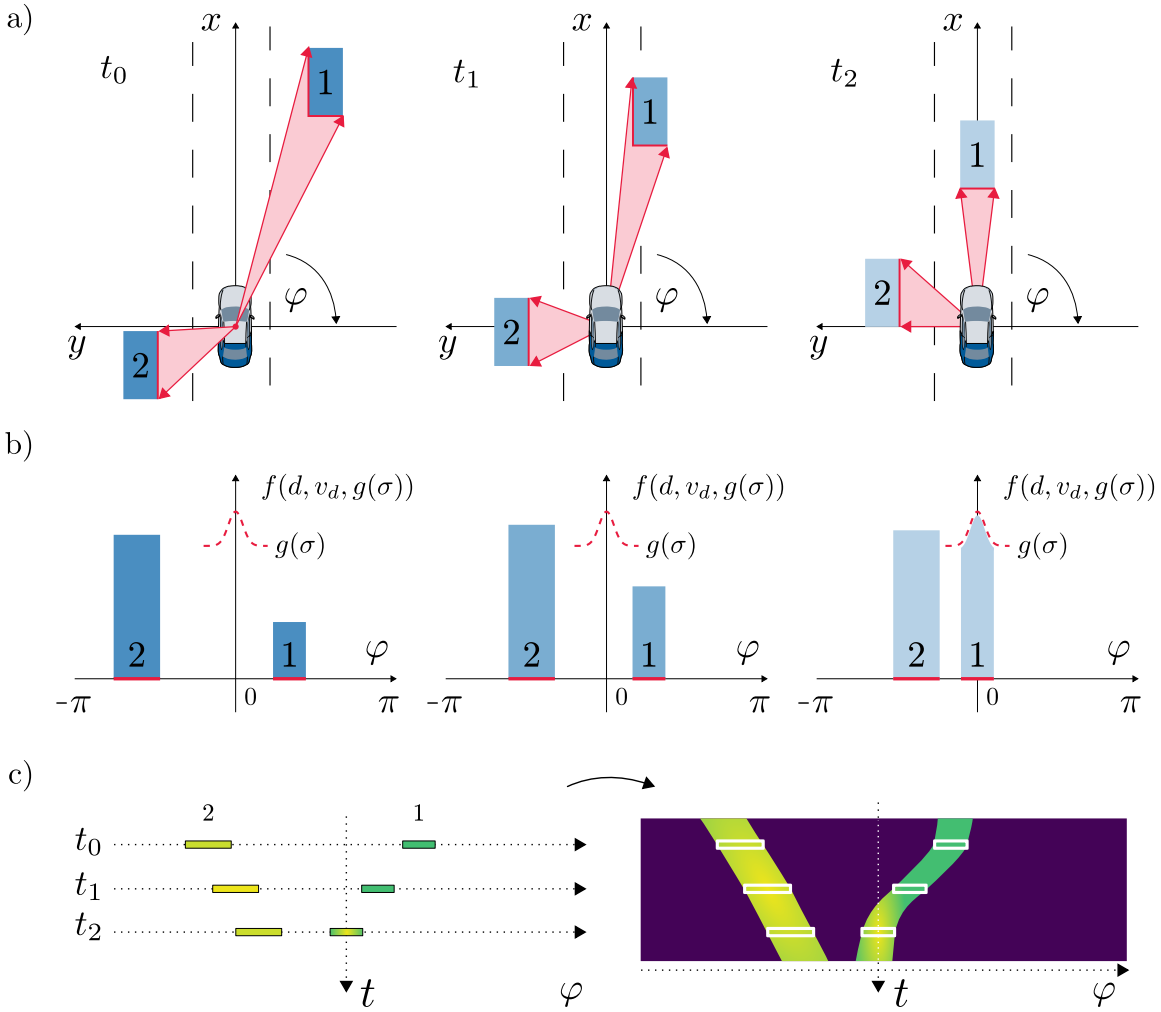

Figure 6.2: *Design of the Polar Occupancy Map representation. a) For each time-step* $[t_0, t_1, t_2]$ *the field of view of all vehicles is shown. b) Each field of view is transformed from the local ego coordinate system* (x, y) *to a new angle-based coordinate system* $(f(d, v_d, g(\sigma)), \varphi)$*, where the* φ *axis tracks the field of view for all vehicles. On the vertical axis we introduce a general function* f *which is used to manipulate the attention of the NN. Considering the classes we want to recognize, we chose as an input the distance* d *from the ego vehicle, together with the ego lateral velocity* v_d *and a scaled Gaussian function* $g(\sigma)$*. c) This representation is a top view of time-steps from figure b), where the values of the function* $f(d, v_d, g(\sigma))$ *are denoted by color. Yellow corresponds to higher and green to lower values. By filling in all intermediate time-steps we get a distinct trace through time which can be used as an input for the NN.*

the information of the lateral movement NNs would not be able to differentiate between movements of other vehicles and the movement of the ego vehicle. By following these design choices, the function f is defined as:

$$\hat{d}_i = k \cdot d_i + n \tag{6.1}$$

$$f_i(d_i, v_d, g(\sigma)) = \hat{d}_i + |v_d| + g(\sigma) \tag{6.2}$$

where d_i is the minimal distance to the vehicle in $[m]$, v_d is the ego vehicle's lateral velocity in $[\frac{m}{s}]$, and k and n are coefficients that create a linear transform of the distance d_i. Vehicles are recorded to the POM representation if the distance is less than 75m, which limits d_i to $[0, 75)$. The coefficients k and n are chosen such that the value $\hat{d}_i$ is bounded between 0 and 1, and when the distance d_i decreases the new value $\hat{d}_i$ increases, i.e., k is negative. This effectively shifts the attention of the neural network to vehicles that are closer to the ego vehicle. The coefficients are chosen as $k = -0.011$ and $n = 1$. The inverted distance $\hat{d}_i$ is only applied when a vehicle is present, whereas the lateral velocity v_d is applied on the whole representation making it possible to detect lane changes when no vehicles are visible. The last argument is the function g which is a scaled Gaussian function centered around zero degrees, where $\sigma = 0.005$ and $max(g(\sigma)) \approx 0.9$. This function is only applied if a vehicle is visible and is located around zero degrees, giving it a higher value and turning the focus of the network to it.

The summation of $\hat{d}_i$ and the absolute value of v_d could mean that a vehicle is either close to the ego vehicle or that the ego vehicle itself is moving. This will not be a problem in our representation for two reasons. First, if the vehicle is really close to the ego we will know that by the size of the field of view, as closer vehicles have wider fields of view and vice versa. Second, if there is lateral movement of the ego vehicle then all fields of view are moving, which can be detected and distinguished by the NN. Furthermore, the summation could also be interpreted as information loss. However, the task of the network is to detect scenarios and report at which time-step they occurred. If the exact state values of the traffic participants are needed, one could easily go to the specific time-step and read it from the available data.

At this step we have already created a representation with a fixed feature size. The 2D plots could be stacked creating a 3D plot with a fixed number of time-steps that can be used as an input to the network. However, in the recent years there has been a significant breakthrough in pattern recognition using CNNs [93]. That is why we use a top view of the 2D plots, as shown in Figure 6.2 c), and stack them in such a way to create an image. The color of the pixels corresponds to the value of the function $f_i(d_i, v_d, g(\sigma))$. By filling in the missing time-steps and assigning a value of zero to non-occupied parts of the field of view, we obtain the final form of the representation as shown in Figure 6.2 d).

Examining the final image, it can be seen that the vehicles create unique traces which are recorded by the POM representation without loss of relevant information. Finally, the traces can be detected and distinguished by the CNN, leading to accurate classifications.

In this approach, there are two further benefits for the extraction of relevant information that need to be addressed. The first one is that for vehicles that are far away from the ego vehicle the occupied angle is getting smaller. This naturally leads the

network to not focus on them. In addition, because we are more interested on what is happening in front and on the sides of the ego vehicle than in the back, we use different resolutions for the POM representation. A resolution of 4 pixels per degree is used on the front from $-10°$ to $10°$, 1 pixel per degree on the sides from $-30°$ to $-10°$ and $10°$ to $30°$. Finally, 0.5 pixels per degree on the back $-180°$ to $-30°$ and $30°$ to $180°$. This selection of resolutions leads to the final size of 270 values per time-step for the POM representation.

The second benefit is that only the first vehicle that occupies a certain field of view is recorded. As we are mostly interested in the closest vehicles, this further decreases the amount of information whilst preserving the relevant parts. In addition, if the ego vehicle is only equipped with camera sensors then this behavior occurs automatically.

6.4 Data Generation

In order to train a neural network, a lot of training examples are required. This is generally a very time-consuming task as we first need to record the data and then label it accordingly. However, because we are using an abstract representation of the environment, which can be replicated in simulation easily, we decided to generate synthetic labeled data for training and then apply the trained NN to real recordings. This way only a subset of recorded data needs to be labeled for validation.

6.4.1 Simulation Environment and Scenario Classes

The simulation environment consists of a three lane highway with three to four participants including the ego vehicle. The exact number of participants depends on the chosen scenario. The behavior and dynamics of the vehicles are programed in Python and scenarios are generated by selecting a set of desired actions. The actions are carried out either by the ego vehicle or the other participants. After defining the desired actions, it is possible to randomize the scenario by varying starting conditions, actions times etc. For this study we chose nine common highway scenarios, as shown in Figure 6.3.

The classes are:

1. *Driving in lane* - The ego vehicle is driving in a lane and there are no maneuvers from other vehicles;
2. *Lane change* - The ego vehicle is switching between free adjacent lanes;
3. *Ego vehicle free lane cut in* - The ego vehicle is cutting in from a free to an occupied lane;
4. *Ego vehicle free lane cut out* - The ego vehicle is cutting out from an occupied to a free lane;
5. *Ego vehicle cut in-out* - The ego vehicle is moving between two occupied lanes, hence making a cut in-out;
6. *Target ahead free lane cut in* - A vehicle is cutting in from a free lane to the ego vehicle's lane;

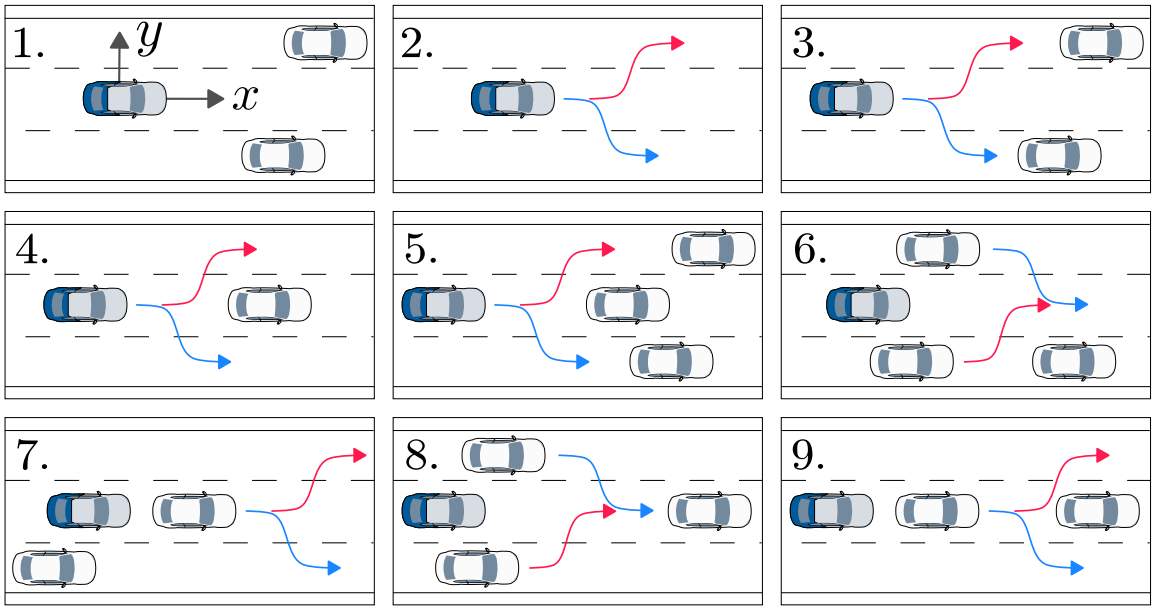

Figure 6.3: *Common dynamic traffic scenarios found on highways*

7. *Target ahead free lane cut out* - A vehicle is cutting out from ego vehicle's lane to a free lane;
8. *Target ahead switch cut in* - A vehicle is cutting in between the ego and another vehicle, which leads to the switch of the current TA;
9. Target ahead switch cut out - A vehicle that was between the ego and another vehicle cuts out to a free lane leading to a switch of the current TA.

All classes should be detected by the neural network irrespective of the shape of the highway. This is achieved by using the Frenet-Serret frame instead of a global frame, as shown in Figure 6.4. The used Frenet-Serret frame is defined by the axis s denoting the length of the path traveled on the highway and axis, d denoting the orthogonal distance from the middle of the highway. In other words, the transformation between the global and Frenet-Serret frame decouples the motion of the vehicles from the shape of the highway.

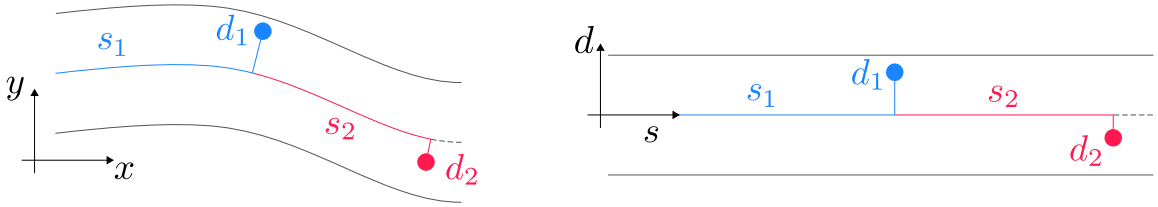

Figure 6.4: *Transformation from a global to the Frenet-Serret frame*

To calculate the transformation from the global to the Frenet-Serret frame, information about the highway curvature is needed. This information is usually available from a map in which the vehicle is localized using Global Navigation Satellite System (GNSS). Alternatively, a camera system could be used to enhance the curvature measurement by tracking the road or lane marking curvatures [94].

In addition, the yaw rate of the ego vehicle relative to the Frenet-Serret frame will influence the POM representation. However, in this use case it was regarded as noise because the relative yaw rate is small on highways and occurs only in the ego lane change scenarios.

6.4.2 Scenario Examples

Figure 6.5 a) and b) show three randomly generated scenarios. The scatter plot shows the motion of the ego vehicle (blue-green) and other participants (red-yellow) during 7 seconds in the highway's Frenet-Serret frame. The images below the scatter plots show the corresponding POM representations. Figure 6.5 b) shows the first channel of the HiG representations for the same scenarios as in a). The HiG's first channel is an aggregation of VeGs constructed for each time-step. The VeG size is fixed to $60 \times 20 \times 3$ and the ego center point is moved back to give more focus to the front of the vehicle. The resolution of the VeG is two meters per pixel on the x and one meter per pixel on the y axis.

For each generated scenario, starting positions, lanes, maneuvers and velocities are generated randomly but satisfy several conditions to assure that the specific scenario is possible. Furthermore, uniform noise was added to the motion of the vehicles to generate more realistic scenarios.

In the first scenario, Figure 6.5 a, b), a right cut in maneuver by the ego vehicle is shown. There are two constraints in this case. First, the ego vehicle cannot be on the most right lane because a right lane change would not be possible. And secondly, there should be a vehicle in front in the desired lane. Similar constraints apply to all scenarios.

In the second scenario, a TA free lane cut out is shown. It is important to notice the difference between the values of the $f(d, v_d, g(\sigma))$ function for this and the previous scenario. In this case, the ego vehicle's lateral velocity v_d is zero and the function $f(d, v_d, g(\sigma))$ only depends on the distance and the g function. However, in the first scenario the ego vehicle had lateral velocity and we can see in the POM representation that a unique pattern was created. Without this pattern it would be impossible to distinguish the first scenario from a *target ahead free lane cut in* scenario.

The third scenario represents a TA switch cut in. Examining the POM representation, it can be seen that there was a vehicle in front of the ego vehicle and after some time a new vehicle made a cut in. Furthermore, we notice how the function $g(\sigma)$ is highlighting the vehicle which is near zero degrees, i.e., in the ego vehicle's lane.

For certain scenarios, e.g., TA free lane cut in, it is enough to have one other participant together with the ego vehicle. However, an additional vehicle which was randomly positioned on the highway was used. This was done in order to teach the network to ignore other vehicles if, they are not relevant to the desired classes. It is important to note that in simulation we will never be able to cover all possible scenarios that can occur in the real world. However, ML driven approaches give us the ability to iteratively improve the models. With time, we will be able to find out scenario where the models are uncertain and where more training is necessary.

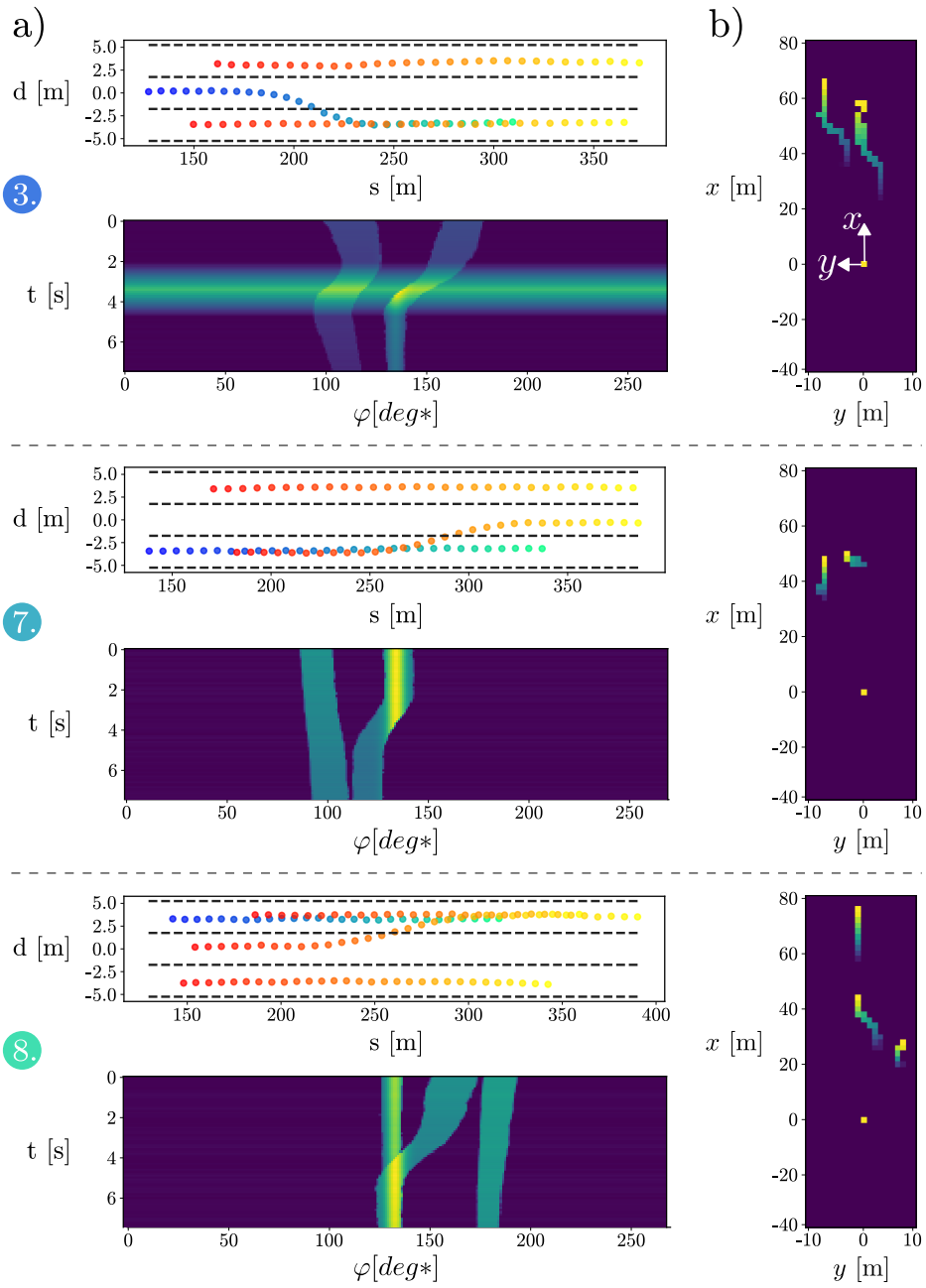

Figure 6.5: *a) Randomly generated scenarios (3., 7. and 8.) with a duration of 7 seconds. Each column consists of a scatter plot showing the motion of the ego vehicle (blue-green) and other participants (red-yellow) on the highway, and the corresponding POM representation. b) First channel of the HiG grid map for the same scenarios as in a)*

In total, we created 15000 random scenarios where 10% is used for the test and 10% for the validation set. Each scenario has a duration of 20 seconds with a sampling rate of $10Hz$. The labels are generated automatically for each time-step. The generated data includes the proposed POM representation together with the Grid Map representations VeG, SVeG and HiG, which are going to be used for performance comparison. Please

note that the scenario generation code, network implementations, and training data are open-source and can be found at [95].

6.5 Scenario Classification

The proposed POM approach creates a representation which is suitable for training CNN. These networks excel in extracting features and recognizing patterns in images. However, to increase the performance compared to the classical CNNs [93], we used several enhancements introduced in recent studies.

Dilated Convolutions were proposed by Yu *et al.* [81] and the main idea is to increase the receptive field of the CNN without adding additional layers. The problem with additional layers is that they increase the overall size of the network and are much harder to train. In a normal convolutional network, the receptive field increases linearly with each layer; however, if we introduce kernel dilation, we can achieve an exponential increase of the receptive field.

K. He *et al.* [82] worked on *Residual Connections* to mitigate the vanishing gradient problem. They introduced skip connections between layers allowing the gradients to propagate more easily through the network. In addition, Veit *et al.* [83] argue that the residual connections increase accuracy because they allow the information to flow through more paths.

The final enhancements that were used are the *Self Normalizing Layers* and *Alpha Dropout*. It is widespread practice to normalize the input data into the network. However, during the training individual layers of the network can change the distribution slowing down the learning process. One solution would be to use *Batch Normalization* after each layer, proposed by Ioffe *et al.* [84]. This approach, however, adds additional complexity to the network. A different solution was proposed by Klambauer *et al.* [85] where a new activation function called SELU was introduced. With the new activation function, outputs from each layer converge to a zero mean and unit variance without any additions. If dropout is necessary, then *Alpha Dropout* [85] is used to preserve the output distribution.

The proposed neural network consists of 9 convolutional layers, combining dilation and residual connections. The convolutional layers represent the feature extraction and are followed by 4 dense layers producing the class probabilities. Each layer uses the SELU activation, and *Alpha Dropout*, with a dropout rate of 10%, is used on the final dense layer. For the evaluation, 50 time-steps, i.e., POM representations, were used leading to the final input size of 50×270 values. The full network architecture is given in Table 6.1.

6.6 Results

The performance of the POM based neural network is evaluated in two steps. First, we compare its ability to extract the desired classes against the Grid Map based models introduced by Gruner *et al.* [78], namely SVeG and HiG. In addition to these two rep-

Table 6.1: *Model architecture for scenario classification with dynamic traffic © IEEE 2019*

layer	kernel	stride	dilation	residual	input size
conv1	3×3	1×1	1×1	no	$50 \times 270 \times 1$
conv2	3×3	1×2	1×1	no	$50 \times 135 \times 32$
conv3	3×3	1×1	1×1	yes	$50 \times 135 \times 32$
conv4	3×3	1×2	1×1	no	$50 \times 68 \times 32$
conv5	3×3	1×1	2×1	no	$50 \times 68 \times 32$
conv6	3×3	2×2	1×1	no	$25 \times 34 \times 64$
conv7	3×3	1×1	4×1	yes	$25 \times 34 \times 64$
conv8	3×3	2×2	1×1	no	$13 \times 17 \times 128$
conv9	3×3	2×2	1×1	no	$7 \times 17 \times 128$

dense layers	conv8	dense1	dense2	dense3	dense4
size	8064	1024	512	256	9

resentations, we also introduced a fully-stacked VeG model which we call Fully Stacked Velocity Grid (FVeG) and a regular CNN network (POM-CNN) without residual connections and dilation. The second step is an evaluation of the trained model on real object data captured by the Mobileye camera [96]. The camera is placed on the front of the vehicle and has a total field of view of around 90°. The simulation data was adjusted to match the field of view of the camera and make the data more realistic.

The generated training data consist of 15000 scenarios with a duration of 20s and a sampling rate of $10Hz$. For each sample the POM and VeG representations are created. To process the whole scenario, a sliding window approach is used. The window size is fixed to 50 time-steps and the networks generate the predictions in the middle of the window. This prediction scheme gives the networks the ability to take into consideration past and future samples in order to produce the correct class label, leading to higher accuracy. The sliding window is moved across the whole scenario generating the class probabilities for each subsequent time-step.

The SVeG model uses the first and last VeG from a given window. The HiG model aggregates all VeGs occupancy in the first channel and uses the remaining 2 channels from the last VeG in the window. The FVeG stacks together all VeGs from a given window leading to the largest input size of $60 \times 20 \times 3 \times 50$.

Table 6.2: *Test and validation accuracy on generated data © IEEE 2019*

accuracy	SVeG	HiG	FVeG	POM-CNN	POM
test	87.76%	85.04%	93.82%	95.26%	96.17%
validation	88.10%	85.70%	94.80%	95.30%	96.00%

The POM, HiG, SVeG and FVeG networks share the same architecture from Table 6.1 with different input sizes. The POM-CNN uses the same number of layers but does not include residual connections or dilation. All networks were trained for 2304 steps using a batch size of 128 windows and a decaying learning rate. The networks were initialized using the variance-scaling initializer [97] and optimized with the ADAM [98] optimizer.

Table 6.2 shows the performance of the models on the test and validation set. We can see that the SVeG and HiG models have the lowest performance. The drawback of the SVeG model is that it only uses the first and last VeG from a window, which leads to significant information loss. The HiG model does aggregate the occupancy of other vehicles; however, because of the way the HiG representation is constructed some information can still be lost as parts of previous maneuvers can be overwritten with the more recent ones. The FVeG representation performs much better because all time-steps of the sliding window are visible to the network. However, the drawback of the FVeG representation is the big input size and longer training times. The proposed POM models have both the best test and validation accuracy. With residual connections and dilation, the POM network shows better performance compared to the regular POM-CNN. Moreover, the POM network uses a 27 times smaller input than the FVeG.

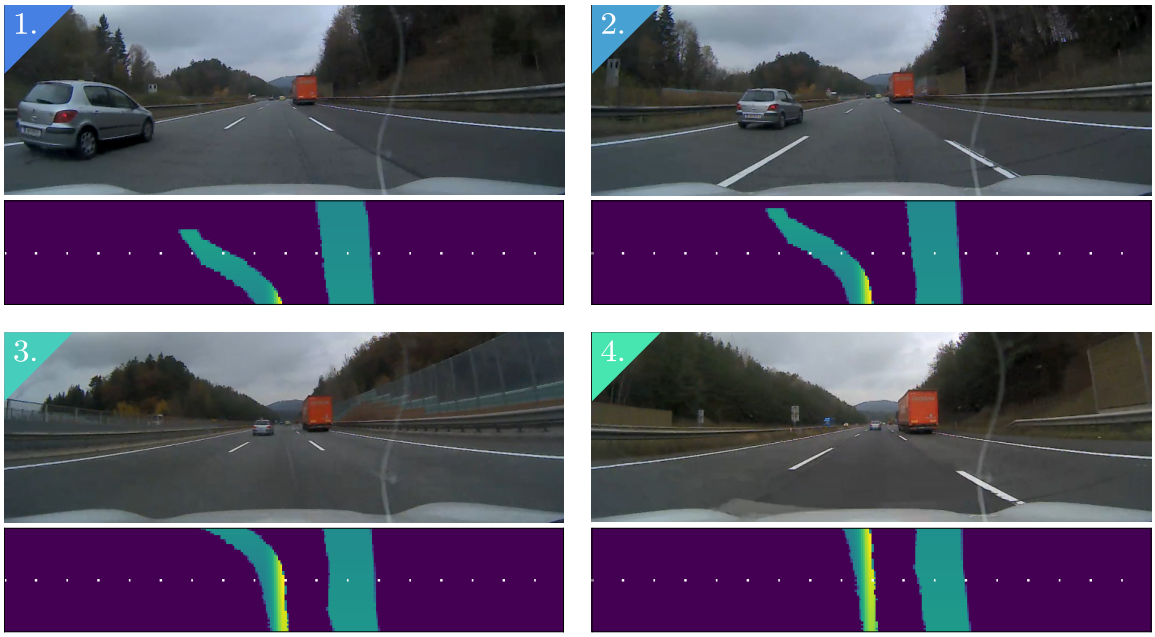

Figure 6.6: *Evaluation of the POM representation on real data using the Mobileye camera. Each image pair shows a camera frame and its corresponding POM representation. The POM representation has a window size of 50 samples. The white dashed line indicates the middle of the sliding window, where the prediction is made. The sequence shows a Target ahead free lane cut in. © IEEE 2019*

Finally, the POM network was evaluated on real object data obtained with the Mobileye camera. The test run was done on the A9 highway near Graz, Austria. A sample

TA free lane cut in scenario from the recorded data is shown in Figure 6.6. Each pair shows the camera frame and the corresponding POM representation for the current time window. We can see how the motion of the cutting-in vehicle is tracked through the representation. The performance of the pre-trained POM model was evaluated using 2900 sequential time windows. The recordings contained curves and they were compensated using the camera's lane curvature information. In order for a scenario to be extracted, the model certainty for that scenario needed to be higher than 95%. With this extraction method we were able to lower the misclassifications and obtain an overall detection accuracy of 90.8%. All data and models used in this research are open-source and available at [95].

6.7 Chapter Conclusion

In this chapter, we introduced a novel traffic representation that can be used to train neural networks for scenario classification. As the input size to the neural network needs to be constant, we designed a representation that is independent from the number of traffic participants in the scenario. The representation is based on the polar occupancy map and we have shown that it is capable of creating unique patterns which can be used for scenario classification. To evaluate the representation, a CNN was trained on 9 highway scenarios, resulting in high classification accuracy. The proposed model was compared to existing traffic representations, showing that it offers improved performance with a more compact representation. Finally, we have shown that the model trained on simulated data has generalized well and that it can be used to make high accuracy predictions on real data.

7

Scenario Exploration

In this chapter, we focus on unsupervised learning methods used for scenario exploration. The unsupervised DL models try to group similar scenarios/signals together in a latent space. By examining this latent space, we can get insights on the distribution, sizes and relevance of individual clusters and their corresponding scenarios. In addition, by examining the distance between scenarios in the latent space, we can get information about the scenarios that are most similar to a query scenario or create some uniqueness score.

7.1 Introduction

In the previous chapters, we have shown how DL can be used to extract relevant scenario if the models were previously trained on labeled data. The models learn the relationship between input data and the desired classes. However, it is not always a straightforward task to defile the labels. If we look back at the example on the highway classes from **Chapter 6**, only nine of them were defined. Those classes represent the most frequent ones; however, there are many more possible scenarios. If we wanted to add a new scenario, we would have to label data for that specific scenario and then retrain the models. Depending on the type of data, labeling can be very challenging or time consuming.

Another way to define scenarios would be to examine the recorded driving data and see if some interesting scenarios occur. Manual examination of scenarios is also not a trivial task. We would have to look at hours and hours of data and try to make some rules how they should be grouped together. This would also change depending on the types of signals we are considering. Instead of doing all this work manually, we could try to use DL models to look at the data and group signals depending on some similarity. Moreover, we can restrict the DL models to generate the groups on a low dimensional space, so that visual exploration is possible. In this chapter, we propose such a clustering approach using contrastive learning and compare it to clustering using Autoencoders (AEs).

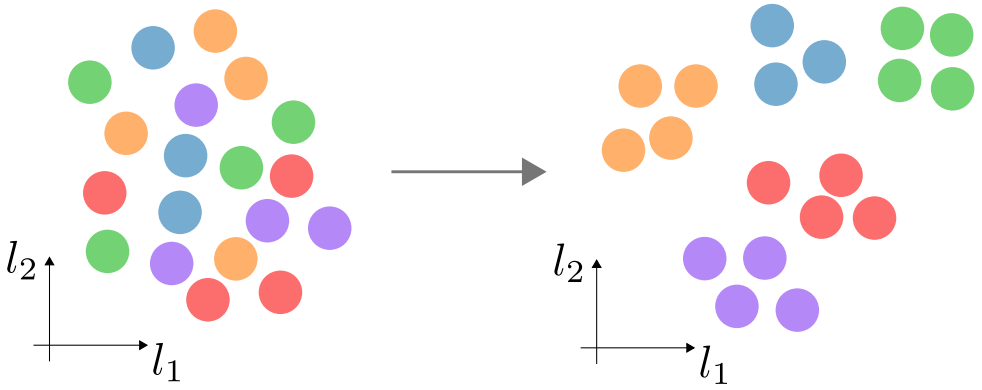

Figure 7.1: *Clustering example with one feature and a 2D latent space.*

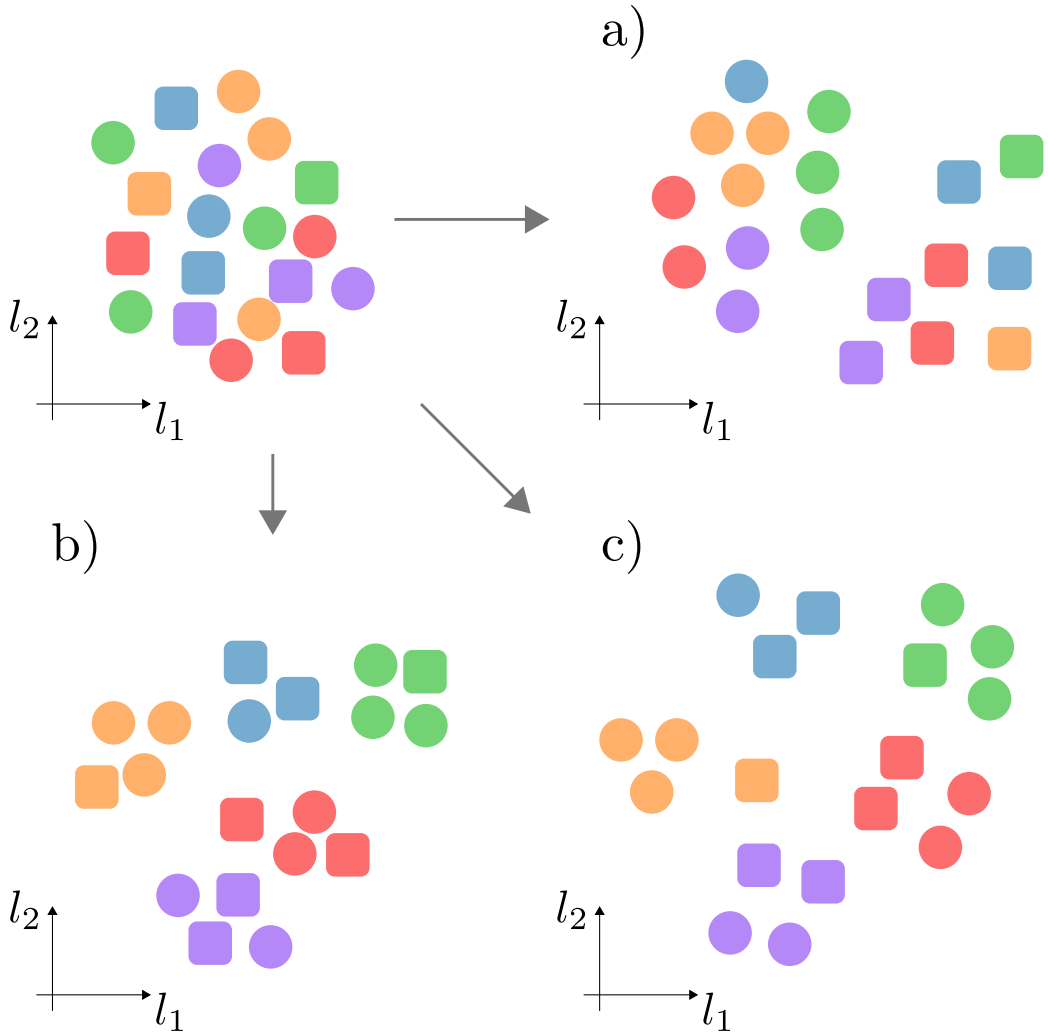

Figure 7.2: *Clustering example with two features and a 2D latent space. Subfigures a), b) and c) show different ways how the DL models could do the clustering.*

To better understand the task of grouping similar scenarios/signals, we will use two intuitive examples, shown in Figure 7.1 and Figure 7.2. If we were given some objects to cluster, we would use the latent space and try to group together instances with similar features. Figure 7.1 shows objects with one feature (color) that is easily differentiable. However, in real world examples the features can not be separated so easily and the boundary between features is not always clear.

Figure 7.2 shows another example with two features: shape and color. As we can see, in this case, the clustering is not straightforward. There are several ways how the objects can be grouped together. In Figure 7.2 a) the objects are grouped by shape. In Figure 7.2 b) the objects are grouped by color and in Figure 7.2 c) the objects are grouped by shape and color. In c), the grouping is done first by color and then by shape - the squares are facing inwards. All of the groupings in Figure 7.2 are correct, it only depends what grouping do we actually want. If no additional information is given to the DL methods, then any of these grouping could occur depending on the random initialization. However, if we give some additional information, we can guide the models to cluster the objects by our liking.

In this chapter, we will show both approaches - clustering with and without additional information. Clustering without any information is done using AE [99], more specifically Convolutional Autoencoders (CAEs) [100]. Clustering with information which signals should be grouped together is done using "Contrastive Learning" [101],[102]. Moreover, we will show that the proposed clustering with contrastive learning shows better results when applied on scenarios related to lane change signals.

7.2 Related Work

AE based models have been used in several applications regarding automated driving, such as feature extraction, behavior classification and modelling. This kind of approach was introduced by Sama *et al.* [103]. They used real world data, collected from a residential area, to extract velocity driving styles. The driving styles were clustered in an unsupervised way and nine clusters were defined. These nine clusters were used to create models that can be applied by an autonomous vehicle to reproduce the desired driving styles.

A similar modelling approach was introduced by Krajewski *et al.* [104]. They have used a different type of AE called Variational Autoencoder (VAE)[105]. VAE is a generative model, and Krajewsi *et al.* have used it for trajectory modeling used in safety validation of automated vehicles. The VAE model was trained on lane change trajectories, extracted from the HighD [26] dataset, to learn parameters that describe them. In their research, they tried both unsupervised and semi-supervised training to learn the parameters.

Our work differs from the aforementioned research as we are considering a more general approach for scenario exploration. The unsupervised clustering is not bounded to specific use cases and can be applied on any type of scenarios. Moreover, the latent space we are using does not exceed three dimensions to allow visual inspection. The research proposed in this chapter is serving as an aid for engineers to identify, label

and inspect interesting scenarios from real data.

Another application of AE is uniqueness estimation of new scenarios. Langner *et al.* [106] have used AE models to estimate uniqueness of scenarios related to a Predictive Cruise Control ADF. They have used the reconstruction loss as a measure of the uniqueness. If the network was trained on a specific scenario, then the reconstruction error is low as the network knows how to reconstruct the input. However, if the scenario is new and the network was not trained on it, then the reconstruction loss would be high as the network did not learn how to reconstruct it properly. As many driving scenarios are very similar and encountered repeatedly, Langner *et al.* used this approach to select only parts of the recorded data that could be interesting for testing.

In this chapter, we are going to estimate the uniqueness of scenarios based on their neighbourhood in the latent space. If the neighbourhood in the latent space is sparse, that means that there were very few similar examples in the dataset and vice-versa.

7.3 Time series clustering using DL

In this section, we are going to explain in detail the methods that were used for scenario exploration, model behavior exploration and uniqueness estimation. We are going to compare two different approaches. The first approach uses AE based models together with the t-distributed Stochastic Neighbor Embedding (t-SNE) dimensionality reduction. The second approach is the proposed clustering using *Contrastive Learning*.

7.3.1 Autoencoders and Convolutional Autoencoders

AEs [99] are models that are used for unsupervised representation learning. They take an input x and first transform it to a desired latent space. Then, from the latent space, the models try to reconstruct the original signal, denoted as $\hat{x}$. This can be expressed with the following equation:

$$\hat{x} = g(f(x)) \tag{7.1}$$

$$loss = \mathcal{L}(x, \hat{x}) \tag{7.2}$$

where, $f(\cdot)$ is the encoder part and $g(\cdot)$ is the decoder part of the model. $\mathcal{L}$ is the reconstruction loss between x and $\hat{x}$ and we are going to use the Square Error (SE) loss. During training, the model tries to reduce the reconstruction loss by learning the mappings $f(\cdot)$ and $g(\cdot)$.

If the latent space dimension is smaller than the dimension of x, then the model has to compress and decompress the information. Such an example is shown in Figure 7.3. By compressing and decompressing the information the model is forced to learn characteristic features of the input signal x and put similar signals close to each other in the latent space. Similarly, different signals are going to be put further apart in the latent space.

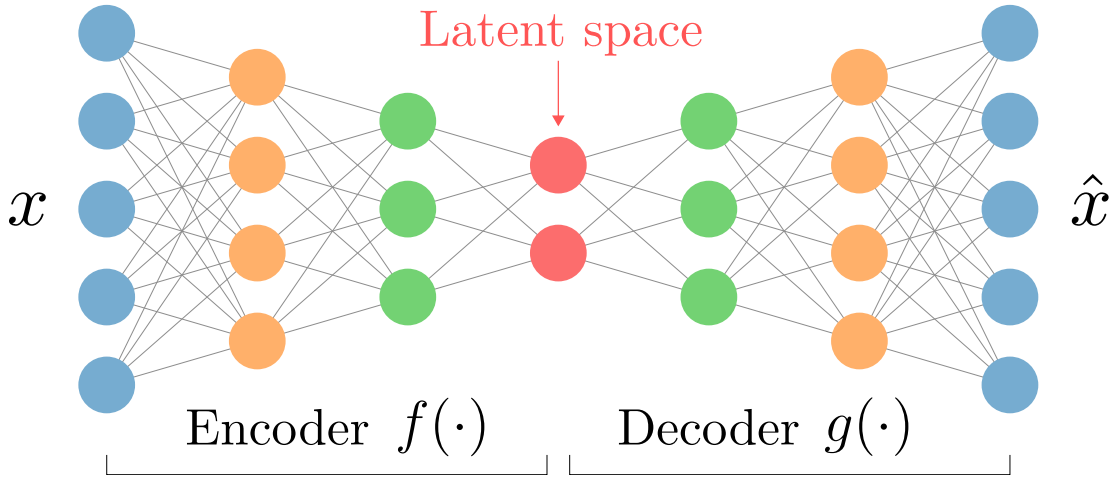

Figure 7.3: *Autoencoder structure. The left part of the network is called the Encoder and it transforms the input x into a lower dimensional latent space. The right part of the network is called the Decoder and it reconstructs the input signal from the latent space.*

The CAE [100] is a special type of AE where the *Encoder* and *Decoder* networks are constructed from convolutions rather than fully connected layers. These types of models are well suited for feature extraction from input signals. Figure 7.4 shows an example with a two dimensional input. Similarly to AE, the information is compressed to a latent space an then decompressed again. In this chapter, we will use CAE in the first layers for feature extraction and follow them with several fully connected layers for the latent representation.

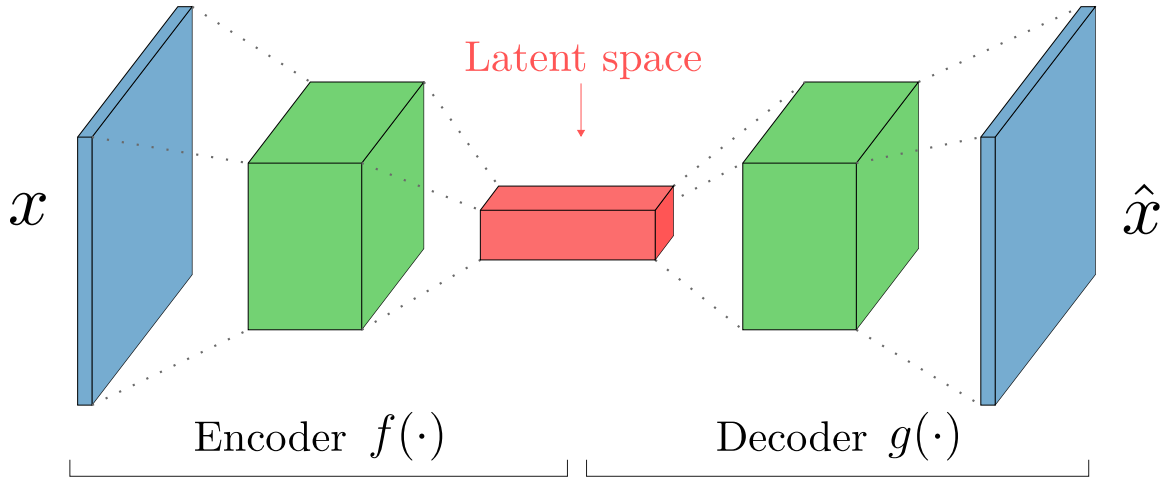

Figure 7.4: *Convolutional Autoencoder structure. Similarly to the AE, the CAE transforms the input x to a latent space and then back to the reconstructed output $\hat{x}$. However, instead of using fully connected layers it uses convolutions for both the Encoder and Decoder.*

The reconstruction accuracy of the AE models depends on the model and latent space size. If we use a bigger model, then the latent space size can be smaller and vice-versa.

A bigger latent space means that the model can use more information when encoding the input. In general, the latent space size is bigger than three, which means that we need to further reduce the dimensionality of the latent space if we want to visualize it. In this research, we will use t-SNE which takes as an input the latent space of the AE model and reduces it further to two dimensions. In this case, the AE model serves as a feature extractor for t-SNE which does the final clustering.

7.3.2 t-distributed Stochastic Neighbor Embedding

t-SNE [107] is a non-linear dimensionality reduction technique used for visualization of higher dimensional data. The main idea is to preserve the local similarity between points in the higher dimensions and transfer it to the lower dimensions, as shown in Figure 7.5. The local similarity is preserved by creating a probability distribution between each pair of points. Points that are close to each other have a high probability, whereas points that are further away have lower probability. The probability in the higher dimensions is modeled as a Gaussian distribution. This can be expressed with the following equations:

$$p_{j|i} = \frac{exp(-\left\|x_i - x_j\right\|^2 / \, 2\sigma_i^2)}{\sum_{k \neq i} exp(-\left\|x_i - x_k\right\|^2 / \, 2\sigma_i^2)}, for \; i \neq j \tag{7.3}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{7.4}$$

$$p_{ij} = p_{ji}, \; p_{ii} = 0, \; and \; \sum_{i,j} p_{i,j} = 1 \tag{7.5}$$

where $p_{j|i}$ is the conditional probability of points x_j and x_i. It denotes the probability that the point x_i will pick x_j as its neighbour. σ_i depends on the density around x_i. Smaller values of σ_i correspond to higher density and vice-verse.

For the lower dimension a Student t-Distribution is used:

$$q_{ij} = \frac{(1 + \left\|y_i - y_j\right\|^2)^{-1}}{\sum_{k \neq i}(1 + \left\|y_i - y_k\right\|^2)^{-1}}, for \; i \neq j \; and \; q_{ii} = 0 \tag{7.6}$$

where y_i and y_j are the lower dimensional mappings for x_i and x_j respectively, and q_{ij} is their similarity.

To get mappings that preserve the local similarity for the original data, a gradient descent optimization is used on the Kullback–Leibler divergence of distribution P from distribution Q, which reads as

$$KL(P|Q) = \sum_{i \neq j} p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{7.7}$$

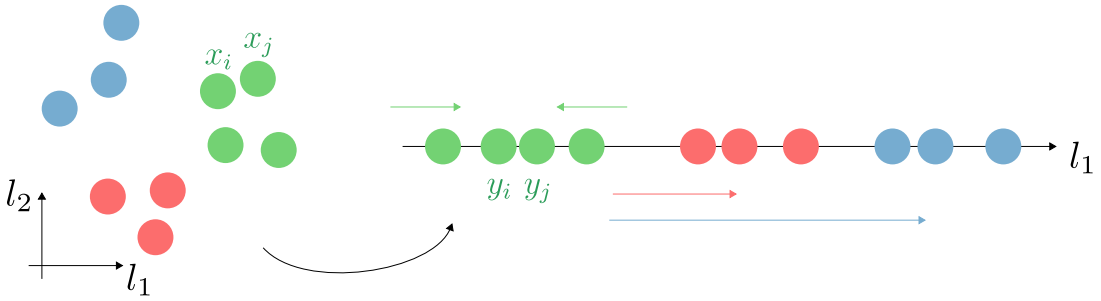

Figure 7.5: *t-distributed Stochastic Neighbor Embedding example. Data points that are close to each other in the higher dimensions are pulled together in the lower dimensions. At the same time, data points that are further away in the higher dimensions are pushed away.*

7.3.3 Clustering using Contrastive Learning

Contrastive learning [101], [102] is a learning technique that tries to pull together latent representations of similar inputs, whilst pushing away dissimilar ones. There are several implementations of this technique where the similarity of the inputs is defined either manually or automatically. As we want to do unsupervised clustering, we are going to use the approach where the similarities are generated automatically by data augmentation.

The main idea of this approach is shown in Figure 7.6. We start by randomly sampling a batch of inputs from the data. Then, for each input, two augmentations are created. The task of the network is to learn to pull together the latent representations of the two augmentations that were created from the same input, while, at the same time, pushing away latent representations from other inputs of the batch.

In Figure 7.6, the task is to group object by their color regardless of their shape. To achieve this, the augmentations are changing only the shape of the object, and the network is learning to pull them together. It is important to notice that we do not need to label the data nor know the exact samples that are being processed. The only information we need to provide are the augmentations, and depending on them the clusters are going to be created.

To get this desired behavior, the network is trained on the following contrastive loss function:

$$\mathcal{L}_i = -log \frac{exp(sim(z_i, z_j)/\tau)}{\sum_{k=1,k\neq i}^{2N} exp(sim(z_i, z_k)/\tau)} \tag{7.8}$$

$$\mathcal{L} = \sum_{i=1}^{2N} \mathcal{L}_i \tag{7.9}$$

where, N is the batch size, and as there are two augmentations per input, we get $2N$ samples. z_i and z_j are augmentations that come from the same input. $sim(\cdot)$ is a similarity measure between the two latent presentations. τ is a temperature parameter

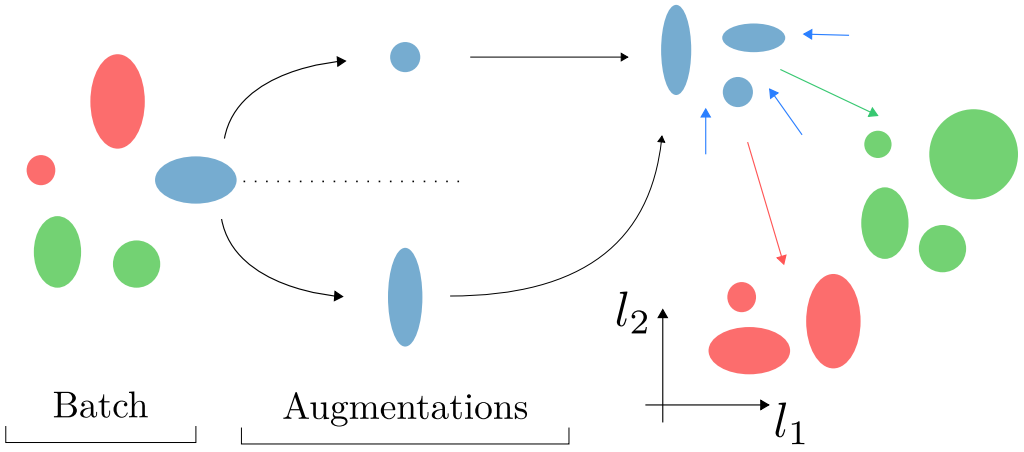

Figure 7.6: *Contrastive Learning example. First, a batch of inputs is randomly sampled from the data. Then, for all examples, two augmentations are created. The network tries to bring together the latent representations of the augmentations coming from the same input, and push away the other inputs from the batch.*

which controls how far apart are the representations in the latent space. The similarity of the two representations coming from the same input should be maximized and it is additionally normalized by all other augmentation pairs, except similarity between the same augmentation ($k \neq i$).

The intuitive explanation of the loss function is that we want to maximize the nominator while at the same time minimizing the denominator, which means that we want to maximize the similarity between augmentation pairs coming from the same input and minimize similarity with other augmentations. The loss is calculated for each pair of augmentations and the final loss is the sum over all individual losses.

Both the AE and Contrastive Learning method share the same idea: Pull together representations of similar inputs and push away dissimilar ones. However, because of the special loss function in the contrastive learning approach, we do not need to have a decoder network as in the AE case. Moreover, as the loss function is directly applied on the latent space, we do not need to reconstruct the original signal as in the AE method. The task of pulling together latent space representations is much easier than learning to compress and decompress the data. This means that we can try to directly use a 2D or 3D latent space, omitting the additional dimensionality reduction step.

7.3.4 Model Architectures

Table 7.1 shows the architectures used for the AE and contrastive learning based models. The left side provides details for the encoder and the right side for the decoder network. The contrastive learning model uses only the encoder part, whereas the AE model uses both the encoder and decoder network. Both the encoder and decoder use convolutional layers as feature extractors followed by dense layers for the latent space output. All networks use the ReLU [108] function for nonlinearity. The batch size was fixed to 250 and a constant learning rate of 0.0001 was used for all experiments. For

optimization the ADAM [98] optimizer is used. The loss function for the AE model is the SE. For the contrastive learning we use the inverse distance as a similarity measure and $\tau = 1$. Both models were trained for 12000 epochs.

We did two experiments with the AE model. In the first one, we use a latent space of size 8 and then we apply t-SNE to lower it to two dimensions for visualization. In the second experiment, we use a latent space of size 2 so that we can directly compare it to the contrastive learning model.

Table 7.1: *Model architectures for scenario exploration.*

ENC	kernel	stride	input size
conv1	1×5	1×2	$50 \times 270 \times 1$
conv2	1×5	1×2	$50 \times 270 \times 1$
conv3	1×3	1×2	$50 \times 135 \times 32$
conv4	1×3	1×2	$50 \times 135 \times 32$
dense1	-	-	128
dense2	-	-	64
latent	-	-	2 or 8 - (AE)

DEC	kernel	stride	input size
latent	-	-	2 or 8 - (AE)
dense1	-	-	64
dense2	-	-	128
conv1	1×4	1×2	$50 \times 270 \times 1$
conv2	1×5	1×2	$50 \times 270 \times 1$
conv3	1×5	1×2	$50 \times 270 \times 1$
conv4	1×2	1×2	$50 \times 270 \times 1$

7.4 Use cases for Scenario Clustering

In this section, we will introduce three use cases related to scenario exploration. The first one is general scenario exploration without any labels for the signals. In this case, we have to manually look into the latent space and decide if there are any meaningful clusters. In addition, we can apply some cluster assignment methods to help us with the exploration. The second use case is model behaviour exploration. In this case, the labels are given by some classification model and we can explore the model's behavior and accuracy. In the third use case, we will explore the uniqueness of synthetic signals by looking at their position and neighbourhood in the latent space.

All use cases will be introduced using the signal called *Lateral Deviation*. This signal represents a normalized offset from the center of the current lane and it is bounded between -1 and 1. Negative values correspond to movements on the left side, and positive to the right side of the lane. If a lane change occurs, the value gradually goes to 1 or -1, as the vehicle moves away from the center of the lane. Then, at the lane boundary, the value jumps to -1 or 1, depending on the direction of the lane change. This signal is used for lane change estimation and in the second use case we will explore the behavior of a lane change classification model. (Example lane changes are shown in Figure 7.8 and 7.9.)

The *Lateral deviation* dataset consists of two hours of driving on highways and motorways. The dataset was recorded using the Mobileye camera [96] with a sampling rate of 10Hz. To precess the data, a sliding window approach was used. The window size

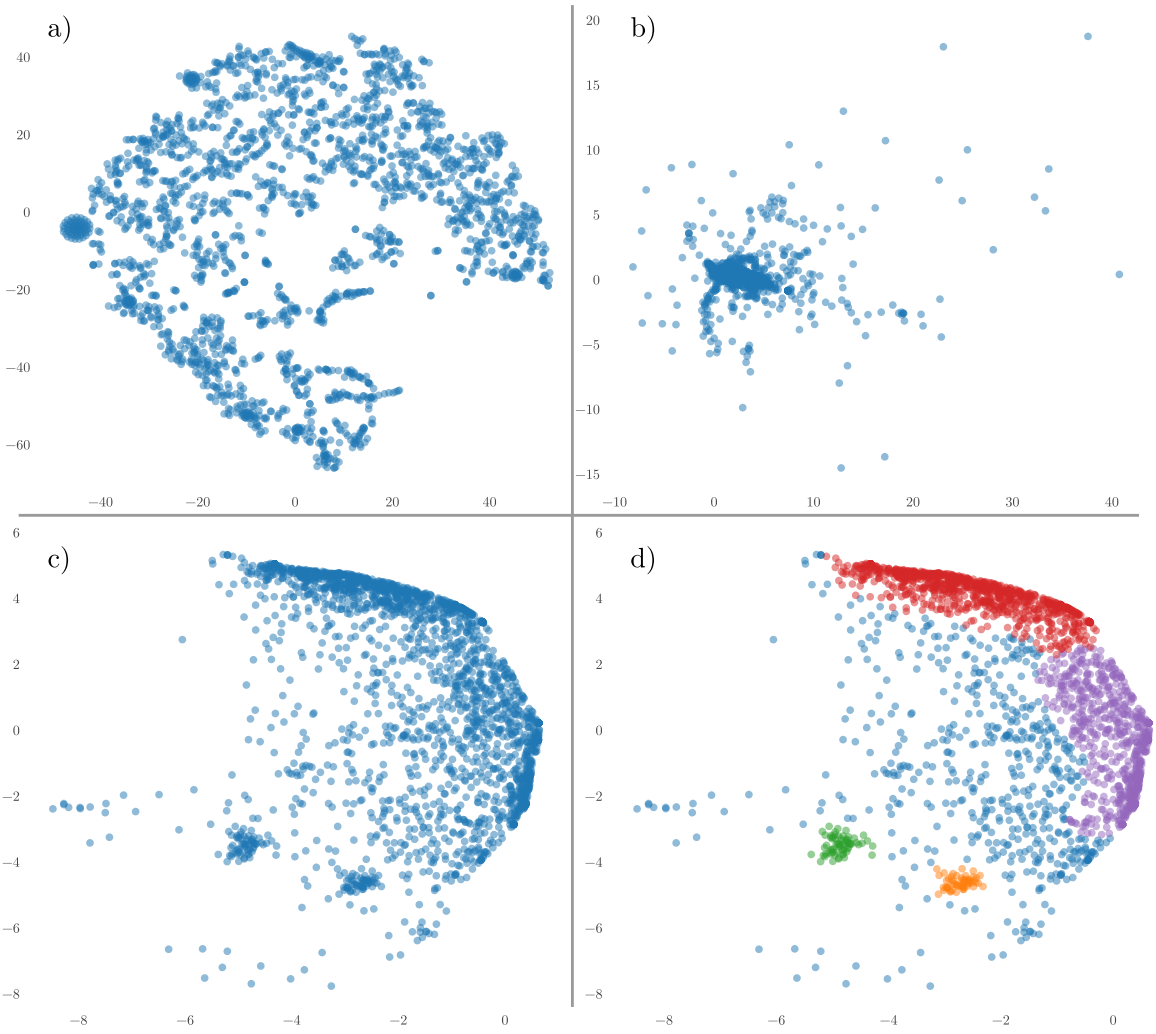

Figure 7.7: *Clusters of the Lateral Deviation signal. a) Clustering using AE and t-SNE. b) Clustering using AE with a two dimensional latent space. c) Clustering using contrastive learning. d) Example cluster assignments using DBSCAN [109].*

was fixed to 50 samples, which corresponds to 5 seconds per window, and the windows have a 50% overlap.

7.4.1 Scenario Exploration

As mentioned before, the main idea behind this use case is to explore scenarios without any information from labels. The proposed approach uses unsupervised clustering which groups data together based on their similarity. After the clustering, we can explore the groups and gain an insight on the distribution of the data, number of clusters, how the data was grouped together etc. This information can help us decide whether to create new scenario labels out of specific groups, or for example, use them for training and improving classification models.

The benefit of having a representation that can be visualized is that we can easily select, inspect and label groups of data. We have developed an interactive tool which makes is very easy to explore the latent spaces. Each point in the latent space is clickable, and after clicking on a specific point the corresponding signal is shown. In addition, with a selection tool the user can select many points simultaneously and compare their signals.

Figure 7.7 shows the clustering results on the *Lateral Deviation* dataset using three methods. Figure 7.7 a) shows the clusters obtained from the AE model combined with t-SNE. In this case, the AE model uses a latent space of size 8 and t-SNE further reduces it to two dimensions. We can see that some clusters were formed and that manual inspection could be beneficial.

In Figure 7.7 b), we can see that the clusters are not well separated. In this case, we used the AE model and directly applied it on a two dimensional latent space. As we can see, it is challenging for the model to learn how to compress and decompress the data with a latent space with only two dimensions. This example can not be used for further analysis.

The results of applying contrastive learning on the dataset are shown in Figure 7.7 c). We can see that the contrastive learning model has done a really good job at separating the data into clusters. The cluster sizes differ but we can distinguish two small classes that are well separated and other clusters where the boundary is not so well defined. The two small clusters correspond to clearly distinguishable lane change signals and the other clusters correspond to various driving patterns and sensor measurements. It is interesting to notice that the distribution of the clusters correspond to what we would expect from real driving data. There are much more scenarios of driving in the lane with various patterns then actual lane changes.

As an additional aid in the manual inspection of the clusters, we can use some of the well known cluster assignment algorithms. We can give these algorithms the two dimensional latent space as input and get out cluster assignments based on various criteria. As an example, in Figure 7.7 d), we applied Density-based spatial clustering (DBSCAN) [109]. This clustering assignment approach makes groups depending on the density of the points. The assigned clusters are: *Green* - lane change right, *Orange* - lane change left, *Blue* - mostly signals that can not be easily assigned or sensor errors, *Red* - driving on the right side of the lane with various patterns, *Purple* - driving on the center or left side of the lane with various patterns. Of course the grouping is not perfect and some overlap between the clusters exist but these are the most frequent signals in each cluster.

To avoid figures with repeating information, the actual signals of the clusters are going to be shown in the next section where we talk about exploring the behavior of classification models. The classification models label the data into three classes: *Lane Change Right*, *Lane Change Left* and driving *In Lane*, so examples for all these cases will be shown.

7.4.2 Behavior Exploration of DL Models

In this section, the latent spaces obtained from the AE and contrastive learning models will be explored. We will highlight signals corresponding to the selected clusters and compare them to classes that come from a lane change classification model.

Figure 7.8 shows the clusters and corresponding signals obtained from the AE and t-SNE model. The figure is divided in three parts. In the middle - Figure 7.8 b), the latent space is shown and the assigned colors correspond to the classes of the classification model. The classes are: *Lane Change Right*, *Lane Change Left* and driving *In Lane*. The first thing we notice, is that the clustering model was able to group together similar signals and that there are several clusters that contain the lane change scenarios.

Let us inspect in detail the clusters corresponding to lane changes. Figure 7.8 a) shows one of the lane change right clusters. On the left side we see the zoomed in latent space and on the right side we see example signals. The first two signals correspond to the two selected points. The third plot shows the comparison of all selected points together. We notice that the cluster indeed contains signals with very similar behavior. The signals are lane change right signals with a small offset between each other.

The offset of the signals comes from the sliding window approach. As we do not know anything about the location of the lane changes in the data, we slice the data into windows sequentially. Therefore, the lane change signals can appear anywhere in the window. We can look at this as an additional feature in the dataset. As we have not given any additional information to the clustering model, it is expected that not all lane changes are going to be grouped together. Some lane changes with different offsets are grouped in other parts of the latent space. By closely inspecting the latent space, we can see several clusters corresponding to lane change right scenarios.

The other thing we notice is that not all signals with the same shape are correctly labeled by the classification model. This information can be used to find out why exactly this is happening. Later, we can correctly label those signals and add them to the training data to improve the classification model.

Similarly to Figure 7.8 a), figure c) shows one cluster corresponding to lane change left scenarios. The conclusion we draw are the same as for figure a). The signals are very similar and have a small offset from each other.

In addition to the two lane change scenarios, in Figure 7.8 b) we have shown some signals from other clusters. The cluster marked with a red triangle correspond to straight driving on the center of the lane. The two other signals show random behavior or sensor errors.

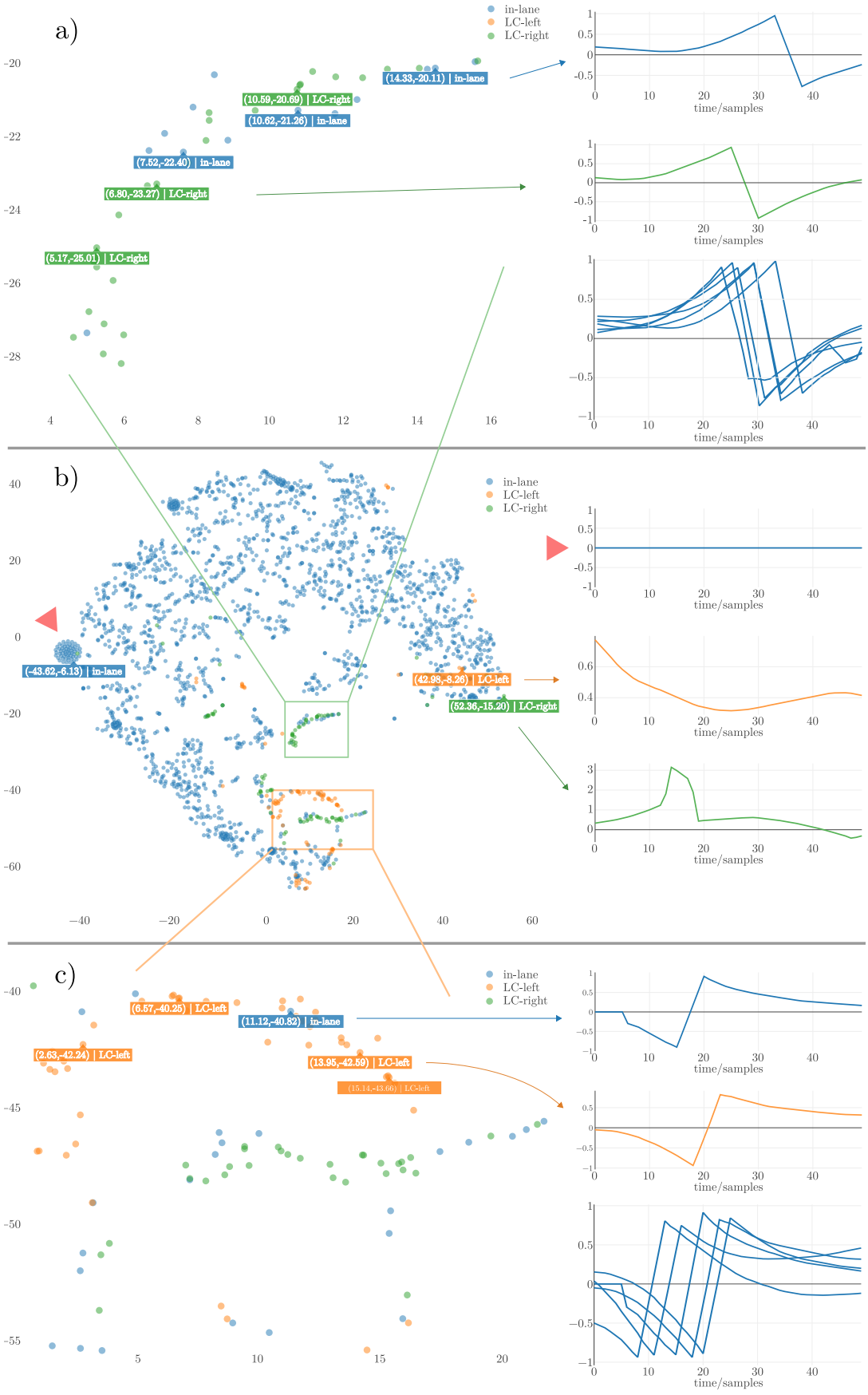

Figure 7.8: *Clustering of the Lateral Deviation signal using AE and t-SNE*

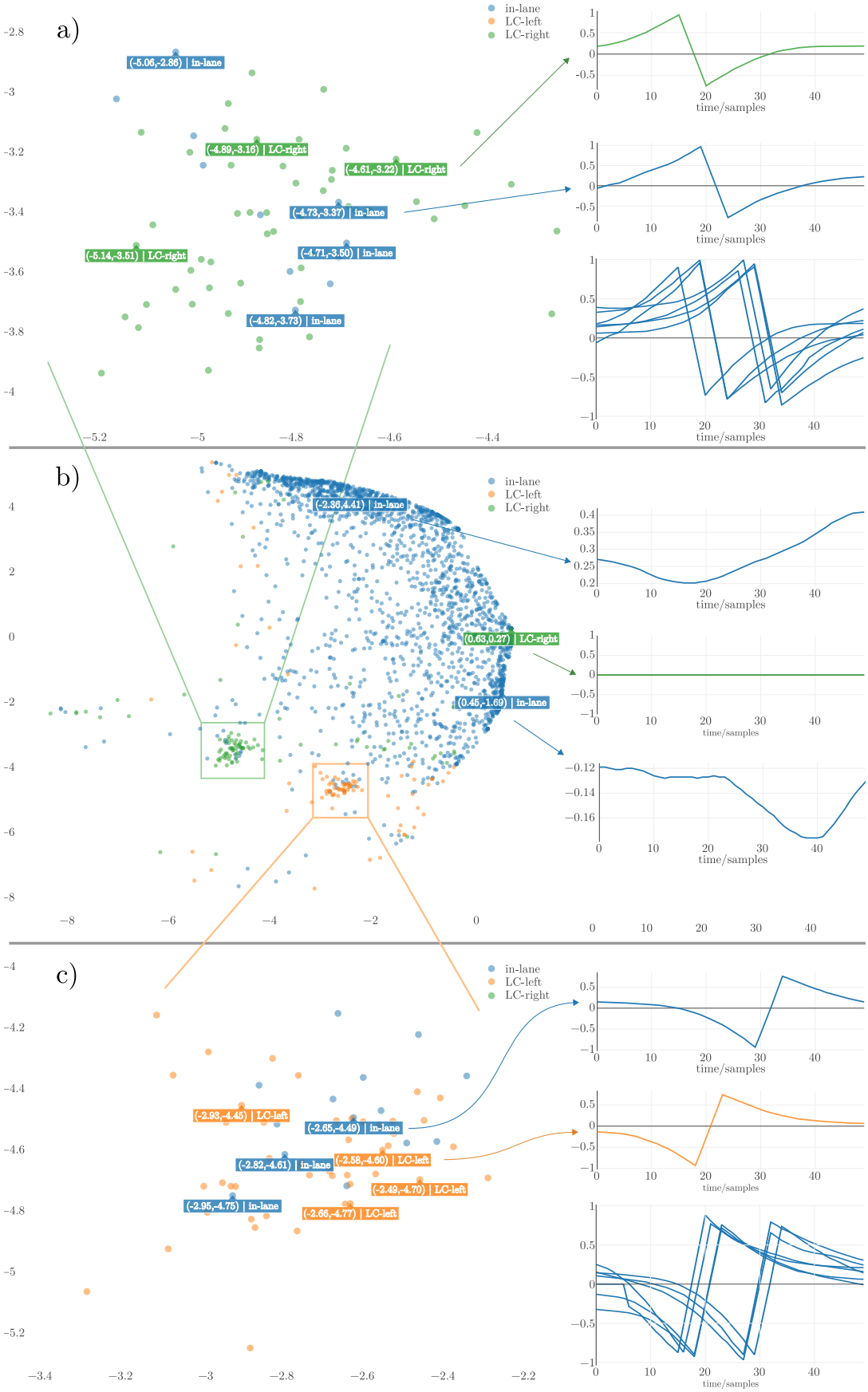

Figure 7.9: *Clustering of the Lateral Deviation signal using Contrastive Learning*

Let us now inspect the contrastive learning clustering. The benefit of the contrastive learning method is that we can specify the augmentation functions and tell the model which signal to pull together in the latent space. In this case, we use an offset augmentation. We sample signals from the data and then create two augmented signals with different random offsets. The model should pull these signals together and learn that we want to have similar signals but with different offsets pulled together.

Figure 7.9 shows the results of the contrastive learning clustering together with the labels from the classification model. The contrastive learning model has done a very good job at creating distinguishable clusters. Similarly to the AE model, the two clusters corresponding to the lane change scenarios are shown in Figure 7.9 a) and c). We notice that our augmentation function has been a good choice and that the lane change signals with different offsets are all grouped together. Again, example signals for the clusters are shown and some misspredictions of the classification model can be recognized.

In addition to these signals, in Figure 7.9 b) we show three random signals corresponding to other clusters. From the top, these tree signals are: random driving on the right side of the lane, driving on the center of the lane and random driving on the left side of the lane.

We can see that the proposed contrastive learning approach has given better results than the AE method. The contrastive learning model uses two times less computation as only the encoder part of the network is needed and the network can map the input data directly to a two dimensional latent space. The drawback of this approach is that we have to manually choose the augmentation functions and their parameters. If, for example, the offset is too small, we would not group together all the similar signals. On the other hand, if the offset is too big, we would mix lane change classes with normal driving. In addition, the contrastive learning approach needs more time for training and convergence.

One way of dealing with the selection of augmentation functions is to treat them as hyper-parameters and train the model with various augmentation functions. At the end, we can choose the one which gives the best results.

7.4.3 Uniqueness estimation

In this section, we will show how the learned latent space can be used to estimate the uniqueness of signals. We are going to use the contrastive learning model and provide five synthetic signals as input. Then, we are going to look at the position in the latent space that each of the synthetic signals has taken and look at its neighbourhood. The results of the analysis are shown in Figure 7.10.

We have chosen the synthetic signals as: S_1 - random driving on the right side of the lane (addition of two sine signals), S_2 - straight driving on the center of the lane, S_3 - moving towards the right lane and back with an unusual pattern, S_4 and S_5 - perfect lane change signals generated with exponential functions.

For each signal, we look at the position and its neighbourhood with a fixed radius $r = 0.5$. The clusters of signals S_1 and S_2 where shown in Figure 7.9. The signals

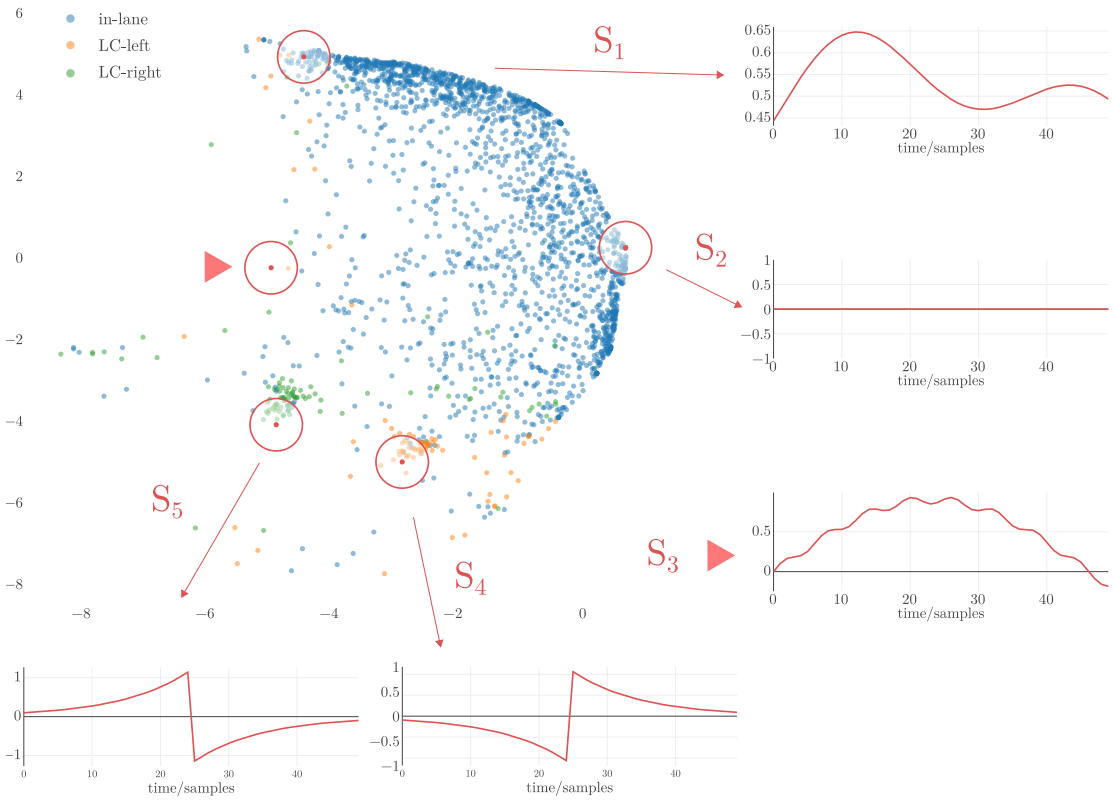

Figure 7.10: *Uniqueness estimation of synthetic signals*

are not unique as they have many similar neighbours. Signals S_4 and S_5 are close to their respective clusters and have also many neighbours. The reason that they are located at the edge of the cluster could be that they are, in general, similar to the lane change signals but there are not so many examples of perfectly smooth and centered lane changes. Finally, signal S_3 is unique as the position is far away from the known clusters and it has only one neighbour.

This method can be useful if we want to filter out well know or already covered scenarios for testing and want to focus on less frequent or unique ones.

7.5 Chapter Conclusion

In this chapter, we have focused on unsupervised clustering of scenarios. We have proposed a clustering approach using contrastive learning and compared it with conventional AE and t-SNE clustering. The contrastive learning approach has shown very good results and we were able to cluster the *Lateral Deviation* dataset into meaningful groups and draw insightful conclusions. In addition, we have examined the behavior of a lane change classification model and have shown how this approach could be used to improve the classification performance by highlighting the instances where the classification model has made errors. Finally, we have shown how the uniqueness of scenarios can be analyzed using the latent space.

8

Conclusion

The focus of this thesis has been testing and validation of ADFs. We started by discussing some of the biggest challenges for their development and introduced four research objectives:

- O1 - Reduction of test runs,
- O2 - Validation using model-based approaches,
- O3 - Scenario classification,
- O4 - Scenario exploration.

The goal of the research was to create a positive feedback loop that makes it possible to decrease the overall testing and validation effort. On one side, we introduced methodologies that required a specific scenario and parameter ranges - O1 and O2. If the scenario and parameter ranges were given, the proposed methods can be used to reduce the number of total test runs or to validate the system on the specified scenario. In O1, we used stochastic optimization and surrogate modeling to create a model of the system's behavior. Then, we tried to steer the parameter exploration to regions where the behavior was not satisfactory. The proposed method can reduce the number of test runs while maximizing the amount of information that can be obtained. In O2, we looked at validation of ADFs when a model of the system is available. We have used a highway scenario as example to tune and validate an ACC. At the end, we were able to define regions where the model was behaving correctly, dangerous test runs occur, as well as the borderline where crashes start.

On the other side, we have conducted research on methodologies that can extract the required scenarios out of recorded data. It has been demonstrated that DL models can be effectively used for scenario classification and exploration - O3 and O4. In O3, we have developed methods to classify scenarios regarding internal sensor measurements and models that can classify scenarios regarding dynamic traffic participants. In both cases, we were able to obtain models that can classify the desired scenarios with high accuracy. Finally in O4, we have shown how DL unsupervised clustering methods can be applied to generate latent space representations that can be used to get a better insight into the data. It has been shown how the clusters can be inspected and how the information can be used to improve existing classification models. In addition, the latent space has been used to estimate the uniqueness of specific scenarios.

When all methodologies are combined in one framework, we get a feedback loop that can help engineer to test and develop ADFs more easily. The developed testing methodologies can be used to lead the classification and exploration towards scenarios where the ADF has undesired behavior, whereas the classification and exploration methodologies can be used to search for similar scenarios from the recorded data and provide them for further testing.

To further improve the proposed framework, additional research on more accurate and capable classification models is needed. More specifically, we need to extend the classification models to include urban and rural driving as well as a much larger variety of scenarios. Similarly, the exploration methodology needs to be extended to handle more complex scenarios with more than one signal and other traffic participants. The results obtained in this thesis pose a good basis for the future improvements.

Glossary

Scenario

A generic traffic situation, i.e., crossing, roundabout, highway driving etc., including variation points for certain environment parameters. For example: Driving on a high way within one lane.

Test Run

Test case with defined values for all parameters. For example: Driving on a two-lane road behind a proceeding vehicle in a distance of 80 m with a defined velocity profile of the proceeding vehicle stepping from 80 to 50 km/h, during light rain and slippery road with defined μ of 0.5. The KPIs is the same as in the test case.

Falsification

A testing procedure with the goal to bring the ADF into a state where it is exhibiting unwanted behavior. Usually, optimization techniques are used in order to lead the ADF to those behaviors.

Ego Vehicle

The vehicle we are currently considering. Either a specific ADF is running on it or it is the reference for the measurement.

Test Case

Traffic scenario with defined discrete parameters – e.g., the number of other traffic participants or the number of lanes + defined KPIs in order to assess the function. Continues variables like starting velocities and positions of the traffic participants are still given as valid ranges only. For example: Driving on a 2-lane road behind a proceeding vehicle in a distance between 80 to 130 m with velocities of the proceeding vehicle between 40 and 150 km/h with defined KPIs like minimum clearance distance happening in the next 5 km of driving.

Highway Pilot

Combination of ADFs like ACC, EBA, and others for a level 4 automation on highways.

Validation

Validation provides assurance that the performance of an ADF meets specific requirements from external customers or stakeholders.

Verification

Verification provides assurance that the performance of an ADF meets regulatory, specification and other internal requirements.

Surrogate Modeling

A technique that is applied when desired values cannot easily be measured so a model is built instead. Usually, this technique is used with optimization where the estimated values of the models are used instead of real measurements.

Stochastic Optimization

A family of optimization methods where some form of randomness is present. This randomness can either come from the objective function, constraints or the optimization algorithm itself.

Global Optimization

A procedure that tries to find the absolute best set of conditions for a given objective, avoiding local minima/maxima.

Model-based Tuning

A procedure where an optimal set of conditions for a given objective is derived using a model built from the real system.

Model-based Validation

A procedure where an assurance of the system's performance is derived using a model

Target ahead switch cut out

A vehicle that was between the ego and another vehicle cuts out to a free lane leading to a switch of the current TA.

Driving in lane

The ego vehicle is driving in a lane and there are no maneuvers from other vehicles.

Lane change

The ego vehicle is switching between free adjacent lanes.

Ego vehicle free lane cut in

The ego vehicle is cutting in from a free to an occupied lane.

Ego vehicle free lane cut out

The ego vehicle is cutting out from an occupied to a free lane.

Ego vehicle cut in-out

The ego vehicle is moving between two occupied lanes, hence making a cut in-out.

Target ahead free lane cut in

A vehicle is cutting in from a free lane to the ego vehicle's lane.

Target ahead free lane cut out

A vehicle is cutting out from ego vehicle's lane to a free lane.

Target ahead switch cut in

A vehicle is cutting in between the ego and another vehicle which leads to the switch of the current TA.

ENABLE S3

European Initiative to Enable Validation for Highly Automated Safe and Secure Systems

Latent space

An abstract representation, learned by a DL model, that encodes a meaningful representation of the input.

Contrastive Learning

A learning approach that tries to pull together latent representations of similar inputs while, at the same time, pushing away dissimilar ones.

Acronyms

AD	Automated Driving
OEM	Original Equipment Manufacturer
ADF	Automated Driving Function
DL	Deep Learning
KPI	Key Performance Indicators
AV	Autonomous Vehicle
V2V	Vehicle to Vehicle
V2X	Vehicle to X
AEB	Automatic Emergency Braking
ML	Machine Learning
SUT	System Under Test
ACC	Adaptive Cruise Control
NN	Neural Network
LKA	Lane Keep Assist
CC	Cruise Control
ESP	Electronic Stability Program
ADAS	Advanced Driver Assistance System
LCA	Lane Change Assist
ABS	Anti-Lock Braking System
SAE	Society of Automotive Engineers
SOTIF	Safety of the Intended Functionality
OSI	Open Simulation Interface
SIL	Software in the Loop
HIL	Hardware in the Loop
VIL	Vehicle in the Loop
ECU	Electronic Control Unit
RBF	Radial Basis Function
DE	Differential Evolution
PSO	Particle Swarm Optimization
EBA	Emergency Brake Assistant
TTC	Time to Collision
MIL	Model in the Loop
GIDAS	German In-Depth Accident Study
MPC	Model Predictive Control

DOE	Design of Experiments
RSI	Relative Significance Indicator
RMSE	Root Mean Square Error
RNN	Robust Neural Network
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
ECG	Electrocardiography
SELU	Scaled Exponential Linear Unit
POM	Polar Occupancy Map
TA	Target Ahead
HPCC	Hierarchical Principal Component Classification
PCA	Principal Component Analysis
SVM	Support Vector Machines
GM	Grid Map
VeG	Velocity Grid
SVeG	Stacked Velocity Grid
HiG	History Grid
VFH	Vector Field Histogram
GNSS	Global Navigation Satellite System
FVeG	Fully Stacked Velocity Grid
AE	Autoencoder
CAE	Convolutional Autoencoder
VAE	Variational Autoencoder
t-SNE	t-distributed Stochastic Neighbor Embedding
SE	Square Error
DBSCAN	Density-based spatial clustering
DAS	Driver Assistance System

List of Figures

List of Tables

Bibliography

[1] M. Alonso Raposo, M. Grosso, J. Després, E. Fernández Macías, C. Galassi, A. Krasenbrink, C. Thiel, An analysis of possible socio-economic effects of a cooperative, connected and automated mobility (ccam) in europe, European Union (2018). (cited on page 1.)

[2] E. R. S. Observatory, Annual accident report 2018 (2018). (cited on page 1.)

[3] C. Wang, The relationship between traffic congestion and road accidents: an econometric approach using gis, Ph.D. thesis (Feb 2010). doi:10.13140/RG.2.1.3180.0087. (cited on page 1.)

[4] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, M. Horn, Search-based optimal motion planning for automated driving, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 4523–4530. doi:10.1109/IROS.2018.8593813. (cited on page 2.)

[5] S. Ishikawa, S. Arai, Cooperative learning of a driving strategy to suppress phantom traffic jams, in: 2016 IEEE International Conference on Agents (ICA), 2016, pp. 90–93. doi:10.1109/ICA.2016.029. (cited on page 2.)

[6] M. Won, T. Park, S. H. Son, Toward mitigating phantom jam using vehicle-to-vehicle communication, IEEE Transactions on Intelligent Transportation Systems 18 (5) (2017) 1313–1324. doi:10.1109/TITS.2016.2605925. (cited on page 2.)

[7] M. Barth, K. Boriboonsomsin, Real-world carbon dioxide impacts of traffic congestion, Transportation Research Record 2058 (1) (2008) 163–171. (cited on page 2.)

[8] J. Lee, B. Park, Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment, IEEE Transactions on Intelligent Transportation Systems 13 (1) (2012) 81–90. doi:10.1109/TITS.2011.2178836. (cited on page 2.)

[9] R. Threlfall, Autonomous vehicles readiness index, Klynveld Peat Marwick Goerdeler (KPMG) International (2018). (cited on page 2.)

[10] Z. Ajanović, M. Stolz, M. Horn, Energy-efficient driving in dynamic environment: Globally optimal mpc-like motion planning framework, in: C. Zachäus, B. Müller, G. Meyer (Eds.), Advanced Microsystems for Automotive Applications 2017, Springer International Publishing, Cham, 2018, pp. 111–122. (cited on page 2.)

[11] S. Tsugawa, S. Kato, K. Aoki, An automated truck platoon for energy saving, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 4109–4114. doi:10.1109/IROS.2011.6094549. (cited on page 2.)

[12] H. Beglerovic, M. Stolz, M. Horn, Testing of autonomous vehicles using surrogate models and stochastic optimization, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 1–6. doi:10.1109/ITSC.2017.8317768. (cited on page 5.)

[13] H. Beglerovic, A. Ravi, N. Wikström, H.-M. Koegeler, A. Leitner, J. Holzinger, Model-based safety validation of the automated driving function highway pilot, in: 8th International Munich Chassis Symposium 2017, Springer, 2017, pp. 309–329. (cited on page 5.)

[14] H. Beglerovic, T. Schloemicher, S. Metzner, M. Horn, Deep learning applied to scenario classification for lane-keep-assist systems, Applied Sciences 8 (12) (2018) 2590. doi:10.3390/app8122590. (cited on pages 6 and 61.)

[15] H. Beglerovic, J. Ruebsam, S. Metzner, M. Horn, Polar occupancy map - a compact traffic representation for deep learning scenario classification, in: 2019 IEEE 22nd International Conference on Intelligent Transportation Systems (ITSC), 2019. (cited on page 6.)

[16] H. Beglerovic, S. Metzner, M. Horn, Challenges for the validation and testing of automated driving functions, in: C. Zachäus, B. Müller, G. Meyer (Eds.), Advanced Microsystems for Automotive Applications 2017, Springer International Publishing, Cham, 2018, pp. 179–187. (cited on page 6.)

[17] I. O. for Standardization, Iso/pas 21448, "road vehicles—safety of the intended functionality" (2019). (cited on page 11.)

[18] D. Watzenig, M. Horn, Automated Driving: Safer and More Efficient Future Driving, Springer, 2016. (cited on page 11.)

[19] T. Hanke, N. Hirsenkorn, C. van Driesten, P. Garcia Ramos, M. Schiementz, S. Schneider, Open Simulation Interface: A generic interface for the environment perception of automated driving functions in virtual scenarios., Accessed: 2017-08-28 (2017).
URL http://www.hot.ei.tum.de/forschung/automotive-veroeffentlichungen/
(cited on page 12.)

[20] H. Winner, W. Wachenfeld, Absicherung automatischen fahrens, 6, FAS-Tagung München, Munich 9 (2013). (cited on page 12.)

[21] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, Carla: An open urban driving simulator, arXiv preprint arXiv:1711.03938 (2017). (cited on page 13.)

[22] S. Shah, D. Dey, C. Lovett, A. Kapoor, Airsim: High-fidelity visual and physical simulation for autonomous vehicles, in: Field and service robotics, Springer, 2018, pp. 621–635. (cited on page 13.)

[23] D. Krajzewicz, M. Bonert, P. Wagner, The open source traffic simulation package sumo, RoboCup 2006 (2006). (cited on page 13.)

[24] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The kitti dataset, The International Journal of Robotics Research 32 (11) (2013) 1231–1237. (cited on page 14.)

[25] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, T. Darrell, Bdd100k: A diverse driving video database with scalable annotation tooling (2018). `arXiv:1805.04687`. (cited on page 14.)

[26] R. Krajewski, J. Bock, L. Kloeker, L. Eckstein, The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2118–2125. `doi:10.1109/ITSC.2018.8569552`. (cited on pages 14 and 75.)

[27] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, M. Tomizuka, INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps, arXiv:1910.03088 [cs, eess] (2019). (cited on page 14.)

[28] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, D. Anguelov, Scalability in perception for autonomous driving: An open dataset benchmark (2019). `arXiv:1912.04838`. (cited on page 14.)

[29] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, V. Shet, Lyft level 5 av dataset 2019, urlhttps://level5.lyft.com/dataset/ (2019). (cited on page 14.)

[30] H. Schafer, E. Santana, A. Haden, R. Biasini, A commute in data: The comma2k19 dataset (2018). `arXiv:arXiv:1812.05752`. (cited on page 14.)

[31] A. Gruber, M. Gadringer, H. Schreiber, D. Amschl, W. Bösch, S. Metzner, H. Pflügl, Highly scalable radar target simulator for autonomous driving test beds, in: 2017 European Radar Conference (EURAD), IEEE, 2017, pp. 147–150. (cited on page 15.)

[32] I. ISO, 26262: Road vehicles-functional safety, International Standard ISO/FDIS 26262 (2011). (cited on page 15.)

[33] M. Dupuis, M. Strobl, H. Grezlikowski, Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks, in: Proc. of the Driving Simulation Conference Europe, 2010, pp. 231–242. (cited on page 16.)

[34] J. Rauh, Opencrg–the new open standard to represent high precision 3d road data in vehicle simulation tasks on rough roads for handling, ride comfort, and durability load analyses, in: Proceedings of the 21st International Symposium Dynamics of Vehicles on Roads and Tracks IAVSD, Vol. 9, 2009, pp. 17–21. (cited on page 16.)

[35] A. for Standardisation of Automation, M. Systems, Openscenario., https://www.asam.net/standards/detail/openscenario/ (2019). (cited on page 16.)

[36] J. E. Stellet, M. R. Zofka, J. Schumacher, T. Schamm, F. Niewels, J. M. Zöllner, Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions, in: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 2015, pp. 1455–1462. doi:10.1109/ITSC.2015.236. (cited on page 18.)

[37] W. Huang, K. Wang, Y. Lv, F. Zhu, Autonomous vehicles testing methods review, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 163–168. doi:10.1109/ITSC.2016.7795548. (cited on page 18.)

[38] M. R. Zofka, S. Klemm, F. Kuhnt, T. Schamm, J. M. Zöllner, Testing and validating high level components for automated driving: simulation framework for traffic scenarios, in: 2016 IEEE Intelligent Vehicles Symposium (IV), 2016, pp. 144–150. doi:10.1109/IVS.2016.7535378. (cited on page 18.)

[39] M. R. Zofka, R. Kohlhaas, T. Schamm, J. M. Zöllner, Semivirtual simulations for the evaluation of vision-based adas, in: 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014, pp. 121–126. doi:10.1109/IVS.2014.6856593. (cited on page 18.)

[40] M. R. Zofka, F. Kuhnt, R. Kohlhaas, C. Rist, T. Schamm, J. M. Zöllner, Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems, in: 2015 18th International Conference on Information Fusion (Fusion), 2015, pp. 1422–1428. (cited on page 18.)

[41] I. B. Jemaa, D. Gruyer, S. Glaser, Distributed simulation platform for cooperative adas testing and validation, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 77–82. doi:10.1109/ITSC.2016.7795535. (cited on page 18.)

[42] M. Sieber, B. Färber, Driver perception and reaction in collision avoidance: Implications for adas development and testing, in: 2016 IEEE Intelligent Vehicles Symposium (IV), 2016, pp. 239–245. doi:10.1109/IVS.2016.7535392. (cited on page 18.)

[43] C. E. Tuncali, T. P. Pavlic, G. Fainekos, Utilizing s-taliro as an automatic test generation framework for autonomous vehicles, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 1470–1475. doi:10.1109/ITSC.2016.7795751. (cited on pages 18 and 20.)

[44] Y. Annpureddy, C. Liu, G. Fainekos, S. Sankaranarayanan, S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-19835-9_21. (cited on page 18.)

[45] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, K. Butts, Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques, IEEE Control Systems 36 (6) (2016) 45–64. doi:10.1109/MCS.2016.2602089. (cited on pages 18 and 19.)

[46] A. Donzé, Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. doi:10.1007/978-3-642-14295-6_17. (cited on page 18.)

[47] R. Ben Abdessalem, S. Nejati, L. C. Briand, T. Stifter, Testing advanced driver assistance systems using multi-objective search and neural networks, in: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, ACM, New York, NY, USA, 2016, pp. 63–74. doi:10.1145/2970276.2970311. (cited on pages 19 and 20.)

[48] A. Forrester, A. Sobester, A. Keane, Engineering design via surrogate modelling: a practical guide, John Wiley & Sons, 2008. (cited on pages 22 and 23.)

[49] H. H. Hoos, T. Stützle, Stochastic local search: Foundations and applications, Elsevier, 2004. (cited on page 23.)

[50] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359. (cited on page 23.)

[51] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm intelligence 1 (1) (2007) 33–57. (cited on pages 23 and 27.)

[52] MATLAB, version 9.1.0.4 (R2016b), The MathWorks Inc., Natick, Massachusetts, 2016. (cited on page 25.)

[53] G. D. Nicolao, A. Ferrara, L. Giacomini, Onboard sensor-based collision risk assessment to improve pedestrians' safety, IEEE Transactions on Vehicular Technology 56 (5) (2007) 2405–2413. doi:10.1109/TVT.2007.899209. (cited on page 25.)

[54] A. Ferrara, C. Vecchio, Collision avoidance strategies and coordinated control of passenger vehicles, Nonlinear Dynamics 49 (4) (2007) 475–492. doi:10.1007/s11071-006-9110-4. (cited on page 25.)

[55] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31. doi:10.1109/TEVC.2010.2059031. (cited on page 27.)

[56] BASt, Bundesanstalt für Straßenwesen, https://www.bast.de (2019). (cited on pages 31 and 39.)

[57] A. Engstle, H. Assmayr, G. Schwab, Softwareentwicklung für autonome fahrfunktionen, ATZelektronik 12 (1) (2017) 42–47. (cited on page 31.)

[58] SAE, Society of Automotive Engineers, https://www.sae.org (2019). (cited on page 31.)

[59] N. Keuth, H.-M. Kögeler, T. Fortuna, G. Vitale, Doe and beyond: The evolution of the model based development approach how legal trends are changing methodology, Design of Experiments (DoE) in Powertrain Development (2015). (cited on pages 36 and 38.)

[60] A. Rainer, H.-M. Koegeler, Iterative doe-improved emission models and better optimisation results within a shortened measurement time, International Journal of Powertrains 6 (2) (2017) 109–124. (cited on page 37.)

[61] H.-M. Koegeler, B. Schick, P. Pfeffer, A. Contini, M. Lugert, T. Schöning, Model-based steering ecu calibration on a steering-in-the-loop test bench, in: 6th International Munich Chassis Symposium 2015, Springer, 2015, pp. 455–466. (cited on page 37.)

[62] Safe distance between vehicles (2010). (cited on page 41.)

[63] J. Holzinger, T. Schloemicher, E. Bogner, P. Schoeggl, Objective assessment of comfort and safety of advanced driver assistance systems, in: FISITA World Automotive Congress, 2016. (cited on page 45.)

[64] Avl-drive® adas., www.avl.com/-/avl-drive-4- (2020). (cited on page 45.)

[65] M. Hermans, B. Schrauwen, Training and analysing deep recurrent neural networks, in: Advances in neural information processing systems, 2013, pp. 190–198. (cited on page 46.)

[66] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112. (cited on page 46.)

[67] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014). (cited on page 46.)

[68] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780. (cited on page 46.)

[69] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014). (cited on page 46.)

[70] P. Malhotra, V. TV, L. Vig, P. Agarwal, G. Shroff, Timenet: Pre-trained deep recurrent neural network for time series classification, arXiv preprint arXiv:1706.08838 (2017). (cited on page 46.)

[71] R. Salloum, C.-C. J. Kuo, Ecg-based biometrics using recurrent neural networks, in: Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on, IEEE, 2017, pp. 2062–2066. (cited on page 46.)

[72] A. Borovykh, S. Bohte, C. W. Oosterlee, Conditional time series forecasting with convolutional neural networks, arXiv preprint arXiv:1703.04691 (2017). (cited on page 46.)

[73] D. Yao, C. Zhang, Z. Zhu, J. Huang, J. Bi, Trajectory clustering via deep representation learning, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 3880–3887. doi:10.1109/IJCNN.2017.7966345. (cited on page 46.)

[74] F. J. Ordóñez, D. Roggen, Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition, Sensors 16 (1) (2016) 115. (cited on page 46.)

[75] Y. Zhang, M. J. Er, R. Venkatesan, N. Wang, M. Pratama, Sentiment classification using comprehensive attention recurrent models, in: Neural Networks (IJCNN), 2016 International Joint Conference on, IEEE, 2016, pp. 1562–1569. (cited on page 46.)

[76] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 1578–1585. doi:10.1109/IJCNN.2017.7966039. (cited on page 47.)

[77] S. Bai, J. Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, arXiv preprint arXiv:1803.01271 (2018). (cited on page 47.)

[78] R. Gruner, P. Henzler, G. Hinz, C. Eckstein, A. Knoll, Spatiotemporal representation of driving scenarios and classification using neural networks, in: Intelligent Vehicles Symposium (IV), 2017 IEEE, IEEE, 2017, pp. 1782–1788. (cited on pages 47, 61 and 69.)

[79] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017). (cited on page 48.)

[80] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation, arXiv preprint arXiv:1801.04381 (2018). (cited on page 48.)

[81] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, arXiv preprint arXiv:1511.07122 (2015). (cited on pages 48 and 69.)

[82] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. (cited on pages 49 and 69.)

[83] A. Veit, M. J. Wilber, S. Belongie, Residual networks behave like ensembles of relatively shallow networks, in: Advances in Neural Information Processing Systems, 2016, pp. 550–558. (cited on pages 49 and 69.)

[84] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167 (2015). (cited on pages 50 and 69.)

[85] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in: Advances in Neural Information Processing Systems, 2017, pp. 972–981. (cited on pages 50 and 69.)

[86] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958. (cited on page 51.)

[87] H. Elrofai, D. Worm, O. O. den Camp, Scenario identification for validation of automated driving functions, in: Advanced Microsystems for Automotive Applications 2016, Springer, 2016, pp. 153–163. (cited on page 60.)

[88] R. Kastner, F. Schneider, T. Michalke, J. Fritsch, C. Goerick, Image-based classification of driving scenes by hierarchical principal component classification (hpcc), in: Intelligent Vehicles Symposium, 2009 IEEE, IEEE, 2009, pp. 341–346. (cited on page 61.)

[89] N. Bernini, M. Bertozzi, L. Devincenzi, L. Mazzei, Comparison of three approaches for scenario classification for the automotive field, in: International Conference on Image Analysis and Processing, Springer, 2013, pp. 582–591. (cited on page 61.)

[90] C. Roesener, F. Fahrenkrog, A. Uhlig, L. Eckstein, A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour, in: Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on, IEEE, 2016, pp. 1360–1365. (cited on page 61.)

[91] I. Cara, E. de Gelder, Classification for safety-critical car-cyclist scenarios using machine learning, in: Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on, IEEE, 2015, pp. 1995–2000. (cited on page 61.)

[92] J. Borenstein, Y. Koren, The vector field histogram-fast obstacle avoidance for mobile robots, IEEE transactions on robotics and automation 7 (3) (1991) 278–288. (cited on page 62.)

[93] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural computation 1 (4) (1989) 541–551. (cited on pages 64 and 69.)

[94] S. Lee, I. S. Kweon, J. Kim, J. S. Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. S. Hong, S.-H. Han, Vpgnet: Vanishing point guided network for lane and road marking detection and recognition, in: 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 1965–1973. (cited on page 66.)

[95] H. Beglerovic, Data, network models and code used for the pom representation research., `https://gitlab.com/hal3e/polar-occupancy-map` (2019). (cited on pages 69 and 72.)

[96] Mobileye, `https://www.mobileye.com` (2019). (cited on pages 70 and 81.)

[97] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034. (cited on page 71.)

[98] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014). (cited on pages 71 and 81.)

[99] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning internal representations by error propagation, Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science (1985). (cited on pages 75 and 76.)

[100] J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, in: International conference on artificial neural networks, Springer, 2011, pp. 52–59. (cited on pages 75 and 77.)

[101] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, arXiv preprint arXiv:2002.05709 (2020). (cited on pages 75 and 79.)

[102] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, D. Krishnan, Supervised contrastive learning, arXiv preprint arXiv:2004.11362 (2020). (cited on pages 75 and 79.)

[103] K. Sama, Y. Morales, N. Akai, H. Liu, E. Takeuchi, K. Takeda, Driving feature extraction and behavior classification using an autoencoder to reproduce the velocity styles of experts, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2018, pp. 1337–1343. (cited on page 75.)

[104] R. Krajewski, T. Moers, A. Meister, L. Eckstein, Béziervae: Improved trajectory modeling using variational autoencoders for the safety validation of highly automated vehicles, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE, 2019, pp. 3788–3795. (cited on page 75.)

[105] C. Doersch, Tutorial on variational autoencoders, arXiv preprint arXiv:1606.05908 (2016). (cited on page 75.)

[106] J. Langner, J. Bach, L. Ries, S. Otten, M. Holzäpfel, E. Sax, Estimating the uniqueness of test scenarios derived from recorded real-world-driving-data using autoencoders, in: 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2018, pp. 1860–1866. (cited on page 76.)

[107] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of machine learning research 9 (Nov) (2008) 2579–2605. (cited on page 78.)

[108] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 315–323. (cited on page 80.)

[109] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: Kdd, Vol. 96, 1996, pp. 226–231. (cited on pages 82, 83 and 101.)