CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Validation and Verification Challenges in a Machine Learning Algorithm for Connected Vehicles

Design Science Research of Developing a Most Probable Path Algorithm

Master's thesis in Computer science and engineering

Axel Hertzberg
Erik Bengtsson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Validation and Verification Challenges in a Machine Learning Algorithm for Connected Vehicles

Design Science Research of Developing a Most Probable Path Algorithm

Axel Hertzberg
Erik Bengtsson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Validation and Verification Challenges in a Machine Learning Algorithm for Connected Vehicles
Design Science Research of Developing a Most Probable Path Algorithm
Axel Hertzberg & Erik Bengtsson

Supervisor: Robert Feldt, Computer Science and Engineering
Advisor: Philip Andreasson, CarmentaAutomotive
Examiner: Christian Berger, Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Validation and Verification Challenges in a Machine Learning Algorithm for Connected Vehicles
Design Science Research of Developing a Most Probable Path Algorithm
Axel Hertzberg & Erik Bengtsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Machine Learning (ML) software in connected and automated vehicles puts new demands on safety regulators and industry standards to keep up with the explosive evolution of technology in the automotive domain. This thesis reports a practical example of developing an ML-based algorithm that predicts the most probable path for an arbitrary vehicle, without knowing the destination. This work is done in collaboration with Carmenta Automotive AB as an industry partner, a company that is aiming to increase situational awareness for vehicles on the roads. The thesis methodology follows an iterative design science research (DSR) approach, developing an artifact consisting of an ML model connected to the company's system. The literature highlights the challenges of validating and verifying (V&V) an ML component, as there are currently no applicable standards for ML software in the automotive domain. This DSR attempts to showcase V&V activities on ML models trained with different data characteristics to assess whether the challenges surrounding V&V can be mitigated when validating the data-driven most probable path algorithm.

Keywords: Computer, science, computer, project, thesis, machine learning, automotive, connected vehicles, validation and verification.

Acknowledgements

We extend our sincerest appreciation to our supervisor, Robert Feldt, for his valuable guidance and support throughout this thesis.
A special acknowledgment goes to our company advisor, Philip Andreasson, for his support and technical expertise from Carmenta Automotive AB. We are also deeply thankful to Carmenta Automotive AB for granting us the opportunity to conduct our research and for generously providing resources essential to our work.

Axel Hertzberg & Erik Bengtsson, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AD	Autonomous drive
ADAS	Advanced Driver-assistance Systems
AI	Artificial Intelligence
AV	Automated Vehicle
CV	Connected Vehicle
DSR	Design Science Research
ML	Machine Learning
MPP	Most Probable Path
POC	Proof of Concept
RQ	Research Question
SA	Situational Awareness
UI	User Interface
V&V	Validation and Verification

Contents

List of Figures

List of Tables

1

Introduction

The use of software in vehicles has been steadily increasing over time, and there is no sign of this trend slowing down. Advancements in Artificial Intelligence (AI) and Machine Learning (ML) are pushing the boundaries of what's possible, with connected and autonomous vehicles being prime examples. This progress is putting new demands on safety regulators and industry standards to keep up with the explosive evolution of technology in the automotive domain.

The growing volume of data in modern connected vehicles (CV) plays a crucial role in facilitating ML components and Advanced Driver Assistance Systems (ADAS). These technologies have been leveraged for various services, including emergency braking, lane detection, and cruise control. Software development for vehicles on roads is often conducted following guidelines from an ISO standard, specifically ISO 26262. This standard serves as an international standard for traditional road vehicle software development. ISO 26262 includes steps on how to validate and verify the software that you are developing to ensure its correct functioning. This is crucial in a safety-critical context in which road vehicles are included. However, its direct applicability to ML development has been questioned by several sources [1]–[4].

In modern vehicles equipped with software, systems are often integrated to provide warnings about upcoming situations. The goal is to reduce the number of accidents and influence the behavior of drivers on the road [5]. When these systems perform correctly, informative warnings enhance Situation Awareness (SA) for the driver, a concept aimed at keeping the driver informed about traffic situations. Simultaneously, warnings from SA systems must be accurate and presented well in advance to allow the driver to adapt to the situation. Suppose the SA system is aware of the driver's intended destination. In that case, a search algorithm can calculate whether the vehicle's path will intersect with the situation and provide an accurate warning. An easy way to determine the intended destination for the driver is when the driver specifies it on a map service, enabling both the driver and the system to know the destination. This can be advantageous for SA systems as it facilitates informing drivers of situations ahead. However, drivers are not currently obliged to specify their end destination, often leaving the destination unknown. To still be able to inform the driver of situations ahead, one can predict the upcoming route for the vehicle, which can be achieved with a Most Probable Path (MPP) algorithm.

1.1 Statement of the Problem

One main objective of the system is to provide connected vehicles with information about the surroundings beyond the vehicle's line of sight to increase SA, which is made by aggregating several data sources. To be able to provide information on the vehicle's surroundings, one is interested in its future positions. If the system is aware of the destination, it is fairly straightforward to predict the route to this point. However, the system is often unaware of the destination and it is then possible to predict future positions and routes using an MPP algorithm. The idea with the MPP algorithm is to use it to send precise and timely warnings to the driving system and increase the quality of SA. Currently, the company considers the current implemented MPP algorithm too "general" in its output and aims to improve the predictions by implementing a more data-driven solution with ML to make it more realistic.

Since cloud-based services operate in a safety-critical context, integrated software components need to be both robust and thoroughly verified and validated. However, the published papers often relate to the ISO 26262 and its lack of applicability to ML components. These papers often lack concrete activities and practical examples for V&V. Although ML software is increasingly used in the automotive domain, it poses safety concerns due to the absence of a unified approach to V&V ML software.

1.2 The Purpose of the Study

The purpose of this study is to implement an ML component to predict the MPP for an arbitrary CV, in cases where the destination is unknown. The ML component will be a supervised trained model that will be used for delivering safety-critical information to vehicles that goes towards the MPP output. The algorithm will be an integrated building block of software functioning in a safety-critical domain. The aim is to deliver an MPP algorithm to the company that increases the systems' possibility to provide timely and accurate information to vehicles and improve the overall SA on roads.

Since there is a gap in the published literature on concrete V&V of ML components in the automotive domain, the primary focus of the research will not be on the ML model itself, but rather on a more distinct emphasis on the V&V processes and activities to establish methods for ensuring the quality of the ML component. The implemented MPP algorithm will function as a software component that will be evaluated based on these processes to ensure that it meets requirements and expected behavior.

The study aims to provide V&V processes applicable to other areas when developing ML software for the automotive domain. This includes other safety-critical applications, involving practitioners in the industry, as well as researchers in Software Engineering who focus on ML software for vehicles on roads.

1.3 Research Questions and Thesis Outline

The thesis aims to answer three research questions and each one has a distinct focus. RQ1 refers to the problem statement, RQ2 the current state of the art in addressing the problems, and the third puts possible solutions in a practical setting.

RQ1: What challenges exist when validating and verifying supervised predictive Machine Learning models used in connected vehicles in a safety-critical context?

This RQ focuses on investigating the current problems in depth and understanding the difficulties encountered during the validation and verification process of ML software.

RQ2: What methodologies and techniques exist to mitigate the challenges of validating and verifying supervised predictive Machine Learning models used in connected vehicles to enable accurate safety-critical information delivery?

The second RQ builds upon the first question, aiming to explore potential solutions to the identified challenges. After delving into the challenges and current methodologies for developing algorithms for connected vehicles the thesis will specifically address how the MPP algorithm can be verified and validated in a practical, real-world setting. This is the distinct focus of the last RQ:

RQ3: How can the performance of an ML component predicting the Most Probable Path be verified and validated in the context of delivering safety-critical information?

The outline of this thesis, designed to address the research questions, is as follows: Chapter 2 includes relevant background information and previous studies. Chapter 3 explains the chosen methodology, including the reasoning for selecting this method and how it aligns with the research questions, as well as the data collection process. Chapter 4 presents the results, including the literature review and the presentation of findings in the form of text, figures, and tables, some of which are linked to the appendix. In Chapter 5, the thesis discusses each research question and summarizes the findings based on these questions, including a section on threats to validity. Chapter 6 concludes with the key takeaways from this thesis.

2
Background

This chapter aims to establish a foundational understanding of the key topics addressed in this thesis. It begins by reviewing relevant literature and then proceeds to define the most significant concepts.

2.1 Related Works

Carmenta Automotive has previously undertaken thesis projects within the field of MPP. In their work, Arvidsson and Hendén [6] concentrated on creating an algorithm to predict emergency vehicle routes during emergencies. Their objective was to share awareness information with drivers along the emergency vehicle's projected path. Wiklund and Rosenberg conducted a continuation of that work [7] to validate and optimize the performance of the algorithm. In the problem statement for this algorithm, the destination was known.

Several approaches to path prediction, encompassing both short and long-term predictions, are documented in the literature. Jeung et al. [8] propose an approach that extends predictions beyond the nearest junction by leveraging historical trajectories and real-time velocity data. This method enables accurate route predictions beyond immediate movements by constructing a probabilistic model that evaluates possible routes based on past frequency and current traffic conditions. While this approach significantly improves prediction accuracy compared to traditional methods, it is limited by the availability and quality of historical data, which can be problematic in areas with sparse data. Combining real-time and historical data was proven to improve the predictive accuracy computational complexity increased significantly compared to traditional, simpler approaches to path prediction. In a related study, Krumm et al. [9] develop an algorithm for route prediction that in contrast to Jeung et al.'s work, is independent from historical data. The algorithm instead plans routes to multiple candidate destinations and assigns higher probabilities to roads leading to more likely destinations. This method is advantageous because it does not require storing extensive historical data for each trip. Instead, it calculates a single parameter characterizing driving efficiency, allowing the algorithm to work effectively even in unfamiliar areas.

Several authors have previously published papers on the challenges of V&V within the automotive domain. Koopman and Wagner [1] identify five challenge areas concerning the testing and validation of Autonomous Vehicles. These areas include requirements, fail-operating systems, and methods that can handle the non-

deterministic nature of many ML models. The authors propose that one solution to address challenges is systematically introducing AVs into increasingly more complex scenarios, ensuring safety at each stage. Knauss et al. [2] investigate software-related challenges for autonomous vehicle testing through focus groups and interviews. In this study, "virtual testing and simulations" and "scenario complexity and many test cases" were frequently mentioned challenges among the participants. Tambon et al. [10] focus on ML in their systematic literature review. They investigate verification and certification challenges specifically for safety-critical ML-based systems and conclude that no ML certification currently exists. However, they believe that by bringing together several sub-fields, ISO 26262 can be used as a basis and adapted to serve ML purposes better.

2.2 Connected Vehicles

The term CV, as defined by Uhlemann [11], refers to vehicles that communicate with external services, such as cloud services, accessible on demand at any time. CV enables creative innovations in the automotive industry. The company's software product establishes communication with CV via the cloud, facilitating in-vehicle warnings such as updates on road conditions and accidents.

Today, different sensors are commonly used and combined to enable various features and functionalities in vehicles, such as lane detection, emergency braking, and cruise control. SA is related to sensors and sensor sight and can according to Endsley [12] be defined as recognizing, understanding, and foreseeing the state and status of elements in given environments over time. SA in the context of a traffic system is explained by Golestan et al. [13] as a goal to reach in a safety application for a CV. A SA system is a data pool with information about a vehicle's surroundings that needs to be processed.

2.3 Machine Learning

There exist different definitions of ML in the literature, but a common definition from Mohri et al. [14] is that ML is a collection of computational methods based on available past information to improve performance to make accurate predictions. The past information is often in the form of collected data in a digital format, which this thesis will refer to as training data. A model consists of a learning algorithm that aims to find patterns in the training data and the choice of a learning algorithm is made based on the nature of the problem.

To evaluate the performance and accuracy of an ML component and its predictions, different metrics are calculated by following a specified procedure. An ML component should perform well on unseen data, i.e. data that is not included when building the model with training data. The step when investigating the model performance is often called Model Evaluation. Raschka [15] states that model evaluation is an experiment of changing parameters to the learning algorithm, which results in different models. The evaluation step includes choosing the best-performing model which

is not a trivial task.

Some different procedures and measurements can be calculated to assess how well the ML model performs. One method that Raschka [15] mentions is the holdout method, where the model is trained with a training data set and validated using a validation data set. These data sets are independent, and the validation data set serves as the ground truth with the goal that the model will predict as similar to the validation set. The holdout methodology measures either the number of correct results if it is a classification problem, and if it is a regression problem then the measure is a numeric value that shows how much total error the model makes compared to the validation set.

The scores of machine learning models are not always sufficiently informative and may be sensitive due to the training data on which the model is built. One approach to demonstrate the model's performance is by calculating confidence intervals, as outlined by Raschka [15], which represent an interval between set percentage values. With confidence intervals, you can visualize the model's performance on the validation set and disregard predictive values that fall below the specified percentage threshold.

2.4 Validation, Verification and Simulation Systems

Validation and verification (V&V) is a procedure to ensure that the software achieves the expected behavior. Validation can be understood as the process of gathering evidence to ensure with a high degree of confidence, that a particular process will produce a result that meets its predetermined specification. Verification means checking and providing proof that certain requirements for a product or process have been satisfied [16].

There is no universally defined process for Verification and Validation (V&V). Various methods are proposed in the literature for different types of software. Sargent [17] discusses the V&V process by outlying three decision-making processes for assessing the validity of a simulation system. The most common method involves the development team making subjective decisions based on test results and model evaluations. The second method is known as independent V&V, which involves a third party determining the validity of the model. The third method mentioned by Sargent [17] is a score-based approach, where scores are calculated from the model and combined into an overall measurable value that must meet a certain threshold to be considered valid.

Software development in the automotive sector belongs to a safety-critical domain. This means that careful V&V of software functionality is required, where ISO 26262 is an international standard for functional safety of road vehicles [3]. The standard has 10 sections and the 6th part is related to product development on a software level. Here, the V-model serves as the foundation for how the V&V process should be conducted. The V-model shows the relation between the design- and testing phases, and the idea is that by following these practices a computer system could be

considered safe [1]. Several studies have explored ISO 26262 and its implications for ML development. Henriksson et al. [18] emphasize the challenge posed by the black box nature of ML models in aligning with the software-related practices advocated by the ISO standard, such as code reviews and code coverage metrics. Based on the research of Salay et al. [3], the authors propose modifications to certain standard elements to enhance its compatibility with ML development, as evidenced by expert interviews. Despite this, little to no information has been published on practical examples exploring which specific methods are useful for verifying and validating an ML model within the automotive industry.

2.5 Geographical Information Systems

A geographical information system (GIS) is a digital application that can be used for various purposes, such as storing or displaying geographical information [19]. In GIS systems, networks are often referred to, which consider the topology, connectivity, path choices, or flow characteristics. An almost universal representation of networks is a set of nodes and links, where the nodes represent points in space and time, and a link corresponds to identifiable pieces of transportation infrastructure such as roads or railways. The links can be either directed or undirected. In a directed link, you travel from one node to another in a specified direction, while in an undirected link, no specific direction of travel is indicated [19].

A subset focus within GIS systems is graph theory, where transportation networks are represented as valued graphs. In this context, a path is considered as a sequence of distinct nodes connected with directed links between them [19]. The links can consist of different properties and information, such as street names, speed limits, and estimated flow. In Lin and Ban's article [20], they illustrate an urban road network with nodes and links, allowing for different methodologies for shortest path search, road capacity, and more. This is accomplished by introducing link weights between the nodes, which can represent different things such as measurements of traffic flow, travel time, and geographical distance.

3

Methodology

This chapter aims to provide an in-depth understanding of the Design Science methodology and its application throughout this thesis. The chapter later describes the literature review procedure and defines its scope. It finishes with terminology and context-specific definitions.

3.1 Design Science Research

The study followed a Design Science Research (DSR) approach which is a research method that utilizes academic insights and practices in a business or company context. A DSR aims to develop and create an artifact that can solve real-world problems in a specific context. DSR is a methodology where the artifact is focused, developed, and evaluated to solve a problem with a scientific understanding. According to Knauss [21], it is suitable in cases where the goal is to address real-world problems that are hard to solve using traditional research methods. Hevner [22] also argues that DSR is a methodology intended to solve technical and organizational problems, which is what this study aimed to do.

Since this study addresses a real-world problem in a natural organizational setting, it can be related to the field experiment discussed in Stol and Fitzgerald's [23] paper concerning software engineering research. They state that a field experiment is considered a natural setting where generalizability is low and the precision of measurement can be affected by confounding factors. A typical method used in field experiments is action research [23], which has similarities with DSR and could be an alternative methodology for this study. Both involve organizational problem-solving and evaluation. According to Baskerville [24], the fundamental difference between DSR and action research is that action research focuses on problems within organizational and social changes, while DSR focuses on problem-solving by creating and placing an artifact in a natural setting. We considered DSR a better fit for this study since the research question does not mainly concern organizational or social changes from using ML, but rather focuses on developing an artifact that will be evaluated and designed to solve a concrete real-world issue.

A DSR research can take different directions and lines of action. This thesis followed the overall guidelines for conducting DSR research within the information systems sector [22]. With this as a foundation, Hevner and Chatterjée [25] created a more concrete framework which this thesis adapted. In this framework, the research

consisted of three different cycles, each with a distinct focus, and the outcome of one cycle impacted the other. In this implementation of DSR Hevner and Chatterjée [25] presents a checklist containing 8 questions that need to be answered for a successful DSR. The questions are presented in table Table 3.1.

1.	What is the research question?
2.	What is the artifact? How is the artifact represented?
3.	What design processes (search heuristics) will be used to build the artifact?
4.	How are the artifact and the design processes grounded by the knowledge base? What, if any, theories support the artifact design and the design process?
5.	What evaluations are performed during the internal design cycles? What design improvements are identified during each design cycle?
6.	How is the artifact introduced into the application environment and how is it field tested? What metrics are used to demonstrate artifact utility and improvement over previous artifacts?
7.	What new knowledge is added to the knowledge base and in what form (e.g., peer-reviewed literature, meta-artifacts, new theory, new method)?
8.	Has the research question been satisfactorily addressed?

Table 3.1: DSR checklist questions based on the framework created by Hevner and Chatterjee [25].

The checklist questions are mapped to the three cycles without any particular order. The cycle that involves the identification of issues and opportunities within the technical system of the company is called the Relevance Cycle. This cycle also involved establishing requirements for the artifact and defining acceptance criteria for artifact evaluation. This cycle is mapped to question (1), (6) and (8).

The cycle related to collecting knowledge in support of the research problem is called the Rigor Cycle. Hevner and Chatterjée [25] described two types of knowledge bases: the first involves research conducted in the current state of the art related to the application domain, and the second includes existing artifacts and processes found in the company domain. In this thesis, the current state of the art was explored mainly through a literature review. To gain domain knowledge from existing artifacts and processes at the company, observation activities were conducted to extract insights from their experience and expertise. This cycle is mapped to the questions (4) and (7) in the checklist.

The final cycle is called the Design Cycle and involves the construction of the artifact, which includes activities such as software development, evaluation, and incorporation of feedback to refine the design. The goal in this cycle is to utilize and combine the findings from the other cycles and the previous iterations to improve the artifact. Hevner and Chatterjée mentioned [25] the importance of having a balance between

construction and evaluation of the artifact to ensure that the findings inform it of the preceding cycles. The Design Cycle is mapped to questions (2), (3), and (5) in the list.

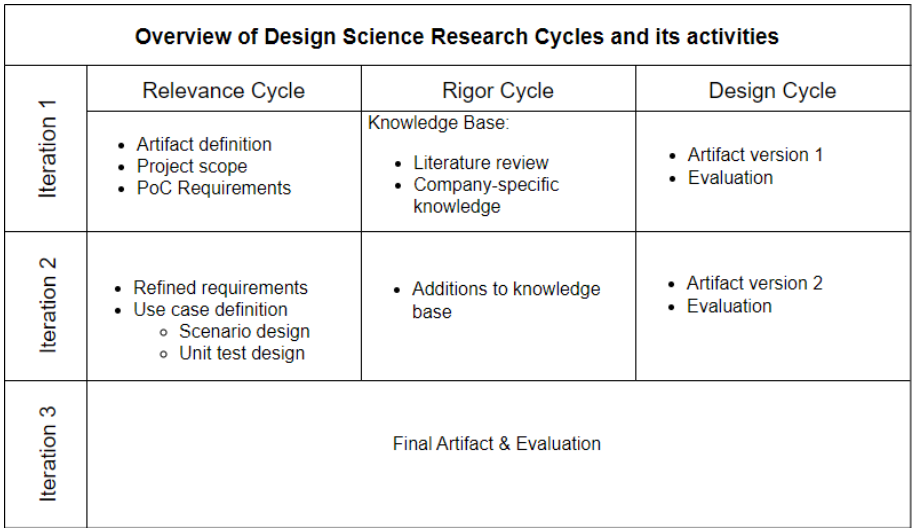

Figure 3.1: Visualisation of the methodology used in this thesis inspired by Hevner and Chaterjée [25] combined with the guidelines derived from Knauss [21].

This study adhered to Knauss [21] recommendations of performing three iterations with the exception that the last one focused on design and performance improvements of the artifact. The iterative nature of DSR made it possible to reflect on the completed iterations and improve their execution stepwise. The Figure 3.1 explains the target and outcome in each iteration categorized by DSR cycle. Each cycle was mapped to RQs based on the content of the cycle. Therefore, RQ1 and RQ2 were mapped to the rigor and relevance cycle, which includes the knowledge base from literature and company expertise. These cycles were mapped this way because their content involved crucial actions for identifying and addressing challenges before designing the artifact in the design cycle. Consequently, the third research question was naturally mapped to the design cycle, where the implementation and evaluation of the MPP algorithm were conducted, directly relating to RQ3.

3.2 Snowballing Literature Review

To investigate the research topic and provide a deeper understanding of the RQs, a literature review was conducted during the Relevance Cycle. It followed a snowballing process which was well-suited for the scope and organization of the study. The literature review was grounded in the guidelines and findings presented by Wohlin [26], which consists of a starting set of papers that you search backward and forward for their citations.

The papers in the start set were initially considered relevant if the topic was about machine learning development and V&V focus in the automotive domain. To further ensure that the start set was appropriate, we followed Wohlin's [26] guidelines for

selecting a good starting set. The following points were addressed with arguments for their inclusion in the start set of this study:

- **The start set should come from different publishers and authors.** The five papers in this study have unique authors and are published in three different journals, which are Institute of Electrical and Electronics Engineers, Society of Automotive Engineers, and SAE International Journal of Transportation Safety.
- **The size of the start set should be related to the breadth of the area being studied.** A more focused area requires fewer papers in the start set. We considered the research questions focused since they pertain to a specific type of technology in a certain domain, with a focus on the challenges related to the technology. This study also considered the type of paper when determining the size of the start set, as systematic literature review papers could help the study snowball into other relevant papers.
- **The amount of citations of the paper.** Since the literature review follows a snowballing approach, the number of citations was important to consider. Two of the papers were systematic literature reviews with a large number of citations, which was beneficial for inclusion in the start set to explore a wider range of papers. The other three papers had more than 36 citations, which still allowed for a round of forward and backward searches for other relevant papers.
- **The existence of relevant keywords in the paper.** This study based this criterion on the title and content of the paper. If it was clear that the topic was about or included the words "validation", "verification", or "challenges", and the area was in the automotive domain, the paper was considered to have enough relevant keywords for inclusion.

The five papers included in the start set are shown in Table 3.2.

Figure 3.2 is a graphical representation of the snowballing procedure, where the number in each box indicates the number of papers considered in that step. Firstly, a backward snowballing was made by looking at the reference list and searching for it on Google Scholar, where relevance first was assessed by looking at the title and publications year. If the title seemed relevant and the publication year within scope, the abstract was assessed. In some cases, a deeper examination of the paper was needed to estimate its relevance. If a paper was considered relevant it was added to the set of papers. Using Google Scholar for the forward snowballing, papers that cited the selected paper could be captured. By doing the same procedure as during the backward snowballing, further papers could be added to the set.

Additional filtering was also performed to maintain a reasonable number of papers and ensure they were up to date. This additional filtering made the literature review more feasible, preventing the need to review all forward and backward citations comprehensively. Therefore, papers with more than 30 citations underwent another filtering among their citations, where a search for the keywords “validation”, “verification”, and “challenges” was conducted. Papers with less than 30 citations were manually investigated directly, without additional filtering. Additionally, no papers

published before 2009 were considered.

Title	Authors	Cites
Challenges in Autonomous Vehicle Testing and Validation	P. Koopman, M. Wagner	690
An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software	Salay R, Queiroz R, Czarnecki K	201
Software Verification and Validation of Safe Autonomous Cars: A Systematic Literature Review	Rajabli N, Flammini F, Nardone R, Vittorini V	73
Machine Learning and Deep Neural Network - Artificial Intelligence Core for Lab and Real-World Test and Validation for ADAS and Autonomous Vehicles: AI for Efficient and Quality Test and Validation	H. J. Vishnukumar, B. Butting, C. Müller and E. Sax	99
Paving the Roadway for Safety of Automated Vehicles: An Empirical Study on Testing Challenges	J Schröder, A Knauss, C Berger, H Eriksson	36

Table 3.2: Table illustrating the start set of the literature review, including how many cites the papers had and what author wrote them.

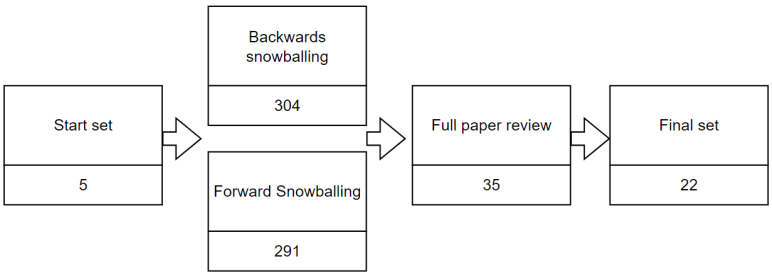

Figure 3.2: A diagram illustrating the progression of steps in the snowballing procedure, where the numbers indicate the number of papers considered at each step.

3.2.1 Scope of the Literature Review

With help from the company's domain knowledge and background regarding the focus of this thesis, there was a need to specify a scope of what to include in the literature review. Since the company's product is a cloud-based off-board system, it is not directly related to AD/ADAS systems and other safety-critical on-board systems. The reviewed literature provided indirect insights or methodologies applicable to off-board systems utilizing ML in the automotive domain. The thesis will further explore these threats of validity insights in section 5.1 regarding how they may differ in the company's system compared to on-board software.

3.3 Terminology, Data, and Company-Specific Settings

The artifacts that were developed in this thesis were evaluated in different procedures to see that the performance, accuracy, and functioning could be validated and verified. This section will go into how the thesis evaluated the artifact by presenting terminology, describing the software that was used, and explaining how the artifact result will be presented.

3.3.1 TrafficWatch

TrafficWatch is the name of the company's software product that is used and developed in this thesis. The software consists of different micro-services with various purposes. TrafficWatch allows you to create simulations where one can select several vehicles, geographical areas, and from which data source the route should be based.

One part of the software is a user interface (UI), where one can create the simulations and also see the vehicles in service. The MPP is also shown in the UI. The vehicle simulations can be created based on several data sources, which also can be selected from the UI.

The micro-service that handles the routing of TrafficWatch also allows the creation of pre-defined scenarios. Scenarios in this context is a data list that consists of road network data points, more explained in subsection 3.3.2. A simulation can then be executed with the defined scenarios which are visualized in the UI.

3.3.2 Terminology and Data Set

TrafficWatch is dependent on processing geospatial data and mapping this data to a road network, which is a process called Map Matching. In TrafficWatch, the road network is represented by a graph of nodes. Nodes are connected by edges that represent possible paths between the nodes and these paths in TrafficWatch are called segments, and represent road segments in a physical road network.

From simulations in TrafficWatch, it was possible to extract data from the road network and its properties. Table 3.3 shows the properties of the saved data exported in a CSV format.

Name	Description and constraints
fromSegmentId	An identifier ID from which segment the vehicle was driving from.
fromRoadSize	A categorical value ranging from 1 to 5, where 1 represents the largest road and 5 the smallest, indicating the road size of the segment from which the vehicle came.
fromSpeedLim	A numerical value that says what speed limit the segment the vehicle came from. A value between 0 and 140.
toSegmentId	An identifier ID which says what segment the vehicle was driving to.
toRoadSize	A categorical value ranging from 1 to 5, where 1 represents the largest road and 5 the smallest, indicating the road size of the segment from which the vehicle traveled.
toSpeedLim	A numerical value that says what speed limit the segment the vehicle went to. A value between 0 and 140.
angle	The angle in radians between the transitioned segments

Table 3.3: Table that shows the name and description of data properties. The table also states constraints for a certain property to show what value it can have.

3.3.3 Training Data Collection with Simulations

The artifact built in the thesis depends on training data. Vishnukumar [27] and Stellet [28] inspired the data collection process, with modifications made to align with the company's product. The training and validation data were collected through simulations in TrafficWatch based on routes from a routing database.

The ML model needed data representing how vehicles have traveled historically, and this thesis used TrafficWatch simulations to collect that data. The simulation system had settings for the number of vehicles running, the region in which they will travel, and the type of vehicle, which could be either a heavy vehicle or a normal vehicle. In this study, the simulation settings included a geographical area covering the whole of Sweden, with 500 vehicles running for 48 minutes. This resulted in a dataset of 6226 points divided into a training set of 5706 points and a validation set of 520 points. The choice of 500 vehicles ensured that TrafficWatch covers various types of roads, including highways, country roads, and city traffic. The simulation collected 6226 data points, which was subjectively considered as a reasonable amount for this study's ML model and enabled the possibility to split the data points into reasonably large training and validation datasets. Each data point tracks how a person makes a choice in traffic, and with 6226 traffic choices, the ML model can recognize driving patterns and understand how drivers intend to navigate.

To ensure the quality of the simulation data, schema testing, and boundary tests were conducted to verify that the parameters were within the correct values and that the headers matched the expected values. Moreover, statistical tools were used to analyze the deviation, mean, and variance of the different parameters in the data sets. This was done to ensure the data set was reasonable and collected realistically. This study highlights the importance of ensuring data quality, which will be further

discussed in the thesis.

The thesis used different data sets to train various ML models. These data sets were sampled using a Python program with the help of Python libraries which are Pandas and NumPy. The sampled data sets were created based on the data from the simulation but generated with different characteristics to provide a variety of data sets.

One of the generated data sets was randomly sampled with constraints as shown in Table 3.3, and its characteristics are illustrated in Figure A.2. Another sampled data set followed a normal distribution based on the mean and standard deviation of the simulated data. The characteristics of this data set are shown in Figure A.3. The third generated data set was with added noise to the simulated data. The noise amplitude was 10%, constrained as mentioned in Table 3.3, and its characteristics are depicted in Figure A.4.

3.4 Alignment with V&V Activities

When showing the result of the artifact, several elements were displayed with the goal that together as a combination, see how the ML Model itself was related to V&V activities, and also see if the artifact could be successfully evaluated. The following elements were displayed:

- **Data set characteristics:** The behavior of an ML model is determined by the combination of its training data and the learning algorithm it employs. An activity undertaken in this thesis involved visualizing an informative representation of the data characteristics, aiming to gain a deeper understanding of the type of data the model was trained on. This was achieved by plotting a graph with a line to illustrate the distribution of the data.
- **Prediction capabilities on validation data:** When the ML component was utilized in TrafficWatch, it had to predict unseen data. By displaying prediction capabilities on validation data, one gained insight into how the ML component performed on unseen data.
- **Test cases in TrafficWatch:** The ML component was utilized in TrafficWatch to showcase how it functions in the company's product. This was displayed in a table that maps test cases, visualizing how it should be predicted compared to what it predicted.
- **Comparison to requirements:** The gathered requirements were reviewed to determine if the artifact met them.
- **Internal model testing:** To ensure that the model was trained with correct specified data within certain constraints, unit tests on the data set and model output were designed and tested. Moreover, schema testing was done to ensure that the server that holds the models' requests and responses is in the correct format.

The aforementioned elements formed the basis of the results to observe correlations between the ML component, training data, and V&V activities. These elements

were also present to determine if an ML component could be successfully verified and validated.

4

Results

This chapter aims to present the main findings of the thesis work. It begins by defining the artifact and discussing the literature review findings. The chapter concludes by describing the design science cycles through iterations in order.

4.1 Artifact

The following section aims to define the artifact that is being developed throughout the design science iterations. As previously mentioned in section 3.1, an artifact refers to the object created to address a practical industry problem.

In this thesis work the artifact corresponds to the Machine Learning prediction model that will foresee the MPP, which is a building block for sending safety-critical information to vehicles. The model will be based on a supervised ML approach suitable for the problem and its context and will be evaluated and improved in each iteration based on findings from DSR cycles. Since this thesis project focuses on verifying and validating such components, the artifact will serve as a foundation for these processes.

Throughout this thesis, we will refer to the artifact as the Machine Learning component holistically, encompassing its training data and V&V activities. The thesis necessitates such a broadly defined artifact as it must be compatible with the company's domain and system while remaining connected to the thesis research questions.

4.2 Literature Review

The literature review is divided into three sections below. The content focuses on how ML software is being developed in the automotive domain and is sorted based on the findings in the literature. The ISO 26262 standard, simulations, and model evaluation will be the main focus, and the content will be used to address the research questions.

4.2.1 ISO 26262 and V&V Challenges for ML Components

The "V" software development model (V-Model) is a reference standard that forms the basis of ISO 26262 and is a methodology and procedure for testing and validat-

ing software components [1]. Koopman and Wagner [1] state that when developing software using the V-model approach, requirements must be well-known and correctly specified in advance, which is challenging for ML models. The requirements for the software component must also be specified within the scope of the system design, as well as what lies outside the scope of the software. Requirements when creating ML solutions must have other forms and specifications [1]. The authors say that validating and testing a trained ML algorithm itself is not much different than traditional validation techniques on normal software. But validation of the applied training set differs substantially and this can't be extracted from the V-Model, but instead being ambiguously planned. The authors think that validating the training data can be done by a combination of characterizing the data and following a data collection process.

Vishnukumar et al. [27] states that a challenge with V&V when developing software for ADAS is the information exchange between the stages of the V-Model. This is partly because the V-Model does not strictly follow the stages in order, and the feedback is then in the form of human-to-human communication which itself has its drawbacks.

Knauss et al. [29] identified different processes and methods when developing software for AVs and conducted a study on the challenges with these. It says that all the requirements are not defined from the start when developing software for AVs. Instead dynamically added and adjusted in a more flexible approach. The same authors also say that there exist challenges in the transition between the development activities when using the V-model and that the V-model's suitability is questionable in this kind of development.

Karunakaran et al. [30] express that traditional testing methods fail to capture the non-determinism of ML systems and the environment in which they operate. Using a finite number of tests with the same input is problematic to evaluate since it might trigger different responses, which makes the results hard to trust.

When developing software for AVs using the V-Model, all requirements must be specified, which is challenging because of their unpredictable nature. This means that all cases cannot be specified explicitly [31]. The author also explains that it is challenging to statistically demonstrate that the system will live up to the criteria while operating.

Rabe et al. [4] state that the development processes defined in ISO 26262 are insufficient for the fundamental changes that the introduction of ML development in safety-critical systems has led to. They propose four phases that the development of an ML application should consist of. The first aims to specify all conditions that the ML model should function in. This can be different weather or road conditions that an autonomous vehicle should be able to operate in. The second phase is data orchestration and preparation, where the authors conclude that in a machine learning model, the data needs to satisfy the defined specifications since it is the training data that represents the source code of a traditional software application. In this phase, they also mention that there is a need to evaluate how well the data represents the specifications. The use of metrics such as coverage of out-of-distribution

and coverage of corner cases can show this. The third phase consists of model training and V&V of the model, where the training step includes model architecture, selection of activation function if applicable, and hyper-parameter tuning. What V&V methods this phased process should include is not implicitly mentioned in the paper. The last step concerns the model's integration with the complete system to assure robust functionality [4].

Radlak et al. [32] state in their report that a dedicated standard built upon ISO 26262, which would cover functional safety of ML-based components in the automotive domain, would be beneficial for reasons such as more efficient work, reduced creation time, simplified review processes, and agreement across the safety expert community. The authors suggest an approach in their report, but a limitation of their suggested standard is that it won't be able to fully validate the ML component since the requirements are not fully defined from the start.

Mohseni et al. [33] gives practical advice on how to, in line with traditional verification and validation for safety critical systems, mitigate the most common challenges for ML models. The authors emphasize the importance of traceability in the V-model from ISO26262 both regarding specification and requirements, which is complex in the ML system since the specification often happens as the model learns which may lead to the discrepancy between the specification and the actual model. The authors mean that this can be avoided by breaking the model down into smaller parts and specifying each one.

4.2.2 Simulation and Scenario Testing

Koopman and Wagner's [1] article about software challenges for autonomous vehicles suggests that it is impractical to create test scenarios that accurately represent real-world scenarios operating randomly and independently for billions of hours. The authors state that alternative validation methods are necessary, such as simulations, fault injection, and formal proofs.

The same authors [1] also mention that testing in fixed geographical regions makes it easier to understand combinations of situations and specify requirements after that. When requirements are specified, progressively more complex situations can be added to evolve the scenarios that are being tested.

Knauss et al. [2] conducted an empirical study in the form of focus groups and interviews, that shows participants of the study mention virtual testing & simulation and scenario complexity & many test cases the most. In Knauss et al. [2], [29], the survey shows the most significant challenges regarding simulation and virtual testing for AVs, which are the points below.

- The legal aspect of certifying certain functionality based on simulation.
- A complex infrastructure is needed to combine a real setting and a simulated setting.
- A simulation environment requires data, and each simulated scenario needs the correct collected data.

Vishnukumar et al. [27] propose a concrete methodology for connecting the testing

and validation of individual functions in a simulation system to create scenarios. The method involves taking data from real-world tests/scenarios and sending it to the machine learning system that is used in the simulation environment. Subsequently, the simulation environment tests the scenarios with the function to be evaluated and provides feedback to the ML system. This feedback is then reviewed by humans to verify the ML system's performance.

Sippl et al. [34] focused their paper on the process of converting simulation data into test cases in a virtual environment for ADAS and ADs. The authors mean that classical methods and concepts for test optimization, such as Equivalence Class Partitioning and Boundary Value Analysis among others, are hard to use to create relevant situations for these systems. They therefore propose concepts for simulation-based development and virtual validation instead of traditional on-road testing for vehicles.

Stellet et al. [28] studied challenges with software in the AD domain with an open focus on programs that are complex and operate in a real-world perception. They concluded that the acceptance criteria of the system are situation dependent, informal, and exist of infinitely many combinations that are hard to predict the outcome of. A set of sampled representative situations has to be done [28], but the number of iterations does not scale which is a challenge. One solution to mitigate this challenge is by using a random sample strategy. The author also mentioned validation challenges with models operating together with vehicle software. This validation has to be clearly shown and explained. One result from their survey about validating software for vehicles is to use a scenario-based approach, where you specify a set of scenarios that work as test cases for the software. The risk and challenge with this approach is to select an incomplete set of scenarios and not achieve completeness in testing. The authors [28] then suggest an approach to use on top scenario-based testing, which is an iterative simulation-driven development that can be described as a continuous improvement of developing software where you include previous scenarios and test-cases in a feedback loop to generate more test cases and validation settings.

Bock et al. [35] stated that data for all relevant situations when developing software for vehicles does not exist, which makes it challenging to validate the functionality with test cases. The authors further say that new methodologies need to be developed to mitigate the challenges and scenario-based development might be one method that can help overcome the challenge.

In Zhang and Li's SLR [36] they investigate ways to ensure completeness and validate NNs. One proposed method is to create scenarios based on formal Scenario Description Language (SDL) and transform them into test cases. This comes with scalability issues, proposed methods only work on small scale or very specific types of models.

Lalli et al. [37] investigate validation methods in their SLR, and it shows that simulation-based validation is popular and stood for 22.2% of the reviewed papers concerning validation methodology, especially in the automotive domain, and the trend is that it will increase even more. One type of simulation is called fully virtual

simulation, where the real scenario inputs are replaced with input derived from the simulator itself. Additionally, Schöner [38] says that computer simulations need defined test conditions and scenarios to validate functionality for AVs. The author states that these situations has to be collected systematically.

Li [39] mentions a scenario-based V testing model, where a scenario database plays a central role in all activities that the method consists of. The author also mentions that scenarios can be created by human experts since generated ones do not cover all cases. The problem mentioned with this approach is that it does not scale well and is expensive to perform.

4.2.3 Data and Model Evaluation

Testing non-deterministic behaviors is challenging [1], and one method is to test if the statistical behavior is correct, instead of testing that the software behavior is correct. Koopman and Wagner [1] state that another challenge for validating an ML component is that it is generally hard to intuitively understand the result of the process since calculations that are used in ML are complex.

Tambon et al. [10] wrote that by verifying an ML component, one aims to check the component concerning a predefined set of specifications. The authors mention that verification approaches of ML models mainly are based on test input generative approaches, which are techniques that aim to trigger failures or unexpected behaviors in systems by exploring corner cases that are carefully selected samples. By feeding the model with these samples, we might explore parts of the data where the model fails to generalize its knowledge, leading to misclassifications or bad predictions. The verification efforts in this domain are split into two separate approaches, where one is based on mathematical verification of the input space and the other is more based on empirically guided testing.

The authors [10] further discuss the idea of testing data directly instead of testing a trained model. They argue that it would add an extra layer of assurance to the model since it learns patterns and makes decisions surely on the data that has been fed to the model.

Salay et al. [3] stated that data requirements can be specified in detail to ensure that appropriate training, validation, and testing sets are obtained for the ML components. The data can be verified to the detailed data requirements and coverage of the test to get a measure of the completeness of the ML component. The authors further mention challenges with this approach, since functionality often needs to fit with a real-world perception of the environment. The author suggests an approach when working with ML components, which is that you should have relaxed specification requirements for the functionality, and only specify functionality specification where it is possible. At the same time, there must be a specification of the training set and the coverage metric. This is important due to the functionality of the ML component is based on the training set.

The literature review by Karunkan et al. [30] touches on the subject of requirements, which they mention becomes even harder to specify and verify in an ML system due to the infinite amount of situations and their complexity. ML systems are built from

training data in contrast to traditional programs that are explicitly programmed from a detailed specification. This means that the data that is being used to train the model needs to reflect these specifications instead. They further mean that if the data fails to do so, it might lead to undesired behavior in the model. To mitigate this, the validation of the dataset should be independent and diverse from the training data set.

Rajabli et al. [31] suggest that the training set collection process can be mapped to requirements specification when developing ML components for AVs. Then, during the V&V stage, you take a subset of the training data as validation data, which was left out to validate the system. The validation data can not have any correlation with the training set, as that can risk overfitting. The same authors also state that with ML components, the incomprehensible structure makes it hard for humans to understand, which makes validation more challenging.

Koopman and Wagner [40] say that even if a machine learning algorithm can perform with 99.999% correctness, is often because of overfitting. Even if you can argue that is not overfitting, it is still an epistemological reasoning you have to do on the empirical data that is used. The same authors mention approaches such as boosting when using a machine learning algorithm, which can be described that you add data from independent different data sets to improve the result.

Kläs and Vollmer [41] assess the uncertainty of ML models due to their complexity and empirical nature. They mean that instead of refusing the use of such models in a safety-critical context since their correctness can not be guaranteed, an alternative is to deal with their uncertainty and propose a classification of uncertainty on such a model. The same authors also proclaim that it is not enough to assess the data quality before training but rather continuously evaluate it through the whole process of collection, training, and model evaluation.

Mohseni et al. [33] highlights the need for transparency in ML models, especially when using Deep Neural Network models. New tools for transparency are often developed for designers to improve the performance of the models and not to increase transparency for safety assessment reasons. The authors also highlight the need for real-time monitoring of ML models to assure safety. One suggestion of real-time monitoring could be Out of Distribution error detection which could warn the system when data outside of the training range occurs. The same challenge is mentioned by Zhang and Li [36] in their literature review. Many papers state that the understandability of a Neural Network model and its decisions is a concern since it is hard to provide real-time explanations. The papers they investigated also mention the lack of standard procedures to verify these types of models.

Myllyaho et al.'s SLR [37] shows that 15 out of 90 papers used a Model-centered validation methodology. This is to gain knowledge about how the model performs based on different metrics. These metrics often come from common ML validation methods such as cross-validation and torture tests.

4.3 Iteration 1

In this section, the initial iteration through the design science cycles in the thesis will be described. The findings of this cycle are based on the literature review and domain knowledge gathered from the company. The artifact resulting from this iteration is a Proof of Concept (POC) ML model, which serves as a baseline for V&V activities and will be improved in later iterations.

4.3.1 Relevance Cycle

During the relevance cycle, the primary focus was to gather requirements for the artifact from the company's context. These requirements formed the foundation for the development of the model. The goal of this cycle was to gather sufficient requirements to support the creation of the initial version of the ML model, avoiding an overload of overly complex specifications that might complicate the implementation. The specific requirements outlined in Table 4.1 were collaboratively defined with the company.

Requirement ID	Description
R1	The MPP model should be compatible with the existing simulation system.
R2	The MPP model should be able to handle 1000 simulated cars at the same time.
R3	The calculation from the model must successfully pass the scenarios defined in the system.
R4	The MPP component should be capable of generating path predictions in real-time to ensure timely response in the existing environment.

Table 4.1: A table including a requirement identifier and a description. These requirements were created together with the company.

This iteration also contained activities on understanding the company-specific applications. With this focus, it was possible to ensure that the model development and integration process would be well-aligned with their current technical stack.

4.3.2 Rigor Cycle

The rigor cycle in this iteration mostly focused on the literature review, which was an important part of creating the foundation of the knowledge base. The literature review itself was carried out with the ambition to answer the first two RQs and what was believed to be the most important details have been compiled in Table 4.2.

Topic	RQ1 - Challenges	RQ2 - Mitigation
ISO 26262 V-Model	Needs to specify predefined requirements [1], [10], [31]–[33]. The procedure of how to validate the training set cannot be extracted from the V-model [1]. A dedicated standard for ML-based components built upon ISO 26262 does not exist [32].	Requirements should be dynamically added and adjusted in a more flexible approach [29]. More relaxed specification of requirements [3]. Usage of testing in fixed geographical regions to make it easier to understand combinations of situations and specify requirements after that [1].
Scenarios Simulations	Function in with real-world perception of the environment [3], [28], [30], [35], [41]. The legal aspects of verifying functionality and complex infrastructure [2], [10]. Having an incomplete set of scenarios and proof of achieving completeness [28]. Having the correct data set [2]. Challenges with the scalability of creating scenarios [36].	Scenario-Based testing [35]–[39]. Progressively add more scenarios for a wide range of situations [1].
Data Validation	It is challenging to collect the correct data for vehicle traffic [29].	Investigate characteristics and follow a data collection process [1]. Testing the data directly and using mathematical-based verification of the input space [10]. Specification requirements for the training set and a data collection process [3], [31].

Table 4.2: A table that includes topics connected with RQ1 and RQ2, presenting a condensed summary of what literature says about V&V of ML software in the automotive domain.

This cycle included extracting information from the company domain, mainly regarding the simulation system and data collection process with the goal of building and later evaluating the model. The company had two main approaches to validate the MPP algorithm:

- Uses a basic scenario-based approach to test the MPP implementation in their simulation system. A vehicle travels from start to end, and unit tests compare the predicted path with the path traveled.
- Displays the most probable path in real-time in the simulation client as a line to visually verify the result.

4.3.3 Design Cycle

This cycle aims to create and evaluate the artifact by incorporating findings from the two other cycles. Specifically, the artifact was created with a foundation from the relevance cycle that included the company-specific environments, such as TrafficWatch, together with the requirements. In contrast, the evaluation processes of the artifact are mostly inspired by the Rigor cycle and its findings presented in

Table 4.2.

The goal for this cycle was to build an initial version of the artifact serving as a POC model, fully functional with the existing system and fulfilling the expressed requirements. It is important to note that this iteration was not conducted to optimize the model's predictive capabilities, which is why this section only consists of high-level descriptions of the POC model. The model is a multivariate RandomForestRegressor that receives information about vehicle and segment attributes and then predicts three output variables, as shown in Table 4.3, where features are inputs and labels are output values. Next, the potential segments are filtered based on their distance from the prediction, and if any segment falls below a predetermined threshold, that segment is returned. Communication with the component is handled through a REST API. Figure 4.1 describes the overall structure and data flow of the most relevant parts of the system, to provide a high-level understanding of how the ML model is being served.

Name	Used as
`fromSegmentId`	Ignored
`fromRoadSize`	Feature
`fromSpeedLim`	Feature
`toSegmentId`	Ignored
`toRoadSize`	Label
`toSpeedLim`	Label
`angle`	Label

Table 4.3: Table showing the data properties and how they were used to train the model. The "Used as" column indicates whether the property was ignored (i.e., not used in the training algorithm), used as a feature (input to the training algorithm), or used as a label (output that the ML model will predict

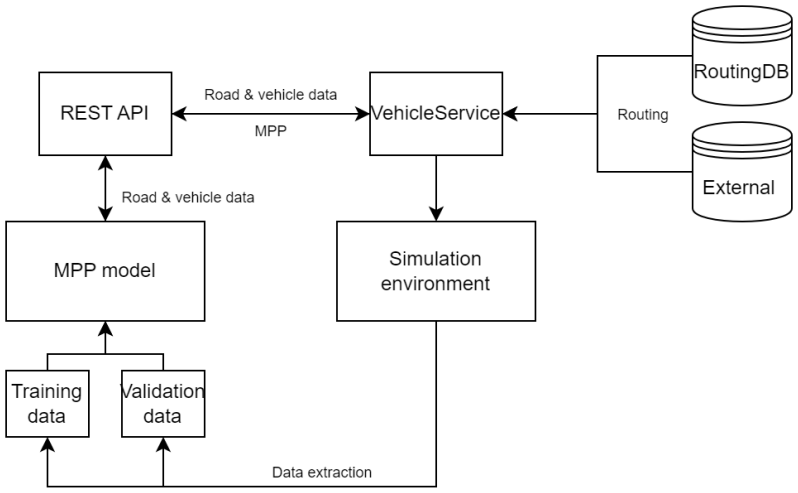

Figure 4.1: Figure illustrating the overall structure and flow of the relevant part of the system. It includes the ML model calculating the MPP, the REST API serving the ML model, visualization of the origin of routing in the simulation system, and the layer where data extraction occurs. The arrows indicate the direction of the flow.

In this development phase, we focused on implementing a robust data collection process, which proved crucial for effectively training and evaluating the model. Vehicle movements were simulated using predefined routes sourced from routing databases. This allowed essential data capturing, including vehicle positions and other relevant vehicle information, which could then be logged and stored within the system. Later, this data was extracted and divided into distinct sets for testing and training purposes. This approach ensured that the models were trained on comprehensive and diverse datasets, enhancing their accuracy and reliability in real-world scenarios.

4.3.4 Review and Evaluation of Artifact

The training data consisted of 5706 rows and the validation set was 520 rows. Each row had 7 properties, which were either used as a feature, label, or ignored in the training process of the model. Both toSegmentId and fromSegmentId were not used in the model's training but were still extracted from the simulation environment to track how vehicles traveled between segments. Graphs on the characteristics of the two data sets are shown in Figure A.1 and Figure A.5.

To provide information from the model in an understandable way, its predictions were extracted and plotted, as seen in Figure 4.2. The model bases its decisions on a combination of all three output values, but here they are presented each one in its plot. The X-axis represents the true value while the Y-axis represents the predicted values and the blue area is where 97.5% of the predictions end up. The plots aim to provide a more intuitive representation of the models' predictions, and can easily be used when comparing the predictive capabilities between different models. These plots also help when trying to identify anomalies in the data. As an example, in this dataset, the vehicles seem to turn with an angle (in radians) close to π or less. In our representation of the turning angle, this is a slight right turn. Since the data was collected in an environment with right-hand traffic, it makes perfect sense to assert that vehicles more often turn right than left, especially on larger roads. In other environments, in left-hand traffic, for instance, this representation would not make sense. These plots can then be used to identify assumptions made in the data and graphically point them out.

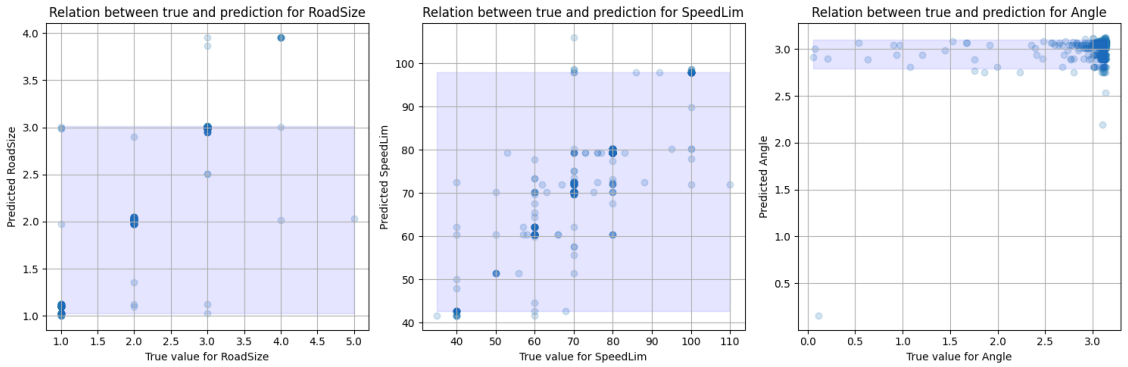

Figure 4.2: ML model prediction on validation data. The blue area is where a 97.5% interval of the prediction is made and each predicted data point is a blue dot with an opacity, which means that the darker the blue color, the more predicted values lay on that coordinate.

4.4 Iteration 2

Iteration had a distinct focus on adding to, improving, and refining the verification and validation activities conducted in the initial iteration. This section systematically presents the activities and findings for each cycle.

4.4.1 Relevance Cycle

Based on the findings from the literature review, the activities in the relevance cycle focused on explicitly specifying the context and environment in which the model should be utilized. The first activity involved demonstrating the POC model functioning in TrafficWatch and evaluating metrics from its prediction capabilities. This demonstration confirmed the model's ability to function within TrafficWatch by predicting and visualizing the MPP but also underscored the need for additional validation metrics to ensure model quality. Thus, a more focused approach was adopted, involving visualization of specific scenarios, comparisons between models, and showcasing the overall predictive capabilities of different models.

The initial model was trained on data covering various roads to generalize the problem as much as possible. This meant the simulated vehicles used to generate training and validation data traveled through the city and sparsely populated environments. To better align with customer demand, the model's focus was refined specifically for larger roads and highways. This consideration influenced the definition of the scenarios.

The literature review highlights the complexity of crafting simulated scenarios due to the environments in which the ML model operates. To develop meaningful scenarios for visual validation of the model's behavior, a more precisely defined scope was established to reduce the number of scenarios. This approach allowed the definition of scenarios closely resembling real-world traffic use cases, tailored to meet customer demands. In Table 4.4, these scenarios are defined along with an ID, description, and the expected behavior.

ID	**Scenario Description**	**Expected Behaviour**
TS1	Vehicles approach a Driveway, and exit onto the highway.	Should predict the route out on the highway.
TS2	Approaching a roundabout and drives out to the highway.	Should predict out on the highway.
TS3	Approaching an highway exit and exit the highway.	Should predict that it is still going to the highway, but when the vehicle takes the exit, a new prediction should immediately be calculated and shown.
TS4	A vehicle drives on a road where two exits is in is future path between A to B. The vehicle is not taking these exits but instead drives on the same road.	The model should predict the same road and not predict the exits both of the times.

Table 4.4: Table showing Test scenario identifier, description, and its intended behavior. This shows how the scenarios are documented and defined.

4.4.2 Rigor Cycle

In this cycle, additional literature was added to the knowledge base to cover more of the topics that were missing from the literature review conducted during the first iteration. Breck et al. [42] researched validating data when the system is deployed and continuously trained using an ML pipeline. The authors split the data validation steps into three different parts where each part has a distinct focus. The first part is called single batch validation in which the authors aim to validate the daily batches with training data that is used to continuously train the model. This is done by using a data schema validation framework that alters the developers that a problem has occurred somewhere in the data generation and that it should be investigated.

The second part of their data validation process is called Inter-batch validation and aims to detect significant changes between different batches of data. This can both be about two sequential batches of training data, but also a difference when the (daily) training data is considered with the staging data that the model operates on.

The third part is model testing, which focuses on finding mismatches between the expected data and assumptions made in the training code. The authors [42] mean that it is reasonable to assume that training code makes assumptions about the data that is not explicitly expressed in the schemas from the previous stages. The unit tests are based on fuzz testing and data can be randomly generated to adhere to the limitations of the schemas.

Braiek and Khomh [43] aggregate findings from published papers regarding data validation for ML models. The authors emphasize that the data is vital for a model's performance, and to test the quality of the data statistical metrics such as mean, variance, and sum are used to detect significant differences between expectation and the data.

Qi et al. [44] investigated the performance of ML models across various datasets amidst injected errors. It reveals a consistent negative impact on model performance metrics such as Precision, Recall, and F-measure due to error-prone data. The extent of this impact varied based on error type and rate. Moreover, the study finds that larger datasets mitigate the adverse effects of errors, advocating for the utilization of large datasets to reduce the impact of data inaccuracies on model training.

4.4.3 Design Cycle

The second iteration of the design cycle focused on inputs from the relevance and rigor cycle. From the relevance cycle the ML model needed to showcase how it acted on specific scenarios and compare metrics to other models. The design cycle resulted in three new ML models in addition to the one in the first iteration. The new models were trained with the data set of other characters compared to the model from the first iteration based on data from a routing database.

The first new model was based on randomly sampled data collected within some constraints to align with TrafficWatch. These constraints are explained in Table 3.3. The second model was trained on data with characters following a normal distribu-

tion with mean and standard deviation extracted from the routing database. The third model was trained on data from the routing database but with manually added noise. The character of these data sets is shown on graphs in Appendix A.

4.4.4 Review and Evaluation of Artifact

In this section, for each ML model, including the scenario result, prediction capabilities, and the testing result inspired by Breck et al.'s [42] research.

Scenario Results

The designed simulation scenarios were tested and evaluated with a visual representation of the MPP in TrafficWatch. This was done with actors from the company. The description and behavior of each scenario are explained in Table 4.4 and its result is shown in Table 4.5. The importance of the scenario results is to observe how different data sets used to train the ML model affect its prediction behavior, which can be validated by running scenarios as a validation activity. By defining scenarios that reflect real-world situations, you can validate the behavior of a software component to determine if it can handle specific cases. The results show that different data sets affect the test scenarios, which is crucial because it helps identify which data sets and ML models are sufficient for a particular context. For example, our scenario results indicate that none of the models were sufficient for the company's context, as TS3 failed. The fact that all models failed TS3 indicates that it requires more careful examination. We identified two possible reasons for the failure: either the data sets were insufficient for the ML model to generalize the intended behavior of the scenario, or the scenario itself is poorly designed and needs to be modified. This result is significant because the functionality of the software component must be validated and verified before being used in real-world applications, especially in the safety-critical automotive domain.

Prediction Capabilities

To evaluate the prediction capabilities of the ML model in iteration 2, the thesis showcases this in the same manner as in the first iteration. This is achieved by presenting various ML models trained on different datasets and observing their performance on validation data.

The Random Sampled data set makes poor predictions on validation data. The model cannot make feasible predictions on unseen data to provide the most probable path. Different features yield various predicted values, as illustrated in graphs in Figure B.1 depicting its behavior on validation data.

Investigating the prediction capabilities of an ML model trained with the normally distributed data set resulted in reasonable outcomes. The model predicts that a vehicle will travel on a road with a higher speed limit and a moderate angle. However, it tends to predict that the road size is small, which is not preferable according to the company's requirements for the MPP functioning. This could be validated by examining the plot shown in Figure B.2.

Data set	Scenario	Result
`Random sampled`	TS1	Failed.
	TS2	Failed. Predicting more rounds in a roundabout.
	TS3	Failed. Predicting nothing, not the exit and not the highway.
	TS4	Failed. Predicting just to the exit and nothing after.
`Normal distributed`	TS1	Passed
	TS2	Failed. Predicting the wrong exit.
	TS3	Failed. Predicting nothing, not the exit and not the highway.
	TS4	Passed.
`From route database`	TS1	Passed.
	TS2	Passed.
	TS3	Failed. Predicting nothing, not the exit and not the highway.
	TS4	Passed.
`Added noise`	TS1	Passed.
	TS2	Passed.
	TS3	Failed. Predicting nothing, not the exit and not the highway.
	TS4	Passed.

Table 4.5: Table of how the ML model was performing on scenarios from Table 4.4 when it was trained with certain types of data.

When using data from a routing database, the model exhibits good prediction capabilities on the validation data. The model prefers turns with small angles and predicts larger road sizes with higher speed limits, as illustrated in Figure B.3. This aligns with the company's requirements, but there is a high risk of overfitting and alignment only with the company's specific context.

As the routing database demonstrates good prediction capabilities, there may be bias in the results due to the structure of the validation data. Therefore, the thesis also trained a model using data from a router database with manually added noise. The prediction capabilities of this model are shown in Figure B.4, which, compared to Figure B.3, predicts wider angles, larger road sizes, and higher speed limits. Additionally, it predicts fewer out-of-distribution results for road size and speed limit.

Unit- and Schema Testing

The artifact evaluation process was in this iteration with a distinct focus on data validation. Since many of the papers in the knowledge base highlighted this topic, it was natural to explore data validation more thoroughly. In line with the approach used by Breck et al., [42], data schemas for both training data and API calls were added to the component, which can be found in Figure C.1. This implied that the system automatically checked the validity of the data to ensure that the model trained on data that meets the expectations.

In addition to schema validation, unit tests were created with inspiration from the

third step of the Breck et al. [42] data validation process and Tambon et al. [10]. Data was here generated from the schemas to ensure that they were coherent with the requirements and functionality of the model. An overview of the test cases can be found in Table 4.6.

Test	Input(s)	Expected	Result
`test having correct headers`	data-sets *	'fromRoadSize', 'fromSpeedLim', 'toRoadSize', 'toSpeedLim', 'with-Angle'.	Passed
`test angle data having correct intervals.`	data-sets *	$0 < Angle < 3.146$	Passed
`test from road size data having correct intervals and type.`	data-sets *	Integer between 1-5.	Passed
`test from speed limit data having correct intervals`	data-sets *	A number between 10-150.	Passed
`test that road size data having correct intervals`	data-sets *	Integer between 1-5.	Passed
`test that speed limit data having correct intervals.`	data-sets *	A number between 10-150.	Passed
`test noise model makes multiple outputs.`	Noise ML model.	Multiple road choices**	Passed
`test normal distributed model makes multiple outputs.`	Normal distributed ML model.	Multiple road choices**	Passed
`test random model makes multiple outputs`	Random ML model.	Multiple road choices**	Failed
`test router database model makes multiple outputs`	Router database model	Multiple road choices**	Failed
`test noise model out to highway scenario`	FromRoadSize = 2, FromSpeedLim = 80 & ToRoadSize = 1, ToSpeedLim = 100	Highway prediction output ***	Passed
`test normal model out to highway scenario`	FromRoadSize = 2, FromSpeedLim = 80 & ToRoadSize = 1, ToSpeedLim = 100	Highway prediction output ***	Failed
`test random model out to highway scenario`	FromRoadSize = 2, FromSpeedLim = 80 & ToRoadSize = 1, ToSpeedLim = 100	Highway prediction output ***	Passed
`test rdb model out to highway scenario`	FromRoadSize = 2, FromSpeedLim = 80 & ToRoadSize = 1, ToSpeedLim = 100	Highway prediction output ***	Passed

* All data-sets that is shown in Appendix A.
** Multiple road choices which are roadSize and speedlimit as [[1,90], [1,100], [2,80], [2,90], [2, 100]], when it comes from [3,80]. The result should include at least 4 options.
*** 0.9 < Predicted roadSize < 1.1 & 91 < Predicted speedlim < 109.

Table 4.6: Unit and behavior tests for the ML component.

4.5 Iteration 3

This section will cover the final iteration of the DSR where the focus was to provide a final artifact to the company, which will include the ML model and the V&V activities.

4.5.1 Final MPP ML Model

The final MPP model utilized a supervised Neural Network machine learning approach with training data derived from a routing database in TrafficWatch. By examining the plot depicted in Figure 4.3 alongside those from iteration 2, it becomes evident that the predictions are slightly more condensed compared to earlier iterations. The model aligned with the company's requirements, and passed the required scenarios and tests, thus becoming the final model for the thesis.

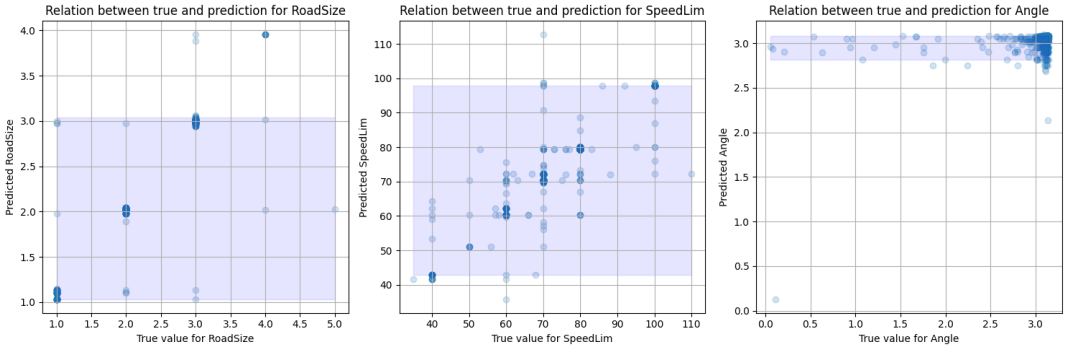

Figure 4.3: An illustration of how the neural network model performs on validation data, where the blue area shows where 97.5% of the prediction takes place.

4.5.2 Validation & Verification

This artifact was evaluated by activities inspired by the two previous iterations of the thesis. In this iteration, a compiled and bundled approach to the activities was made for the V&V of the entire artifact. Figure 4.4 explains the high-level structure of the system and how different components interact. The numbers on the arrows represent a V&V activity and are further explained in the following list:

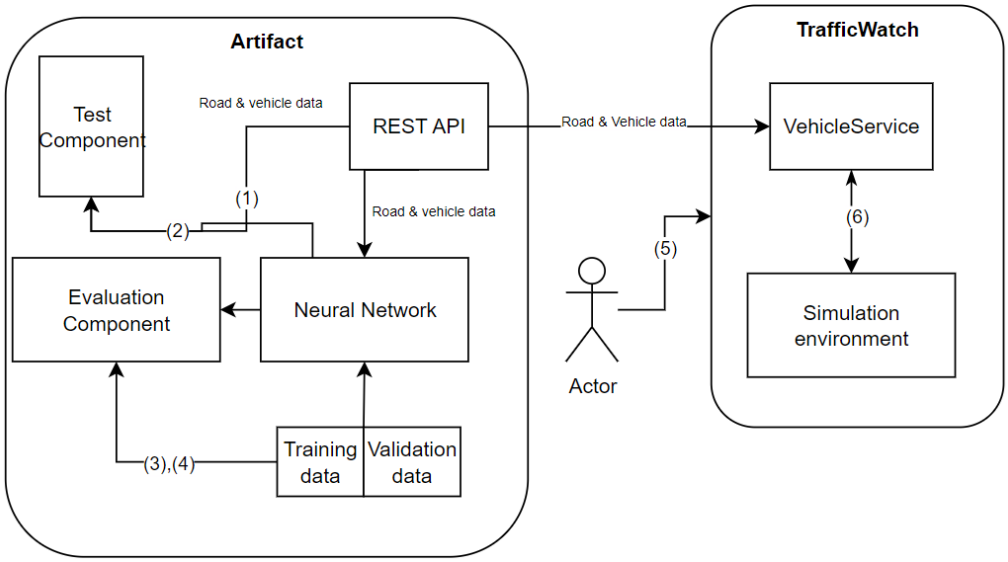

Figure 4.4: Figure describing the final version of the artifact interacting with TrafficWatch. The numbers in the figure illustrate where different V&V activities for the artifact took place. The V&V activity numbers are mapped to the enumerated list under subsection 4.5.2.

(1) **Data Schema Testing:** A verification process that ensures that the road and vehicle data have the correct structure and fall within the boundary values. This activity ensures that requests and responses are handled correctly.

(2) **Unit Testing:** A verification process that ensures that the expected behavior of the functions in the ML model (a Neural Network for the final iteration) is

working correctly.

(3) **Data Characteristics:** A verification activity that includes generating plots for the data characteristics to provide a visual understanding of how the ML model is trained.

(4) **Prediction Capabilities:** Verification activity involves visually demonstrating the prediction capabilities of the ML model to understand how it predicts the given problem.

(5) **Visual Scenario Testing:** A validation activity that involves an actor visually inspecting TrafficWatch to assess if the MPP algorithm works as expected on predefined scenarios.

(6) **TrafficWatch Specific Tests:** When modifications are made to the company's existing system, tests need to be run to verify that the existing system functions as intended.

5

Discussion

This chapter is dedicated to discussing the results obtained from the DSR. It is organized by first delving into the main findings and the company-specific findings. Following this, each research question is addressed, and the results are analyzed and discussed.

The main findings of this study are, firstly, that there are no suitable dedicated standards for ML software development in the automotive domain. Secondly, there is a lack of literature addressing concrete V&V activities of ML software in the automotive domain and illustrating how they assure their software's correctness. Moreover, challenges for V&V of ML components exist, with the main ones being: the difficulty of specifying requirements in advance for an ML component [1], [10], [31]–[33], the need to reflect real-world perception [3], [28], [30], [35], [41], and the challenge of collecting appropriate data for the ML model [2]. Furthermore, some high-level concepts are presented in the literature for mitigating the challenges mentioned above. These include adopting a dynamic approach to specifying requirements [29], utilizing scenario testing [35]–[39], and thoroughly investigating data characteristics to ensure the collection of appropriate data and creating requirements for the data collection process and the data itself [1], [3], [31].

There are some areas in the literature that align with the findings from the company. Firstly, defining specific requirements in advance for the MPP was not feasible due to the complexity and the numerous aspects to specify. Since the company clearly understands the goals and expected overall behavior of the algorithm, their expertise can be used to formulate scenarios for scenario testing, making it an appropriate validation method. The visual nature of scenario testing provides the company with a deeper understanding of how the model might function in a real-world setting. It also complements their unit testing of the model, which has proven to be both inefficient to create and insufficient for supporting the overall quality assessment of the model.

RQ1: What challenges exist when validating and verifying supervised Machine Learning models used in connected vehicles in a safety-critical context?

The initial RQ focused on investigating the challenges that follow V&V processes in ML development for the defined scope. The results show the need for refinement of existing standards and V&V processes. Many papers focus on the ISO 26262 standard and the V-model, stating that these processes are more suited for traditional

software development than ML and its implications on the development process. Mainly because these processes and standards rely on general software engineering principles, such as stating requirements in advance. Literature states that specifications in ML development differ substantially, and hence they cannot be extracted directly from the V-model [1], [30]. ML models also tend to be complex components, meaning it can be difficult to utilize traditional testing methods such as equivalence partitioning testing for verification. The non-deterministic nature of many models also impacts the assessment of test results in contrast to current standards and processes, as we cannot be sure of receiving the same results each time the tests are run [1], [41].

Based on the nature of ML algorithms, data is vital which raises concerns in the literature on how to collect and assess the quality of the data sets used in the training phase of the model. Subsequently, the need for simulation-based testing and the challenges that follow with scenario creation and assessment were highlighted. This topic was considered a major concern especially in the AD domain since the complexity of scenarios aims to represent the complexity of the real world.

Assessing the literature about the current challenges in this domain provides a basic understanding of what topics need to be assessed when performing V&V on an ML component. The complexity of an ML model arises from its construction using numerous building blocks within the training algorithm, making it challenging to interpret the flow and decision-making of the model. This is important to highlight and be mindful of when developing ML components.

The challenge in the data collection process arises from the strong dependence of an ML model on data, necessitating strict requirements for systematically defining the collection process. This also extends to the evaluation process of ML models in the automotive domain, as relying solely on prediction capabilities is insufficient for safety-critical systems. The overall significance of these observed challenges lies in identifying the gaps in existing processes, paving the way for the ML development of a comprehensive common standard that the automotive industry can adopt.

The relevance of the findings lies in their application to the development of ML models across a broad spectrum within the automotive domain. This is particularly important in safety-critical contexts, such as in the development of ADAS and AD technologies. Our findings emphasize the absence of dedicated standards regarding requirements specification and scenario testing for ML development in the automotive domain.

RQ2: What methodologies and techniques exist to mitigate the challenges of validating and verifying supervised predictive Machine Learning models used in connected vehicles to enable accurate safety-critical information delivery?

The primary recommendations for mitigating the challenges identified in RQ1 primarily revolved around three key areas: specification or requirements creation, scenarios, and visual validation, and data collection and data validation. The issue of specifying requirements and specifications in advance can be alleviated through the adoption of a more dynamic and iterative approach. Emphasizing the incorporation

of ML component specifications into the dataset utilized for model training underscores the importance of domain expertise, a systematic data collection approach, and a thorough data validation process. Domain knowledge also plays a crucial role in the creation of simulation scenarios to cover the most vital and common use cases.

One major challenge identified in the initial research question pertains to establishing trust in the model and its predictions. Through a systematic approach to data collection and data validation, together with a visual display of its predictions, the study suggests that the interpretability of the model can be increased. This implies that with a rigorous data validation process in place, stakeholders can have greater confidence in the reliability and accuracy of the model's outputs.

Related studies on this topic often highlight the type of quality assurance required to enhance trust in ML components. However, they often hesitate to propose specific activities to achieve this goal. In contrast, this study diverges by systematically documenting challenges and, within the domain of ML, endeavors to identify and suggest actionable activities to address them. The activities mentioned at the beginning of the discussion section include using a dynamic approach to adding requirements, utilizing scenario testing, and adopting a systematic approach to data requirements and tests on the data.

It is important to note that establishing trust in a model is a multifaceted process that goes beyond data validation and interpretability alone. While these factors are undoubtedly important, trust in the model is also dependent on factors like model performance, accountability, and the alignment of model outputs with stakeholders' expectations and objectives. Therefore, a more holistic approach to addressing trust-related challenges may be necessary.

RQ3: How can the performance of an ML component predicting the Most Probable Path be verified and validated in the context of delivering safety-critical information?

The focus of this RQ has been to propose, execute, and analyze various tools and methods to ensure that an ML component works as intended. By leveraging the knowledge base, several activities have been proposed and implemented throughout the process. Some of the identified techniques used for V&V in this thesis have been evaluated using models trained with different datasets. The results obtained underscore some of the findings from the literature review, emphasizing the importance of specifying requirements for the training data and utilizing scenario testing as a validation method. As an extension to that, this thesis highlights the necessity of visualization, both regarding data characteristics and a model's predictive capabilities to mitigate the complex nature of ML models, since it helps understand how the ML model behaves.

Simulations and simulated scenarios were another well-documented approach on how to validate ML components, especially within the automotive domain. Even if the artifact is not directly connected to AD/ADAS, as the literature often was related to, this thesis work could provide a practical example of how to create, focus, and evaluate scenarios. This was proven to be particularly effective when validating different models. The primary significance of the findings lies in the application of suggested

mitigation methods derived from existing literature into practical contexts, where they can undergo assessment and evaluation by domain experts. Notably absent are concrete examples of V&V procedures for ML components within the context of ADAS and AD. This thesis aspired to bridge this gap by combining practices from the fields of ML and automotive engineering.

One unexpected result emerged in the scenario testing during iteration 3, showing that all models failed TS3. This outcome helps the company identify potential real-world situations where the model may struggle, allowing them to act accordingly. It is worth discussing the importance of designing scenario tests correctly and interpreting the results carefully. Notably, the ML model trained on a dataset with 10% noise produced as many correct scenarios as the data from the routing database. This indicates that lower-quality data, which yielded poorer prediction capabilities, performed equally well on the set of scenario tests.

This thesis utilized visual assessment to validate simulated scenarios where different model was deployed. The difference in results between different models proves that this visual evaluation could be used as an effective validation tool. However, it is important to note that this type of evaluation is always subject to personal bias and in some situations also dependent on the system's environment and how the scenario has been designed.

5.1 Threats to Validity

This section aims to identify possible threats in the findings of this thesis and it is categorized into internal- and external validity.

5.1.1 Internal Validity

The internal validity of the findings in the thesis is questionable due to potential biases resulting from collaboration with a company that creates a contrived setting for the study. The company's involvement may influence the findings to align with their product, thus potentially compromising objectivity in addressing customer satisfaction. The recommendation to visually display the MPP and use simple scenario testing aligns with existing literature. However, it is worth noting that these studies often use more complex infrastructure when designing scenarios, which could impact the findings. This thesis still maintains that visually validating the output of an ML component aids in validating its functionality. We believe this effect will extend to other automotive contexts as well.

Additionally, the findings may be limited to TrafficWatch and its ability to simulate vehicles, neglecting other simulation systems in the automotive domain with differing underlying infrastructures. To address this limitation, it's crucial to consider two factors: firstly, the developed ML component in this thesis is an outsourced component communicating solely with TrafficWatch, thus remaining isolated and usable independently. Secondly, most simulation systems utilizing geographical maps, typically rely on fundamental concepts outlined in the background, such as weighted graphs with nodes and edges, suggesting at least some degree of commonality across

systems. This is still important to mention since this study's results might not apply to all simulation systems.

5.1.2 External Validity

Generalizability is a well-known concern when conducting research within a clearly defined setting, as has been the case in this thesis work. Given that DSR was chosen as the research methodology, there has been a distinct focus on providing findings that are applicable beyond the confines of the company. The artifact developed is specifically tailored and integrated to support the company's needs and is considered to have low or no generalizability, as its implementation details and data are proprietary to the company.

In contrast to the artifact, the V&V processes presented and implemented throughout this thesis could be considered general and relevant for ML purposes, V&V activities, and other implementations in the safety-critical domain. However, it is important to note that these activities, such as scenarios, are not designed to be directly applicable to other settings without modification to suit the specific needs of those settings.

Another factor that influences the findings of the thesis is related to the chosen methodology, DSR, which was selected due to the nature of the research questions and the real-world industry problem. This study takes place in a natural setting influenced by the company. Conducting this kind of research affects the results since there might be confounding organizational factors that influence the outcomes, and there is a lack of generalizability in the results [23]. Therefore, there is a risk that some findings might not be relevant in other similar settings. For example, the discussion about how visual aids will help validate ML software in a safety-critical context might only be relevant to the company or only for ML software that includes a simulation environment related to GIS.

5.2 Future Work

While this study has provided insights into V&V challenges for ML models in the automotive domain, we have identified several possibilities for further investigation. As previously mentioned in subsection 5.1.2, this study was carried out in a natural setting with the company, risking confounding factors and generalizability. Future studies could expand this by further generalizing the activities suggested in this study to provide extended value to this and closely related research fields. One suggestion is to examine a more general process to create scenarios from high-level specifications on the software component.

This study has investigated challenges with existing processes and standards in the automotive domain. With the continuation of increasingly autonomous vehicles, this subject will be more relevant in the upcoming years. Since this study focused on identifying challenges within ML development in the automotive domain, we suggest that future research provides concrete suggestions for refining current standards and development processes to align with this domain.

Another interesting and active area of research is Machine Learning Operations (MLOps), which focuses on the maintenance and deployment of ML models. We believe that several of the V&V activities suggested in this study, especially data validation and scenario-based testing, are well-suited for inclusion in MLOps practices. Therefore, it would be interesting to see a continuation of this study by extending the scope towards MLOps.

6

Conclusion

The initial objective of this thesis was to investigate the most critical challenges for validating and verifying machine learning components in the automotive domain. Subsequently, this design science research aimed to provide practical examples of how to mitigate the identified challenges in a real-world setting by using activities identified in the literature. A predictive most probable path machine learning algorithm was developed and served as the artifact and objective when performing the validation and verification activities.

The results show that there is currently no common suitable process for verifying and validating a machine learning component in the automotive domain. Additionally, traditional software development standards and processes need to be updated to better suit machine learning development in a safety-critical environment, particularly supporting a more dynamic approach to requirements elicitation and not having all requirements specified in advance. This study emphasizes the necessity of understanding data characteristics for evaluating the structure and predictive abilities of machine learning models. Such comprehension is crucial for demystifying the decision-making processes within these models, thereby tackling their inherent complexity and the machine learning model's dependency on data. Additionally, the study underscores the critical role of domain knowledge in the validation and verification phases, particularly in the formulation of requirements and validation scenarios.

By presenting an artifact created in a real-world context, this study contributes to academia by highlighting one practical attempt to address challenges found in the published literature. The results provide industry practitioners with examples of conducting validation and verification activities for machine learning components, drawing on resources from both the automotive and machine learning literature. The study evaluated the artifact in a natural setting, showcasing the results of validation and verification activities and their relation to different datasets with various characteristics. To provide deeper insights into the automotive and related domains, we suggest that future studies explore these activities in a more general setting.

Bibliography

[1] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016, ISSN: 23275626, 23275634. [Online]. Available: http://www.jstor.org/stable/26167741 (visited on 02/19/2024).

[2] A. Knauss, J. Schroder, C. Berger, and H. Eriksson, "Software-related challenges of testing automated vehicles," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017, pp. 328–330. DOI: 10.1109/ICSE-C.2017.67. (visited on 04/16/2024).

[3] R. Salay, R. Queiroz, and K. Czarnecki, "An analysis of iso 26262: Using machine learning safely in automotive software," *arXiv preprint arXiv:1709.02435*, 2017. (visited on 04/16/2024).

[4] M. Rabe, S. Milz, and P. Mäder, *Development methodologies for safety critical machine learning applications in the automotive domain: A survey*, Sep. 2021. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021W/SAIAD/papers/Rabe_Development_Methodologies_for_Safety_Critical_Machine_Learning_Applications_in_the_CVPRW_2021_paper.pdf (visited on 02/21/2024).

[5] B. Ryder, B. Gahr, P. Egolf, A. Dahlinger, and F. Wortmann, "Preventing traffic accidents with in-vehicle decision support systems-the impact of accident hotspot warnings on driver behaviour," *Decision Support Systems*, vol. 99, pp. 64–74, 2017, Location Analytics and Decision Support, ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2017.05.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923617300829 (visited on 04/16/2024).

[6] A. Arvidsson and J. Hendén, "Real-time route prediction of emergency vehicles," *odr.chalmers.se*, 2022. [Online]. Available: https://odr.chalmers.se/items/c2c26d43-4f03-4a1b-b201-698d6c7251b7 (visited on 05/02/2024).

[7] V. Wiklund and A. Rosenberg, "Optimizing road network weight calculation for emergency vehicles," *odr.chalmers.se*, 2023. [Online]. Available: https://odr.chalmers.se/items/f660b7db-aa59-487f-b3e1-fe4c492a7709 (visited on 05/02/2024).

[8] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen, "Path prediction and predictive range querying in road network databases," *The Vldb Journal*, vol. 19, pp. 585–602, Aug. 2010. DOI: 10.1007/s00778-010-0181-y.

[9] J. Krumm, R. Gruen, and D. Delling, "From destination prediction to route prediction," *Journal of Location Based Services*, vol. 7, pp. 98–120, Jun. 2013. DOI: 10.1080/17489725.2013.788228. (visited on 03/09/2020).

[10] F. Tambon, G. Laberge, L. An, *et al.*, "How to certify machine learning based safety-critical systems? a systematic literature review," *Automated Software Engineering*, vol. 29, Apr. 2022. DOI: 10.1007/s10515-022-00337-x. (visited on 08/18/2022).

[11] E. Uhlemann, "Introducing connected vehicles [connected vehicles]," *IEEE Vehicular Technology Magazine*, vol. 10, no. 1, pp. 23–31, 2015. DOI: 10.1109/MVT.2015.2390920. (visited on 04/16/2024).

[12] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors: the Journal of the Human Factors and Ergonomics Society*, vol. 37, pp. 32–64, Mar. 1995. DOI: 10.1518/001872095779049543. [Online]. Available: https://www.researchgate.net/publication/210198492_Endsley_MR_Toward_a_Theory_of_Situation_Awareness_in_Dynamic_Systems_Human_Factors_Journal_371_32-64 (visited on 04/16/2024).

[13] K. Golestan, R. Soua, F. Karray, and M. S. Kamel, "Situation awareness within the context of connected cars: A comprehensive review and recent trends," *Information Fusion*, vol. 29, pp. 68–83, 2016, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2015.08.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253515000743 (visited on 04/16/2024).

[14] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning, second edition* (Adaptive Computation and Machine Learning series). MIT Press, 2018, ISBN: 9780262351362. [Online]. Available: https://books.google.se/books?id=dWB9DwAAQBAJ (visited on 04/16/2024).

[15] S. Raschka, *Model evaluation, model selection, and algorithm selection in machine learning*, 2020. arXiv: 1811.12808 [cs.LG]. (visited on 04/16/2024).

[16] P. Pierce, "Software verification and validation," in *IEEE Technical Applications Conference. Northcon/96. Conference Record*, 1996, pp. 265–268. DOI: 10.1109/NORTHC.1996.564924. (visited on 04/16/2024).

[17] R. G. Sargent, "Validation and verification of simulation models," in *Proceedings of the 24th Conference on Winter Simulation*, ser. WSC '92, Arlington, Virginia, USA: Association for Computing Machinery, 1992, pp. 104–114, ISBN: 0780307984. DOI: 10.1145/167293.167311. [Online]. Available: https://doi.org/10.1145/167293.167311 (visited on 04/16/2024).

[18] J. Henriksson, M. Borg, and C. Englund, "Automotive safety and machine learning," *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, May 2018. DOI: 10.1145/3194085.3194090.

[19] "Gis and network analysis," in *Spatial Analysis and GeoComputation: Selected Essays*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 43–60, ISBN: 978-3-540-35730-8. DOI: 10.1007/3-540-35730-0_4. [Online]. Available: https://doi.org/10.1007/3-540-35730-0_4 (visited on 04/16/2024).

[20] J. Lin and Y. Ban, "Complex network topology of transportation systems," *Transport Reviews*, vol. 33, no. 6, pp. 658–685, 2013. DOI: 10.1080/01441647.

2013.848955. eprint: https://doi.org/10.1080/01441647.2013.848955. [Online]. Available: https://doi.org/10.1080/01441647.2013.848955 (visited on 04/16/2024).

[21] E. Knauss, "Constructive master's thesis work in industry: Guidelines for applying design science research," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, Madrid, ES, 2021, pp. 110–121. DOI: 10.1109/ICSE-SEET52601.2021.00021.

[22] A. Hevner, S. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, pp. 75–105, 2004. DOI: 10.2307/25148625. [Online]. Available: http://wise.vub.ac.be/sites/default/files/thesis_info/design_science.pdf (visited on 04/16/2024).

[23] K.-J. Stol and B. Fitzgerald, "The abc of software engineering research," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 3, 2018, ISSN: 1049-331X. DOI: 10.1145/3241743. [Online]. Available: https://doi.org/10.1145/3241743.

[24] R. Baskerville, "What design science is not," *European Journal of Information Systems*, vol. 17, no. 5, pp. 441–443, 2008. DOI: 10.1057/ejis.2008.45. eprint: https://doi.org/10.1057/ejis.2008.45. [Online]. Available: https://doi.org/10.1057/ejis.2008.45.

[25] A. Hevner and S. Chatterjee, "Design science research in information systems," *Integrated Series in Information Systems*, vol. 22, pp. 9–22, 2010. DOI: 10.1007/978-1-4419-5653-8_2. (visited on 04/16/2024).

[26] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14)*, ser. Article 38, New York, NY, USA: Association for Computing Machinery, 2014, pp. 1–10. DOI: 10.1145/2601248.2601268. (visited on 04/16/2024).

[27] H. J. Vishnukumar, B. Butting, C. Müller, and E. Sax, "Machine learning and deep neural network — artificial intelligence core for lab and real-world test and validation for adas and autonomous vehicles: Ai for efficient and quality test and validation," in *2017 Intelligent Systems Conference (IntelliSys)*, London, UK, 2017, pp. 714–721. DOI: 10.1109/IntelliSys.2017.8324372.

[28] J. E. Stellet, M. Woehrle, T. Brade, A. Poddey, and W. Branz, "Validation of automated driving a structured analysis and survey of approaches," in *13. Uni-DAS eV Workshop Fahrerassistenz und automatisiertes Fahren*, 2020, p. 10.

[29] A. Knauss, J. Schröder, C. Berger, and H. Eriksson, "Paving the roadway for safety of automated vehicles: An empirical study on testing challenges," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1873–1880. DOI: 10.1109/IVS.2017.7995978. (visited on 04/16/2024).

[30] D. Karunakaran, J. S. Berrio, S. Worrall, and E. Nebot, "Challenges of testing highly automated vehicles: A literature review," in *2022 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*, 2022, pp. 1–8. DOI: 10.1109/RASSE54974.2022.9989562.

[31] N. Rajabli, F. Flammini, R. Nardone, and V. Vittorini, "Software verification and validation of safe autonomous cars: A systematic literature review," *IEEE*

Access, vol. 9, pp. 4797–4819, 2021. DOI: 10.1109/ACCESS.2020.3048047. (visited on 04/16/2024).

[32] K. Radlak, M. Szczepankiewicz, T. Jones, and P. Serwa, "Organization of machine learning based product development as per iso 26262 and iso/pas 21448," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2020, pp. 110–119. DOI: 10.1109/PRDC50213.2020.00022. (visited on 04/16/2024).

[33] S. Mohseni, M. Pitale, V. Singh, and Z. Wang, *Practical solutions for machine learning safety in autonomous vehicles*. [Online]. Available: https://arxiv.org/pdf/1912.09630.pdf (visited on 04/16/2024).

[34] C. Sippl, F. Bock, D. Wittmann, H. Altinger, and R. German, "From simulation data to test cases for fully automated driving and adas," in *Testing Software and Systems*, F. Wotawa, M. Nica, and N. Kushik, Eds., Cham: Springer International Publishing, 2016, pp. 191–206, ISBN: 978-3-319-47443-4. (visited on 04/16/2024).

[35] F. Bock, C. Sippl, S. Siegl, and R. German, "Status report on automotive software development," in *Automotive Systems and Software Engineering: State of the Art and Future Trends*, Y. Dajsuren and M. van den Brand, Eds. Cham: Springer International Publishing, 2019, pp. 29–57, ISBN: 978-3-030-12157-0. DOI: 10.1007/978-3-030-12157-0_3. [Online]. Available: https://doi.org/10.1007/978-3-030-12157-0_3 (visited on 04/16/2024).

[36] J. Zhang and J. Li, "Testing and verification of neural-network-based safety-critical control software: A systematic literature review," *Information and Software Technology*, vol. 123, p. 106 296, 2020, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2020.106296. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584920300471 (visited on 04/16/2024).

[37] L. Myllyaho, M. Raatikainen, T. Männistö, T. Mikkonen, and J. K. Nurminen, "Systematic literature review of validation methods for ai systems," *Journal of Systems and Software*, vol. 181, p. 111 050, 2021, ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2021.111050. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221001473 (visited on 04/16/2024).

[38] H.-P. Schöner, "Challenges and approaches for testing of highly automated vehicles," in *Energy Consumption and Autonomous Driving*, J. Langheim, Ed., Cham: Springer International Publishing, 2016, pp. 101–109, ISBN: 978-3-319-19818-7. (visited on 04/16/2024).

[39] X. Li, "A scenario-based development framework for autonomous driving," *arXiv preprint arXiv:2011.01439*, 2020. (visited on 04/16/2024).

[40] M. Wagner and P. Koopman, "A philosophy for developing trust in self-driving cars," in *Road Vehicle Automation 2*, G. Meyer and S. Beiker, Eds., Springer, Cham: Springer International Publishing, 2015, pp. 163–171, ISBN: 978-3-319-19078-5.

[41] M. Kläs and A. M. Vollmer, "Uncertainty in machine learning applications: A practice-driven classification of uncertainty," *Developments in Language The-*

ory, pp. 431–438, Aug. 2018. DOI: 10.1007/978-3-319-99229-7_36. (visited on 04/16/2024).

[42] N. Polyzotis, M. Zinkevich, S. Roy, E. Breck, and S. Whang, "Data validation for machine learning," in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 334–347. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2019/file/928f1160e52192e3e0017fb63ab65391-Paper.pdf (visited on 04/16/2024).

[43] H. B. Braiek and F. Khomh, "On testing machine learning programs," *Journal of Systems and Software*, vol. 164, p. 110 542, 2020, ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2020.110542. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121220300248 (visited on 04/16/2024).

[44] Z. Qi, H. Wang, J. Li, and H. Gao, *Impacts of dirty data: And experimental evaluation*, arXiv.org, Apr. 2021. DOI: 10.48550/arXiv.1803.06071. [Online]. Available: https://arxiv.org/abs//1803.06071 (visited on 04/16/2024).

A

Data set Characteristics

The following appendix section shows graphs of the training data characteristics. These data sets were used to train the ML models that is shown in the result section.

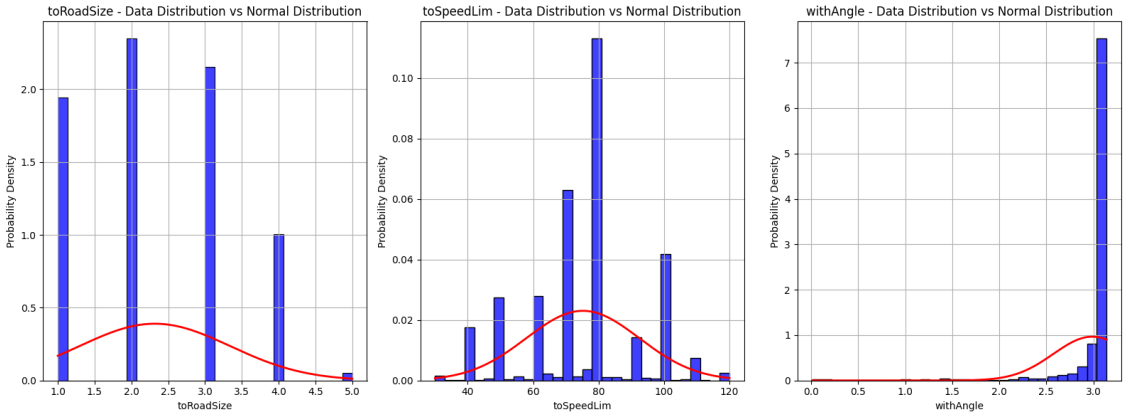

Figure A.1: Training data sampled from a routing database. The red line is the probability density function for the data.

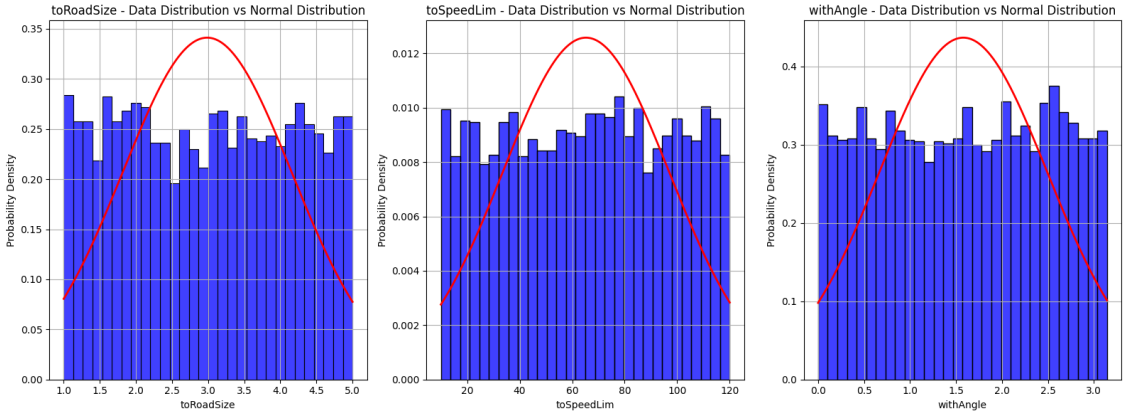

Figure A.2: Randomly sampled training data within the data boundaries. The red line is the probability density function.

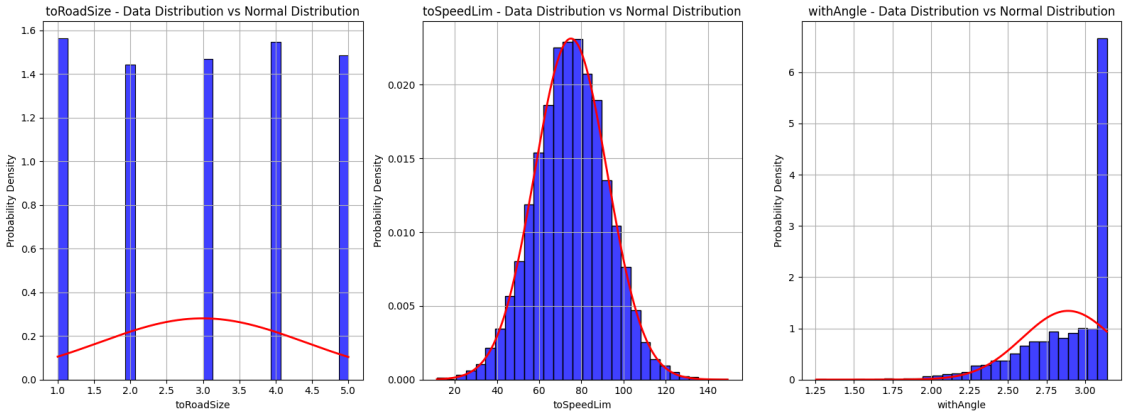

Figure A.3: Training data sampled from a normal distribution with respect to the data boundaries. The red line is the probability density function.

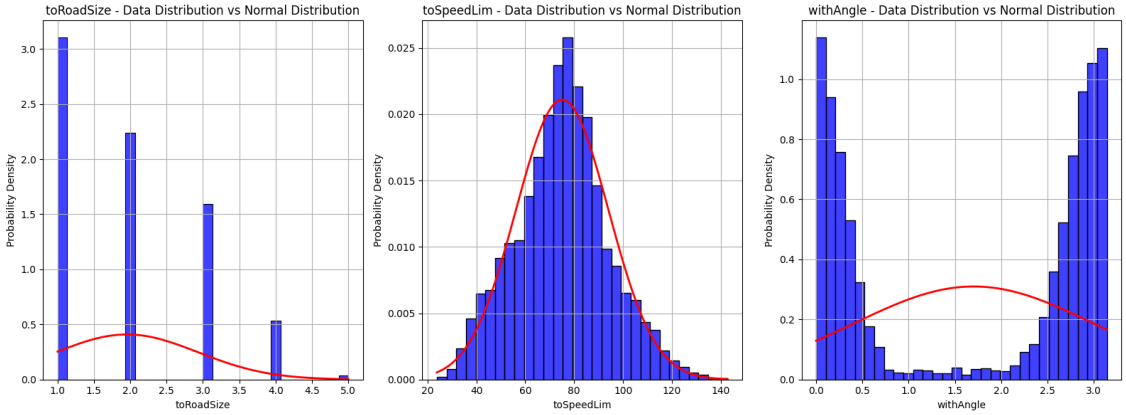

Figure A.4: Training data generated from the routing database with noise. The red line is the probability density function.

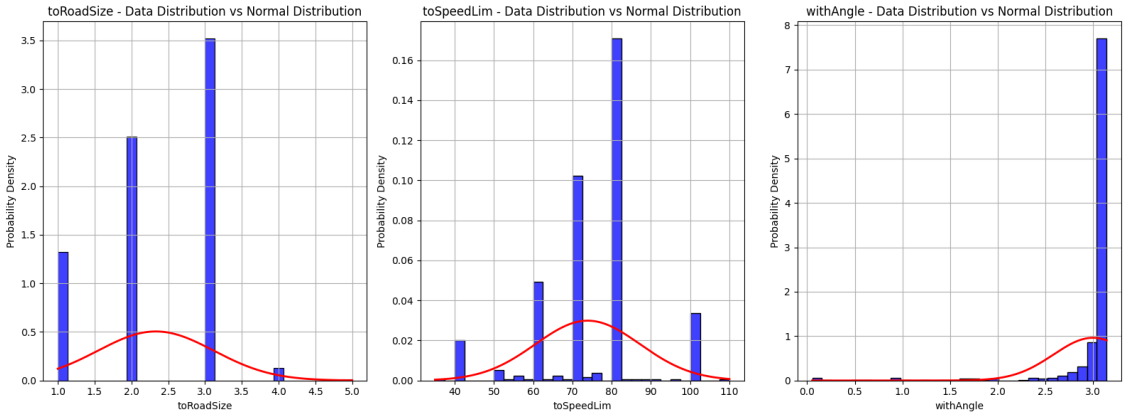

Figure A.5: Validation data characteristics.

B

Prediction Capabilities

Below are graphs on the prediction capabilities of the different data sets. The blue area is where the 97.5% of the prediction is made.

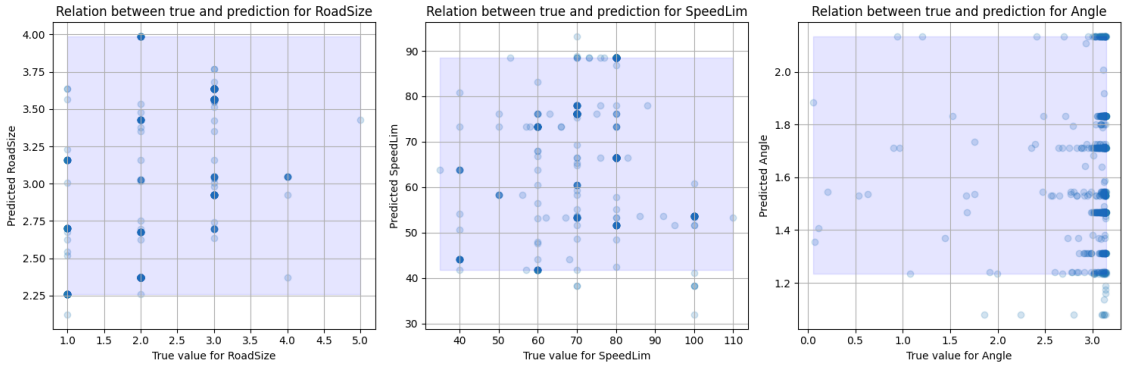

Figure B.1: Prediction capabilities of an ML model that has been trained with randomly distributed data.

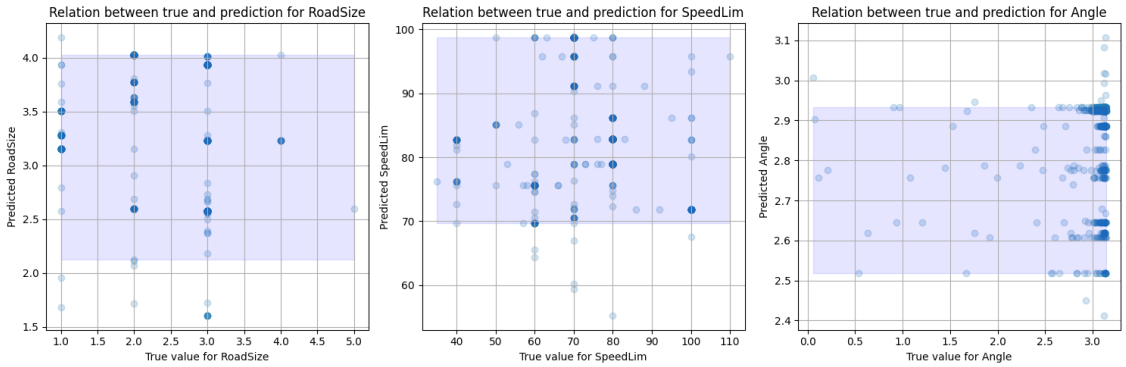

Figure B.2: Prediction capabilities of an ML model that has been trained with normally distributed data.

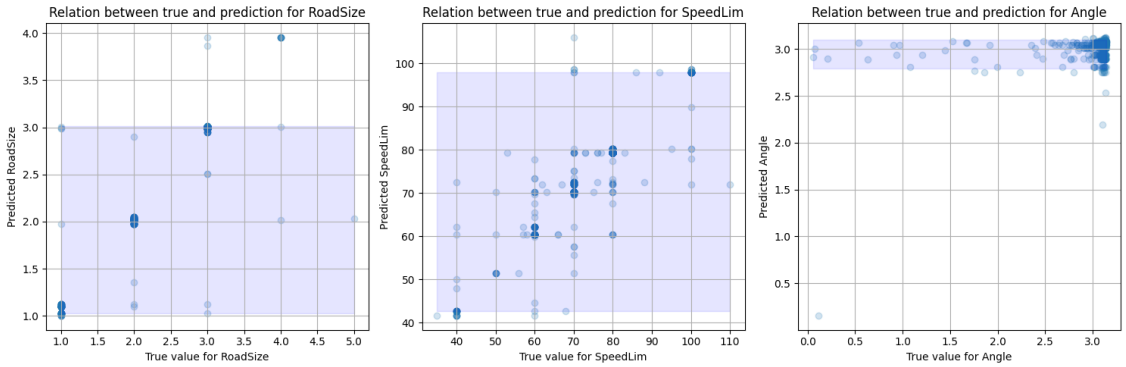

Figure B.3: Prediction capabilities of an ML model that has been trained with router database data.

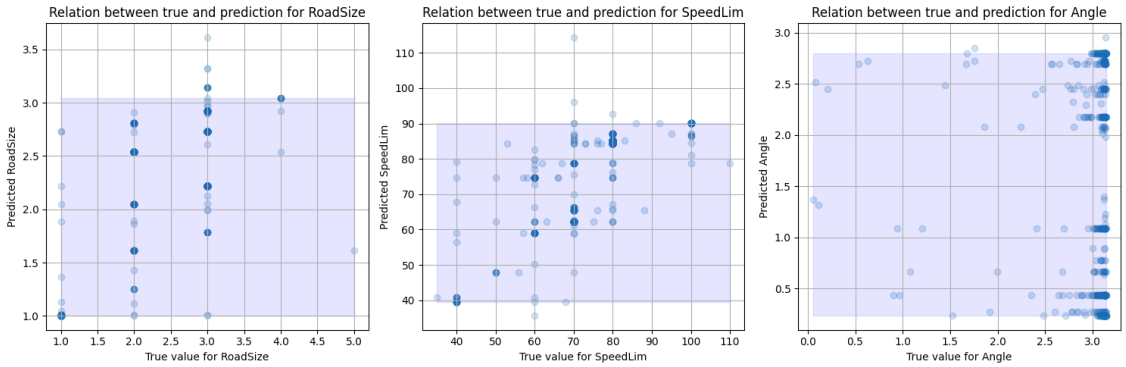

Figure B.4: Prediction capabilities of an ML model that have been trained with router database data, but with added random noise.

C

Validation Schemas

In this part of the appendix are JSON formatted schemas that were used to validate the structure of the data, response, and requests for the ML model.

```
"type": "object",
    "properties": {
      "fromRoadSize":
        {"type": "integer", "minimum": 1, "maximum": 5},
      "toRoadSize":
        {"type": "integer", "minimum": 1, "maximum": 5},
      "fromSpeedLim":
        {"type": "number", "minimum": 10, "maximum": 150},
      "toSpeedLim":
        {"type": "number", "minimum": 10, "maximum": 150},
      "withAngle":
        {"type": "number", "minimum": 0, "maximum": 3.1416}
    }, "required": [
        "fromRoadSize", "toRoadSize",
        "fromSpeedLim", "toSpeedLim", "withAngle"
    ]
```

Figure C.1: JSON schema for validating the data sets statically.

```
 "type": "object",
  "properties": {
    "fromRoadSize": {
      "type": "integer",
      "minimum": 1,
      "maximum": 5
    },
    "fromSpeedLim": {
      "type": "number",
      "minimum": 10,
      "maximum": 150
    }
  }, "required": ["fromRoadSize", "fromSpeedLim"]
```

Figure C.2: JSON schema for validating the request when calling the API service.