Short-Term Traffic Forecasting on Highways Using Neural Networks

A Congestion Pattern-Based Assessment

Koen Simons

Faculty of Civil Engineering

Short-Term Traffic Forecasting on Highways Using Neural Networks

A Congestion Pattern-Based Assessment

by

Koen Simons

to obtain the degree of Master of Science

at the Delft University of Technology

Student Number:	4381483	
Project Duration:	September 8, 2020 – May 31, 2021	
Thesis Committee:	prof. dr. ir. J. W. C. van Lint	TU Delft
	dr. P. K. Krishnakumari	TU Delft
	dr. M. M. de Weerdt	TU Delft
	C. Achterberg MSc	CGI Nederland

Preface

When I started my bachelors in Civil Engineering in Delft seven years ago, freshly graduated from high school, I would not have guessed I would be the person I am right now. Besides providing me with knowledge, studying in Delft shaped me to the to the person I am right now. And writing this master thesis, especially during COVID-19, has been one of the most challenging parts of my study. Having my first course on traffic data from professor Hans van Lint first sparked my interest in the possibilities traffic data has to offer. I feel very lucky to have had the possibility to dive into the fascinating world of artificial intelligence and to further improve my knowledge in this field.

I would like to thank the following people for their guidance while writing this thesis, first of all, to my supervisors from the Delft University of Technology. First of all, I thank Panchamy Krishnakumari for her guidance, valuable insights, and the confidence and motivation you gave me while writing this thesis. Secondly, I would like to thank professor Hans van Lint for his enthusiasm on this topic and my research, and the interesting discussions we had during our meetings. Lastly, I would like to express my gratitute to professor Mathijs de Weerdt for his critical comments and valuable suggestions for structuring this thesis.

However, this thesis would not have been possible without the guidance of CGI. Thank you Chantal for your sharpness and attention to detail, the lunch walks eating all possible sandwiches Delft has to offer and all the energy and time you had for me. Henk, I am very grateful for your enthusiasm, the attention you have for personal development and for welcoming me to iAMLAB. I would also like to thank Laurens Lapré for the interesting discussions we had.

Lastly, I could not have gone without the support and encouragement from my family and friends. First and foremost, I would like to thank my parents, my biggest supporters, who never doubted my abilities to succeed in my studies in Delft. Lara and Carlijn, thanks for the nice lunch walks, shared frustration and productive hours while studying. Sanne, thank you for being there throughout my studies in Delft, which we (almost) did synchronously. Special thanks to my roommates for cheering me up, providing me with food and patiently listen to all my complaints. Finally, thanks to my boyfriend Rob, for providing me with love and kindness and making the last weekends off a lot of fun.

Koen Simons
Delft, April 18th, 2021

Research Summary - TU Delft

As traffic demands are ever increasing and building new infrastructure poses challenges in densely populated areas, it is important to optimally utilise existing infrastructure. Short-term traffic forecasting can help with this task, as its predictions can help to prevent congestion by rerouting vehicles. Recently, neural networks developed for traffic forecasting have lead to an unprecedented accuracy, but deploying these on large scale can be difficult as the resulting models likely overfit to the highway they were trained on. This thesis therefore performs an in-depth assessment of the accuracy of neural networks traffic speed predictions on highway stretches containing different types of congestion patterns. By relating the results back to traffic flow theory, these results can be put into perspective.

Based on traffic flow theory, it was concluded that predicting traffic speed is more relevant than flow for the detection of congestion. However, both flow and speed are needed as input data, since both contain important information for traffic predictions. In general, the traffic prediction process could be split up in correctly predicting the (i) *emergence* of congestion due to traffic breakdown and (ii) the *propagation* of the upstream and downstream congestion head. Although the propagation can be predicted, the stochastic nature of traffic breakdown makes it difficult to predict emergence.

A large variety of design choices where identified when designing a neural network architecture. For a a gated recurrent unit (GRU) recurrent neural network (RNN), these can be categorised in (a) the temporal structure, (b) spatiotemporal data, (c) model architecture, (d) input/output variables and (e) data source. Promising design choices were presented and gaps regarding (a), (c) and (d) were identified. For the temporal structure, a good substantiation of the prediction horizon and input sequence length was missing. Little substantiation is given on the model architecture, making it unclear what the benefits of this new architecture are besides having a higher accuracy. No direct comparison was made on the effect of including extra traffic-related data, especially in case many variables were used.

Different neural network architectures were formulated and trained on predicting traffic speed 20 minutes in the future. Both a one-shot and a sequence-to-sequence GRU RNN model architecture were evaluated, while changing the number of hidden layers, units per layer and loss function. The effect of adding flow data to the inputs has been assessed as well as the sensitivity of the predictions to the input sequence length and prediction horizon. Moreover, new spatiotemporal error metrics were proposed resulting predictions were assessed on different spatiotemporal levels. To retrieve data, several roads were simulated in AIMSUN using different demand patterns, leading to a mix of different congestion patterns mostly consisting of moving synchronised patterns and general patterns.

When comparing the results from different congestion patterns, large differences in error could be observed. In general, it was found that a moving synchronised pattern (MSP) can be predicted with a reasonable accuracy and multiple shockwave speeds could be predicted correctly. This especially is the case if congestion propagates over a period of time longer than the prediction horizon, due to mispredictions during emergence. Predicting a general pattern (GP) containing stop-and-go waves is much more difficult, therefore, predicting the emergence and propagation of stop-and-go waves was only possible at much shorter prediction horizons of 4 to 10 minutes.

Assessing the resulting errors spatially and temporally using custom error metrics, theoretical assumptions could be confirmed. Over time, errors were higher during periods where congestion occurs and spiked during emergence and times where changes in traffic demand resulted in

changes in shockwave speed. Over space, errors were higher at bottleneck locations and the upstream jam head locations, since traffic conditions can quickly change at these locations.

Based on the highest accuracy neural network structures, the following conclusions could be drawn. Sequence-to-sequence models are more suitable compared to one-shot models, as the errors were lower and the resulting congestion patterns better. A mean absolute error (MAE) loss function lead to a higher accuracy and more stable results compared to a mean squared error (MSE) loss function. As traffic speeds can fluctuate quite quickly and strongly, the relatively high errors caused by a MSE loss function can result in unstable training. The optimal hyperparameter configuration fluctuated per dataset, although mostly architectures of moderate depth performed best.

Using both flow and speed data as input lead to mixed results regarding the accuracy. On less complex MSPs, adding flow data resulted in an increase in prediction accuracy, which is in line with traffic flow theory. However, adding flow data lead to a accuracy decrease for more complex GPs when using a sequence-to-sequence architecture. As the input and output parameters have to be equal in an autoregressive sequence-to-sequence model, this is likely caused by the complex traffic flow pattern which has to be predicted as well.

To conclude, it was proven that relating the results of a traffic forecasting neural network to traffic flow theory can provide valuable insights in the strengths and limitations of its application. By assessing the individual prediction results of several congestion patterns, it was found the accuracy can differ greatly per congestion pattern. Reflecting these results back to traffic flow theory provides insight in the general limitations of traffic forecasting using neural networks, namely the prediction of the emergence of congestion. The proposed custom error metrics were proven to be a good tool to assess the accuracy of newly proposed neural networks.

Management Summary - CGI Nederland

In the Netherlands, congestion is prevalent om many roads, especially in built-up areas. However, expanding the current road infrastructure often is not possible, due to high costs and limited space available. Short-term traffic forecasting can be a helpful tool to increase capacity by optimally using existing infrastructure. Many efforts have been done in order to successfully forecast traffic, such as the Velsertunnel project, in which CGI and TU Delft participated, and Praktijkproef Amsterdam, which had mixed results. Within academia, neural networks developed for traffic forecasting have lead to an unprecedented accuracy, however, there are questions on the practical use of these models.

In this thesis, the accuracy of neural networks for short-term traffic forecasting is evaluated for different congestion patterns, i.e. types of traffic, in order to get a in-depth view of the strengths and weaknesses of neural networks. Short-term in this context means a prediction horizon of up to 30 minutes. By assessing the model error over both space and time, more insight is retrieved on the locations and times where accuracy might decrease. Doing so for different congestion patterns provides more insight on its ability to predict traffic.

To structure this management summary, important conclusions and recommendations regarding three different topics. First, general conclusions will be presented regarding the use of a recurrent neural network (RNN) for spatiotemporal forecasting. Secondly, conclusions and recommendations regarding traffic forecasting will be given. Lastly, it will focus on recommendations on the practical implementation of traffic forecasting neural networks.

RNNs for Spatiotemporal Forecasting

As the methodology presented in this master thesis is applicable to to many different spatiotemporal prediction problems, some general comments on using neural networks will be given first. An important factor determining the success of using a neural network in general was found to be the input and output data used. A neural network simply is a function approximator which learns a very complex nonlinear function between input and output data. However, if there is no correlation or causation present between the input and output data, even the most complex neural network is not able to capture this non-existent function. Even if a model in this case can reach a high accuracy, it is likely to suffer from *overfitting*. Overfitting means that the model achieved high accuracy on the training and test data, but the learnt input-output relations are not present in unseen data. This makes the model not generalisable and not useful in practice. Therefore, a first good step is to consult with experts or dive into the theory of a certain problem. First, one has to establish whether an input-output relation is likely to be present. Secondly, one has to pick the input features carefully and in accordance with theory. By doing many design iterations, starting off with simple neural networks of which the complexity increases per cycle, and comparing the models to a solid baseline, one can quickly get an idea on whether the problem can be solved using neural networks.

Secondly, it was found that the gated recurrent unit (GRU) recurrent neural network (RNN) was very suitable for spatiotemporal prediction problems. Especially a sequence-to-sequence architecture was found to result in a higher accuracy. Depending on the specifics of the prediction problem, it is important to correctly choose the loss function, as this can have implications on both the results and the stability of the training process.

Conclusions Regarding Traffic Forecasting

When assessing literature on neural network architectures for traffic forecasting, very complex architectures are sometimes proposed. However, when looking at traffic flow theory, these complex neural network architectures cannot always be justified. Based on assessing both traffic flow theory as previously created traffic forecasting neural networks, four *data* requirements were identified which will likely increase prediction accuracy and the usefulness of the predictions. First of all, it is more useful to predict traffic speed compared to traffic flow, as the former is a better predictor for congestion compared to the latter. Travel time, which is a very relevant metric for taking traffic measures, can also be easier expressed when traffic speed is predicted. Using both flow and speed as input data is likely to increase model accuracy, although this depends on the congestion patterns in the data and the neural network architecture chosen. Secondly, using short-term traffic flow and speed data as input is recommended over historical traffic data. Short-term traffic data contains more information from a traffic flow theory point of view, which describes that previously observed flow and speed data can be used to correctly predict congestion patterns. Thirdly, using spatiotemporal input data, i.e. data from different adjacent highway segments over time, leads to better results compared to data only over space or time. However, it is important that the assessed highway segment has sufficient length, so enough information is present so enough information is present to forecast traffic given the set prediction horizon. Lastly, he sensitivity analysis which has been done showed that it is important to choose the prediction horizon in accorance with the congestion patterns present in the dataset. Predicting highly unstable traffic conditions might only be possible in case a lower prediction horizon is given. For the input horizon, it is important this is chosen so that causality can be expected between the input horizon and the prediction results.

Based on the results of the experiments done, a gated recurrent unit (GRU) recurrent neural network (RNN) seems to be a good benchmark for traffic speed prediction, as most congestion patterns are accurately reproduced. Some recommendations can also be done on correctly implementing this neural network structure. Overall, using a sequence-to-sequence model leads to better results compared to a one-shot model, as error metrics were lower and the congestion patterns were more accurately reproduced. Using a mean absolute error (MAE) loss function also lead to better results and more stable training compared to a mean squared error (MSE) loss function.

The model accuracy was found to strongly fluctuate with the type of congestion patterns present in the dataset. On a moving synchronised pattern (MSP), accuracy was high and the predictions could correctly replicate the congestion patterns. However, when a general pattern (GP) with stop-and-go waves occurred, traffic breakdown, i.e. the moment at which congestion occurs, could not be predicted for a prediction horizon of 20 minutes. In general, the neural network could accurately predict the propagation of congestion when it had occurred, but had trouble predicting congestion emergence, i.e. the moment of traffic breakdown. Prediction accuracy is therefore likely to drop on roads where congestion is caused by accidents or traffic conditions are highly unstable, causing stop-and-go waves to occur.

Recommendations for Practical Implementation
From this thesis can be concluded that neural networks have the potential to 'learn' traffic, as congestion patterns can be correctly replicated in numerous scenarios. Compared to the baseline of not predicting traffic patterns, the models proved to have added value for short-term traffic forecasting on highway segments. Besides, the newly proposed spatiotemporal error metrics can help potential clients to get more insights in the model accuracy. This can both guide the interpretation of the results as well as increase the confidence in the model, as users get more insight in what the model can and cannot predict.

To be able to fully assess the model accuracy, it is important to research the following elements. First, it would be important to analyse a model which was trained and tested on empirical loop detector data. Using this, the results retrieved using simulation data can be validated. Secondly, a thorough assessment is needed of the road stretch on which the neural network will be trained. If a large part of congestion is caused by stop-and-go waves or accidents, a neural network might not be suitable for predicting traffic flows. Lastly, it is important to check whether traffic supply and demand conditions remain stable over time. If changes in travel behaviour or the road network result in changes to traffic conditions on a road, retraining might be required.

Contents

Acronyms

ARIMA	Auto Regressive Integrated Moving Average.
ASM	adaptive smoothing method.
FFNN	feed-forward neural network.
GP	general pattern.
GRU	gated recurrent unit.
HA	historical average.
IQR	interquartile range.
ITS	intelligent transportation system.
LSP	localised synchronised pattern.
LSTM	long short-term memory.
MAD	median absolute deviation.
MAE	mean absolute error.
MAPE	mean absolute percentage error.
MSE	mean squared error.
MSP	moving synchronised pattern.
ReLU	rectified linear unit.
RMSE	root mean squared error.
RNN	recurrent neural network.
RW	random walk.
SP	synchronised pattern.

List of Figures

List of Tables

1 Introduction

Traffic demand is ever growing, putting pressure on the accessibility of many cities all around the world. However, in many built-up areas, increasing supply by building or upgrading roads or other transportation networks is becoming more difficult due to space limitations and environmental concerns. As a result, the resulting traffic congestion can harm the economy - due to lost hours and extra fuel consumption - and cause extra pollution. Therefore, it is important to develop methods which help to optimally use the existing infrastructure, which can then be integrated into an intelligent transportation system (ITS) to enforce measures reducing the amount of disruptions on roads. A key feature of an ITS is short-term traffic forecasting, as high-quality predictions of flow and/or speed can aid taking real-time control measures in a timely way to reduce the impact of a traffic disruption [11].

1.1. Problem Analysis

Many different prediction methods have been proposed or developed with the aim of trying to predict traffic on road networks or road stretches. The methods and models used are categorised in Figure 1.1, based on a categorisation developed by Van Lint and Van Hinsbergen [12]. Naive methods, such as historical averages or instantaneous travel times, make no assumptions and deduce no model structure and parameters from data. They have low computational effort and are easy to use and implement, but also have low accuracy. Secondly, traffic model-based methods rely on theoretical or physical assumptions regarding traffic propagation over time. They can be subdivided in traffic simulation models, which explicitly model traffic conditions over a road network, or analytical methods, which only make use of the underlying functions. Their parameters, such as link capacities or fundamental diagrams, and inputs, e.g. origin-desination matrices, have to be calibrated within a feasibility region for the model to be in accordance with the data. They are very useful for understanding traffic flow and can be used to assess hypothetical scenarios [13], but calibration is time consuming. The third category consists of generic (mathematical) prediction methods, with flexible structures and adjustable parameters instead of traffic-based assumptions. Within this category, a large amount of methods are possible, which can be subdivided into parametric and non-parametric methods [13]. Parametric methods consist of time series analysis approaches, mostly statistical models such as Auto Regressive Integrated Moving Average (ARIMA), which have fixed, a-priori determined model structures for which only parameter values have to be determined. Non-parametric models mostly consist of artificial intelligence-based approaches, such as support vector regression, clustering and (deep) neural networks, which both 'learn' their model structure as well as parameter values.

There is quite a big difference between the parametric and non-parametric modelling culture, as discussed by Breiman [14]. For the development of parametric statistical models, the goal is to create well-interpretable, explanatory models which comply to statistical tests and have statistically significant parameters. This makes it easier to implement them and analyse their outcomes. A

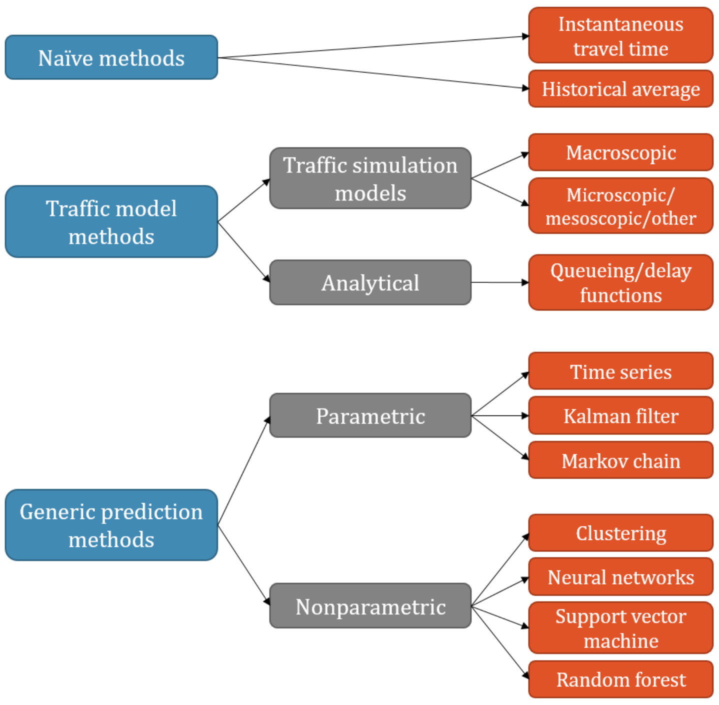


Figure 1.1: Classification of prediction techniques

Table 1.1: Summary of differences between parametric and non-parametric models. Based on [10].

Function	Statistical models	Neural networks
Philosophy	Inference and estimation (data generated by stochastic model)	Implementation (data from 'black box')
Goal	Offer (self-explanatory) models providing insights in data	Provide efficient representation of data properties with good prediction
Learning process	Model assumed a-priori, including restrictions and hypotheses	Model generated through learning process, 'data will learn the model'.

downside is that the model accuracy is often neglected since the a-priori assumed model structure often oversimplifies reality. Within the non-parametric modelling culture, the goal is to have the lowest prediction error 'at all costs'. Due to their complex model structure, they are capable of capturing the complex nonlinear nature of traffic, making their accuracy higher compared to parametric methods. However, non-parametric models are 'black boxes' which often provide little insight into processes, leaving their low error to be unexplained, which might be due to biased estimation and overly complex models. Their differences are summarised in Table 1.1.

1.2. Research Gap

This research focuses on traffic flow prediction on freeway stretches through generic prediction methods, specifically through non-parametric artificial neural networks. In the last years, quite some research has been conducted in improving these neural networks proposing a large variety of neural network architectures. These highly complex newly-developed models show unprecedented prediction accuracy compared to other (non)parametric statistical methods. However, Karlaftis and Vlahogianni [10] state that over the years, the tendency to develop complex nonlinear and high dimensional models is made possible due to an increase in computing power but not necessarily justified by logic or fundamental research needs. As the main focus has been put on prediction accuracy, little questions are raised on topics which can form a broader-view 'model performance' such as simplicity and suitability.

Three different problems can be identified regarding the application of prediction methods within a traffic forecasting context, namely the interpretability of these models, the generalisability of these models over roads and whether the resulting predictions are in line with current knowledge on traffic flow theory.

It was found that all neural networks architectures developed for traffic forecasting are optimised and tested on only one specific road network. Some researchers have acknowledged that most neural networks used for traffic prediction are site-specific, since most model structures are optimised for and trained on data from one specific location to increase accuracy further [15]. The focus of creating more complex neural networks can result in a complex, overestimated structure leading to an over-specified model with limited to no generalisation power [10]. Researchers tend to focus on developing deeper and more complex models, which are also easier to overfit [16] and therefore hard to generalise. Overly complex models only applicable to specific road stretches would not be appropriate in enhancing an ITS [17] as, in combination with low computational efficiency, it might require too much skill and manpower to roll out these models in practice. These complex models have many tunable hyperparameters and require a lot of data and computational power in order to be trained and validated. The increase in accuracy complex models could possibly bring might be of limited added value when deployed in practice.

Therefore, it would be interesting to create neural networks which could be applied on multiple roads. This would significantly decrease the amount of time and effort needed to develop a neural network for traffic forecasting on a specific road. This would be similar to the field of transfer learning, where a neural network trained to perform a task in a certain domain, such as natural language processing, can be reused to perform a different task within that domain, increasing the generalisability of this neural network. In this case, the model should 'learn' the general concept of traffic forecasting, and should be able to apply this on several different roads. However, there are some ifs and buts which make it difficult to accomplish using a similar neural network architecture on different roads. Three of them will be explained below.

First of all, since differences in traffic demand, driver behaviour and road layout can lead to big differences in congestion [18], the neural network should either be able to make distinctions here or the different roads should be sufficiently similar with respect to these topics. Secondly, as neural network architectures are not only optimised on but also built for a specific road or network segment [10], one should research how input data from different roads or road networks, having a different dimensionality. can be captured by a similar neural network structure. Lastly, the input-output relations the neural network is fitted on should be **generalisable**, as the network should 'learn' general relations between traffic variables instead of correlations specific to one road location only.

However, assessing whether a a neural network has 'learnt traffic' is difficult as they are black boxes [14]. Although most researchers put a lot of effort in comparing the accuracy of their neural network to others, they do so by comparing general error metrics instead of analysing the resulting predictions based on a practical point of view, i.e. through the eyes of the traffic engineers which might use these models. Therefore, it is unknown which congestion patterns the model can predict, what its predictions look like and what the practical implications of this higher accuracy is. Ideally, a neural network should extract correlations and learn 'patterns' which are in line with traffic flow theory, as this would mean these relations will also be valid on other roads. However, due to the black-box nature of these neural networks and the lack of a qualitative assessment of the resulting predictions, it remains unclear whether it does so.

1.3. Research Objective and Research Question

We could conclude that little is known on the actual predictions a neural network outputs, as the main focus of researchers is increasing accuracy by decreasing the value of general aggregated error metrics. However, this gives a quite one-dimensional view of the quality, as no insight is given on the practical implications of this accuracy on the resulting predictions. Therefore, little is known on what the model actually learns and to which extent models are generalisable. Besides focusing on accuracy only, it would also be valuable to research what congestion patterns are possible to predict and which boundary conditions arise from traffic flow theory in order to have a well prediction. Therefore, it would be important to connect traffic flow theory with theory regarding the functioning of neural networks to get more insight in whether it is possible to bring the two closer together. A first step in more deeply assessing the accuracy of traffic predicting neural networks would be to look at their errors over space and time. By doing this for different traffic situations, more insight can be retrieved on how predictable traffic situations are. Hence, the *objective* of this research is to assess the predictive accuracy on different spatiotemporal levels. As road type, a highway section is chosen as most prediction papers focus on highways (due to its relative simplicity compared to urban road networks) and the absence of a complex network structure does not introduce the need for incorporating graph structures in the neural network.

Based on this research objective, the following research question could be formulated:

What is the accuracy, assessed on different spatiotemporal levels, of neural network-based short-term traffic speed forecasting models on highways for different congestion patterns?

This research question can be divided in the following subquestions.

1. What different congestion patterns can be formed on highways and to what extent can these be predicted correctly using macroscopic traffic variables?
2. Which design choices can be made when applying neural networks for short-term traffic forecasting regarding the input and output data as well as the neural network architecture?
3. What is the effect of using both traffic flow and speed as input data on the accuracy of the predictions?
4. In which way can metrics assessing the prediction error over space and/or time provide more insights in the model accuracy for the different congestion patterns present on a road?
5. What is the effect of different neural network hyperparameters, neural network architectures, input sequence lengths and prediction horizons on the resulting model accuracy?

1.4. Thesis Structure

In order to answer the aforementioned research questions, the thesis is structured as follows, which is also schematically depicted in Figure 1.2. In chapter 2, an overview is given on macroscopic traffic flow theory on highways, assessing congestion patterns by analysing both traffic flow theory as well as theory on empirically observed congestion patterns. chapter 3 focuses on neural network theory by giving a brief overview of the theoretical foundations, elaborating on time series prediction and giving a literature analysis on the design choices one faces when designing a traffic forecasting neural network. A data analysis is given in chapter 4, analysing flow and speed patterns as well as spatial and temporal correlations using empirical loop detector data. In chapter 5, the research methodology is elaborated and the used neural network architectures will be presented. chapter 6 considers the experimental setup of the several datasets which have been created using the microscopic traffic simulation software AIMSUN. The results can be found in chapter 7, which will be discussed in chapter 8. Since the research objective and research question ultimately can be used to create neural networks which are applicable to multiple roads, thoughts and challenges on this topic will be presented separately in chapter 9, before the final conclusion, as several ideas and insights regarding this concept were developed during research. To conclude, conclusions and recommendations will be presented in chapter 10.

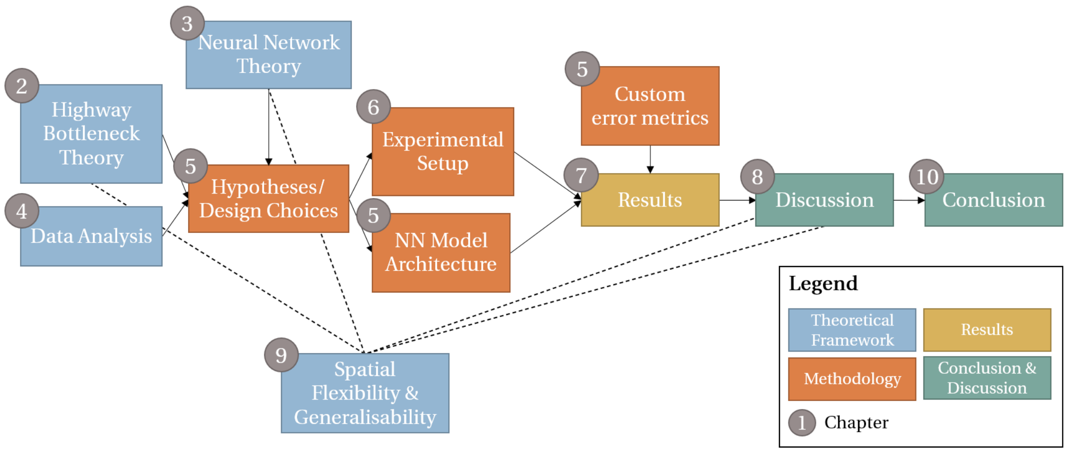


Figure 1.2: Structure of the report.

2 Macroscopic Traffic Flow Theory on Highways

Since the introduction of the fundamental diagram by Greenshields *et al.* [19], many research has been done on traffic flow theory, ranging from urban traffic flow to traffic flow with highways. Over time, complex theories have been created, borrowing concepts from thermodynamics and fluid mechanics, which can be used to classify and explain the traffic patterns forming due to traffic supply and demand. However, in most papers and researches regarding traffic prediction, the authors mainly focus on explaining the model structure they used, the data types they selected and which data preparation techniques were used. Often, an exploratory data analysis is used to find correlations between variables and the results are analysed using statistical methods. However, one can imagine that traffic flow theory can be used to assess which congestion patterns can be captured and which not and what the limits of a prediction method can be from a traffic point of view. Or when from traffic flow theory, one can assume there is a clear (spatiotemporal) relation between traffic variables, it might be valuable to use these for the traffic prediction model. Since a part of this research is on connecting traffic flow theory with neural networks, it is important to analyze literature on traffic flow theory to get an idea of what the model should be able to base its predictions on.

Since traffic flow theory is a very broad field with many different levels and focus points, let us first start with demarcating the topic. First, only traffic flow theory on highways is assessed, as traffic conditions on highways are less complex compared to urban conditions since there are limited access and egress points, no at-grade intersections and relative homogeneity of speeds and vehicle types. Secondly, there are two different levels at which one can describe traffic, namely the **microscopic** and the **macroscopic** level [1]. Within a microscopic level, all vehicles are described individually: by modelling each vehicle and its interactions with other vehicles separately, traffic on a road (network) can be modelled. Therefore, the model can include vehicle dynamics, car following models, driver characteristics, traffic signal plans, etcetera. On a macroscopic level, one does not describe the movements of individual vehicles on a road but rather focus on aggregated variables which describe traffic per point in space and time.

Both approaches have different strengths and weaknesses, making them suitable for different applications. The microscopic level is very good for detailed traffic flow modelling on a local level and gives much information on driving behaviour. However, the large amount of data needed for analyses on this level makes it unsuitable to analyse actual road traffic on this level. For traffic forecasting, we are not interested in detailed traffic flow characteristics, but in how average speeds and flows develop over time. Therefore, in this chapter, we will focus on traffic flow theory related to this macroscopic level.

This chapter mainly focuses on the theory of highway bottlenecks, traffic breakdown and congestion patterns. Let us first focus on defining these three terms, as they will be used quite a lot. A highway

bottleneck is a location at which traffic demand exceeds supply, therefore inducing congestion. Most of the time, a bottleneck is situated at a location where road capacity is lower. This can be of temporary nature, such as bad weather conditions, accidents or road works, due to its road design, as a lane merge, sharp curves or road gradients decrease capacity, or because of unstable traffic due to lane changing [4]. **Traffic breakdown** is the transition of traffic from a free-flow phase to a congested phase. There are many different opinions on the exact nature of this congested phase, but in general, it is characterised by traffic having a lower speed and (often) higher flow compared to free-flow conditions. **Congestion patterns** are the speed and flow conditions over space and time during congestion which provide information on how the congestion propagates over a road over time and what the traffic conditions are within this congestion. There are large differences in congestion patterns, due to differences in supply, determined by combinations of traffic in- and outflows at a road, and demand, governed mainly by road design.

Correctly predicting a congestion pattern can roughly be split into two different parts, both having their difficulties, namely:

1. **emergence**, which is predicting the point in time when traffic breaks down, causing congestion, and

2. **propagation**, which is a correct prediction of the development and propagation of the congestion pattern over space and time resulting from this breakdown.

Therefore, this chapter is structured as follows. First, important macroscopic traffic variables and relationships between them and some basic traffic flow theory them will be discussed, introducing the fundamental diagram. Secondly, we will focus on the propagation aspect, diving in both the theory of congestion patterns as well as the empirical patterns found. Thirdly, the emergence part will be discussed, discussing some of the theory behind and factors influencing the traffic breakdown process. To conclude, the results of the *propagation* and *emergence* parts will be used to comment on the predictability of congestion.

2.1. Macroscopic Traffic Variables and Their Relationships

The three defining variables that describe traffic phases on a macroscopic level are flow, average speed and density. **Traffic flow** (q), usually expressed in the unit vehicles per hour (veh/h), defines the number of vehicles that traverse a road section within a certain time window. **Average speed** (u), expressed in the unit km/h, is the average speed of vehicles on a road section within a certain time window. **Density** (k), expressed in the unit veh/km, is the number of vehicles *on* a road (section) *at* a certain point of time. Where flow aggregates vehicles over time, density does so over space.

These three variables together shape the fundamental relationship within traffic, which is:

$$q = ku \tag{2.1}$$

Knowing two of the three variables makes it possible to estimate the third one, although this estimation is only accurate during stationary and homogeneous traffic conditions. Traffic **stationarity** means that these traffic variables are constant over time, which means they can only change at points in space. Traffic **homogeneity** means that the traffic variables are constant over space, but changes over time are possible. Stationary traffic conditions are quite common as most changes in traffic phases are due to changes on a spatial level, for example at on- or off-ramp locations where traffic flow changes or at locations with different speed limits. Homogeneous traffic conditions are much rarer, as they require traffic conditions over a whole road section to change instantaneously at a certain point in time, such as dynamically changing speed limits.

Within traffic flow theory, it is assumed that there is a clear relationship between the three macroscopic traffic variables mentioned before, which can be drawn as a single line. Greenshields *et al.* [19] was the first to capture the relationship between speed and flow in a fundamental diagram, although any two of the three variables will do. Theoretically, under stationary and homogeneous traffic conditions, observed combinations of flow, speed and density follow a value on the fundamental diagram. Besides, one can also use it to derive other properties of the traffic phases on a road, such as the road capacity, the free-flow speed, the density during a jam, the propagation of so-called stop-and-go waves and the capacity drop after congestion [20]. Since it contains this much information and it makes it possible to derive many traffic flow phenomena from it, it is widely used when modelling and controlling traffic flow. A (triangular) fundamental diagram with its most imporant parameters can be seen in Figure 2.1

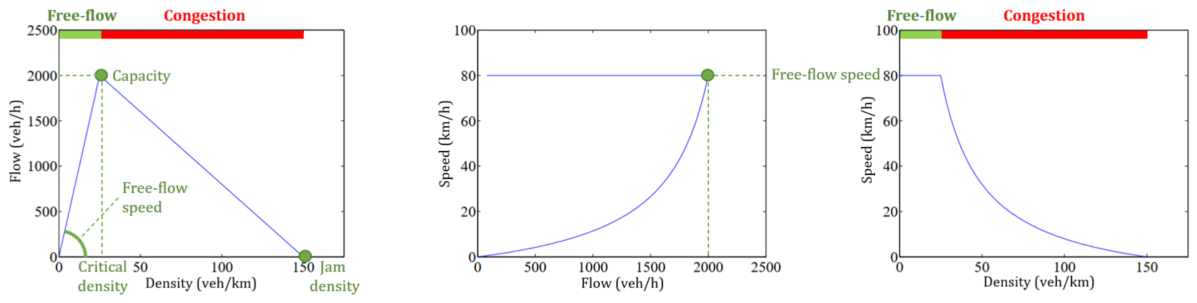


Figure 2.1: Triangular Fundamental Diagram with important traffic parameters (adapted from [1])

Most experts agree that traffic conditions can be subdivided into two phases, free-flow or congestion, both having different properties and combinations of flow, speed and density. In the fundamental diagram, both phases are represented by two branches: depending on its phase, the traffic variables change according to either the free-flow or the congested branch. The ranges of both branches are color coded in Figure 2.1. In theory, if one knows the phase and one of the traffic variables, it is possible to derive the other two using the fundamental diagram. Loosely, the phases can be described as follows. In **free-flow**, the average speed is equal to the free-flow speed and no congestion is experienced as an increase in flow only results in an increase in density and to no (or a limited) decrease in speed. However, once traffic flow exceeds capacity, a traffic breakdown occurs and traffic gets stuck in the **congestion**, characterised by a low speed, a low flow and a high density [21].

Although in theory, the fundamental relationship and fundamental diagram describe traffic quite well, there are some phenomena which cannot be described by it. Let us recall that the fundamental relation only holds when traffic is stationary and homogeneous. In practice, traffic conditions are never fully stationary, as traffic inflow and outflow of a highway can fluctuate greatly over time. Besides, local and temporal disturbances in traffic can often temporarily lead to a violation of constraints. Although on average, traffic conditions might look homogeneous, large differences in speed and vehicle dynamics can cause perturbations on a microscopic scale which eventually can become visible on a larger, macroscopic scale. This means that when calculating vehicle density from flow and speed data, it is important to carefully interpret these values and assess whether it is possible stationarity and/or homogeneity conditions are violated.

Besides, although the fundamental diagram is a good approximation of reality and gives good results in traffic models, observed values can be significantly scattered around the fundamental diagram. In free-flow, the errors are relatively small but when in congestion, actual measurements can deviate greatly from the congested branch line [4]. According to Treiber *et al.* [21], this can be due to measurement errors, traffic flow not being at equilibrium or violation of stationarity and the presence of multiple vehicle types (as motorcycles, cars and trucks each have different dynamics). However, Kerner [22] argues the 'empirical fundamental diagram' is different from the fundamental diagram and the large scatter can be explained by a different phase, which will be elaborated in subsection 2.2.2.

2.2. Theory on Highway Bottleneck Congestion Patterns

After introducing these traffic variables, their meaning and interactions, let us use this to assess traffic conditions for highway bottlenecks. There are many different bottlenecks possible. On the one hand, there are many types of location at which traffic can break down, such as on-ramps, lane merges or even off-ramps, on the other hand, traffic can break down due to critical traffic flow and distracted drivers. A lot of research has been done on finding theories which describe the underlying processes resulting in the emergence and development of a traffic breakdown at highway bottlenecks, and translating them into models which accurately model traffic flow and speed. Although neural networks only detect relations within the input-output data and do not, having insight in these processes can help selecting the right parameters and network structure which helps capturing these processes. Besides looking at theories used for traffic modelling, explaining congestion patterns present in empirical data is also relevant, since a neural network is trained in a data-driven way.

This section is therefore structured as follows. First, a theoretical overview of different kinds of congestion patterns for fixed bottlenecks will be presented, focusing on how they would emerge in a traffic model. Secondly, empirical congestion patterns will be assessed and discrepancies between theory and practice will be discussed, as most neural networks are trained on empirical data.

2.2.1. Theoretical Congestion Patterns

Most traffic models used to model flow and speed conditions near highway bottlenecks in a macroscopic way are based on the fundamental diagram. A basic theory that explains how most models use the fundamental diagram to model traffic flows and speeds is shock wave theory. **Shock waves** are the spatiotemporal boundaries between two different traffic phases and therefore indicate how congestion heads propagate. Within shock wave theory, an assumption is that these speed changes happen instantaneously, so the shock wave itself has not physical length. The number of vehicles 'entering' the wave must be equal to the number of vehicles exiting the wave. When calculating the entry and exit rate in the frame of a moving observer moving with the speed of the shockwave, the exit rate is

$$q_{exit} = k_{down}(u_{down} - w) \tag{2.2}$$

where k_{down} and u_{down} are the density and speed downstream of the shock wave and w is the shockwave speed. 'Upstream' of the shockwave, the entry rate is calculated in a similar way

$$q_{entry} = k_{up}(u_{up} - w) \tag{2.3}$$

where k_{up} and u_{up} are the density and speed downstream of the shock wave. Since the conservation of vehicles holds, $q_{entry} = q_{exit}$, as no vehicles can enter or leave the road at this (spatially) infinitely small shockwave, the equations can be rewritten as follows:

$$k_{up}(u_{up} - w) = k_{down}(u_{down} - w) \tag{2.4}$$

$$q_{up} - k_{up}w = q_{down} - k_{down}w \tag{2.5}$$

$$w = \frac{q_{up} - q_{down}}{k_{up} - k_{down}} = \frac{\Delta q}{\Delta k} \tag{2.6}$$

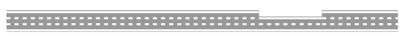
Substituting in $q = ku$ (assuming that the traffic situation is stationary and homogeneous), one could rewrite this to:

$$w = \frac{q_{up} - q_{down}}{\frac{q_{up}}{u_{up}} - \frac{q_{down}}{u_{down}}} \tag{2.7}$$

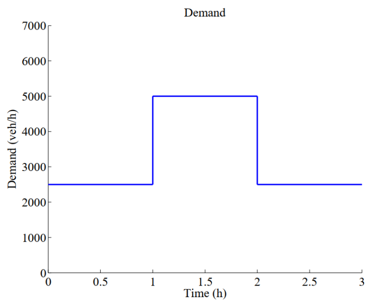
Based on this, we can conclude that, according to the shockwave theory, the propagation of shockwaves between two traffic phases is equal to differences in flow, speed and density upstream and downstream from the bottleneck. Based on this, we can calculate these shockwave speeds based on the demand pattern of a road and the capacity characteristics of a road, governed by the fundamental diagram. Let us now apply shockwave theory to explain the propagation of two basic types of congestion often experienced at highway bottleneck.

Fixed bottleneck

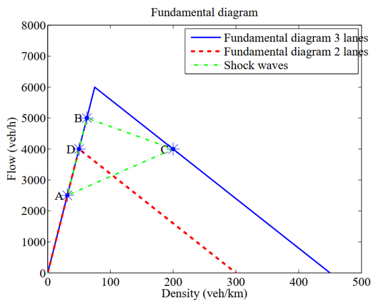
A fixed bottleneck is a bottleneck that occurs due to traffic flow exceeding road capacity. The simplest bottleneck one can think of is a lane merge, as road capacity is reduced significantly at that point. An example of a lane merge is given in Figure 2.2. In the demand pattern and fundamental diagram, we can see that traffic demand rises from 2500 veh/h (phase A in the fundamental diagram) to 5000 veh/h (phase B in the fundamental diagram), leading to flow exceeding the capacity at the two-lane section (4000 veh/h) for one hour. Since the inflow of traffic downstream of the lane merge is restricted to the bottleneck capacity, traffic flow at the two-lane section will be restricted to its capacity 4000 veh/h, equal to phase D in the fundamental diagram, and congestion will occur upstream of the lane merge. Since at the shockwave located at the lane merge the conservation of vehicles holds $(q_{up} = q_{down})$, the flow within this congested road section is equal to the bottleneck capacity but its speed and density are situated at the congested branch of the fundamental diagram, depicted by phase C. This leads to a shockwave between phases B and C with a shockwave speed of $w_{BC} = -7.3$ km/h, propagating upstream. When traffic demand decreases again, the inflow in the congested area will be equal to 2500 veh/h, represented by phase A, resulting in a shock wave between phase A and C having a speed of $w_{AC} = 8.9$ km/h. Since this shockwave is propagating upstream, congestion will solve and traffic will go back to free-flow.



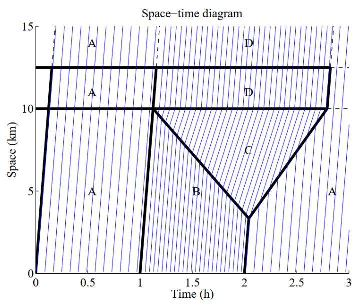
The road



Demand pattern



Position of traffic states at the fundamental diagram



Shock waves in space-time

Figure 2.2: Shockwave theory at lane merge bottleneck. Adapted from [1].

Stop-and-go waves

Another traffic phenomenon that can cause congestion are stop-and-go waves. They arise during 'critical' traffic conditions, which means that traffic flow is near capacity and having a relatively high density. Within these conditions, local instabilities in traffic can result in a traffic breakdown, locally reducing the average speed to almost stand-still. This can be elaborated with an example. Since flow and density are both high, the headways between cars are small too. Sudden maneuvers of other drivers together with distraction of others can introduce situations where sudden braking is required. During non-critical free-flow conditions, traffic flow is low and as traffic flows out of the downstream jam head at the road capacity. Since road capacity is much higher than the traffic inflow, the downstream head will move upstream faster than the upstream head of the stop-and-go wave, leading to the dissolution of the wave. However, when traffic is critical and traffic flows are (almost) equal to the road capacity, the upstream and downstream head will move at (almost) the same speed. This leads to a stop-and-go wave which will propagate through the network until traffic flow decreases (or it reaches a road segment with a higher capacity). Depending on the road characteristics and the resulting fundamental diagram, the shockwave speed of stop-and-go waves ranges from 15 to 20 km/h [23] and is equal to the slope of the congested branch in the (k,q) fundamental diagram. An example of a speed heatmap showing stop-and-go waves are present Figure 2.3, where speeds within the wave are close to standstill and traffic speeds around the stop-and-go waves are relatively low (synchronised flow), meaning that traffic conditions are critical.

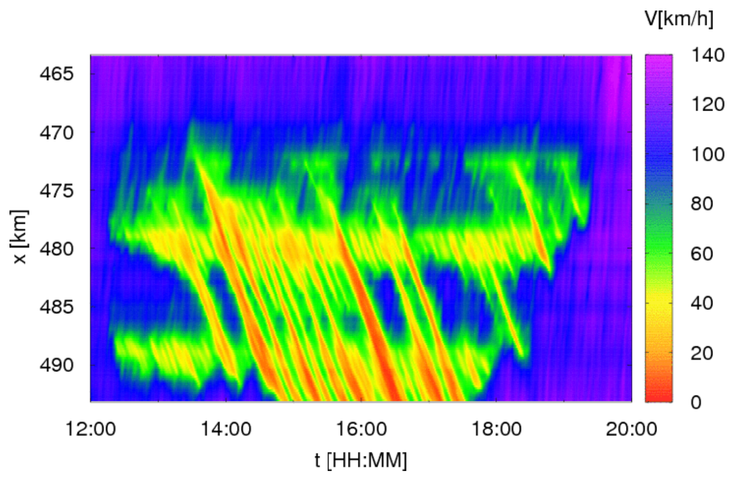


Figure 2.3: Stop-and-go waves from the German A5. Adapted from [2].

In practice, the queue discharge rate of the traffic jam is lower than the road capacity. This phenomenon is called 'capacity drop' and is captured by several fundamental diagram shapes. Due to this effect, stop-and-go waves can even propagate if traffic flow is lower than road capacity and can also trigger 'fixed bottleneck' congestion at the location where the stop-and-go waves are formed. By temporarily reducing road capacity of a segment, this segment is effectively transformed into a bottleneck. Off-ramp bottlenecks are a good example for this: a large amount of vehicles exiting a highway can cause turbulence due to lane changing. In combination with high through-going traffic flows, this can lead to stop-and-go waves which eventually can lead to fixed-head congestion.

2.2.2. Empirical Highway Congestion Patterns

Although traffic congestion patterns can be described quite accurately using shockwave theory and the fundamental diagram, some congestion patterns which can be observed empirically are hard to explain using these theories. Besides, it was explained that the fundamental diagram and fundamental relationships have some flaws. Contrary to using a classical model, neural networks learn based on input data only, making it important to look at discrepancies between theory and practice. Since many research also focuses on analysing empirical congestion patterns and explaining these, a brief summary on their view of congestion pattern classification is also given. This is done according to the three-phase theory of Boris Kerner: although some experts have disproven his theory, his idea of synchronised flow and his classification are still relevant when discussing empirical congestion patterns.

According to Kerner [24], traffic flow does not consist of two but rather three phases: free flow (F), *synchronised flow* (S) and congestion (J), as he proposes a new *empirical* fundamental diagram including this synchronised flow phase, as can be seen in Figure 2.4. This fundamental diagram also shows the queue discharge rate q_{out} besides the capacity q_{max}^{free}. The phase is characterized by relatively high speeds (40-70 km/h), whereas traffic flow ranges from low to near-capacity. According to his theory, it is not directly possible to have a phase transition from a free flow to a congested phase, but only via the synchronized flow phase. Traffic phases always move from free-flow to synchronized flow and then (possibly) to congestion, so an $F \rightarrow J$ transition is not possible, but an $F \rightarrow S \rightarrow J$ is. Rather than having a congested branch in the fundamental diagram, the synchro-

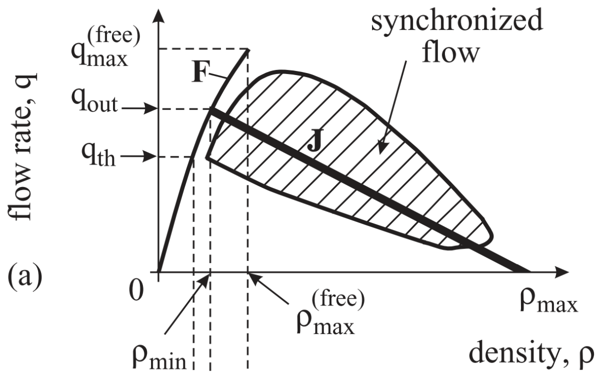


Figure 2.4: Empirical Fundamental Diagram. Adapted from [3].

nized flow phase is situated on a plane around the congested branch, and traffic in this phase can freely move to any point within this plane [3]. This also explains why congestion does not always lead to significantly lower traffic flows compared to free-flow, contrary to the theoretical fundamental diagram. There are a large amount of locations at which traffic breakdowns are possible, such as on-ramps, off-ramps, lane merges or even spontaneous, due to several causes, such as an upstream-moving jam wave, spillback of a synchronized flow section or a sudden reduction in capacity. These can all lead to different speed patterns as they result in different flow rates and (temporary) capacities [24]. Therefore, a short overview of possible patterns is given in this chapter, including the characteristics leading to these patterns.

Although some researchers have found this can also be described with two-phase traffic flow theory, it still is relevant to present this as it explicitly treats some interesting characteristics of empirical traffic patterns. Therefore, a few concepts regarding empirical congestion patterns according to this synchronised flow theory will be elaborated.

Synchronized Flow Patterns
An $F \rightarrow S$ traffic breakdown at an isolated bottleneck can cause various synchronized flow patterns (SPs), each having differences in the spatiotemporal propagation of the upstream and downstream jam heads, namely

1. Localized SP (LSP)
2. Widening SP (WSP)
3. Moving SP (MSP)

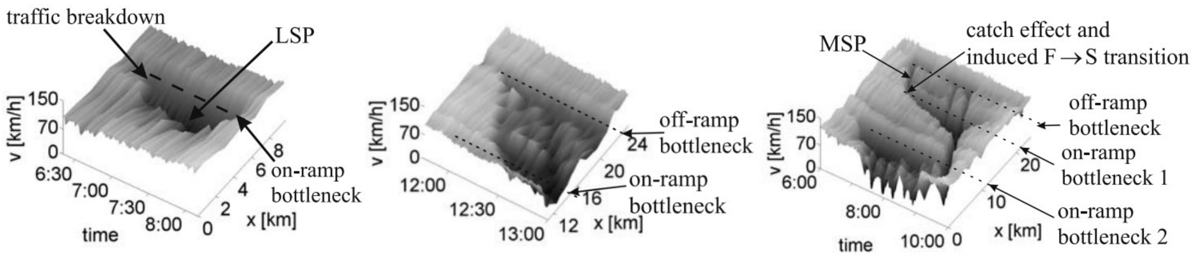


Figure 2.5: Speed data from LSP (left), WSP (center) and MSP (right) patterns. Adapted from [4].

Depending on the flow rates at a road, both the inflow of an on-ramp as well as the throughgoing flow on a road, a different SP can occur, as can be seen in Figure 2.6. An **LSP** can occur at relatively low flow rates, both from the highway as well as on the on-ramp. A figure containing traffic speeds over space and time for this pattern is the left figure of Figure 2.5. Its downstream front is fixed at the bottleneck and the upstream front has a stabilized location upstream of the bottleneck. Since flow rates are just higher than capacity, the location of the upstream front is fixed. When (in)flow rates increase, the probability of a **WSP** increases, which speed patterns can be seen in the center figure of Figure 2.5. Its downstream front is also fixed at the bottleneck but the upstream front continuously propagates upstream over time. At higher (in)flows, **MSP**s can occur, which can be seen in Figure 2.5 on the right. Both its downstream and the upstream front propagate, and it can result in a stop-and-go waves. MSPs show two different interesting congestion effects, leading to its complex shape. The first effect is called the *pinch effect*, which means that from the synchronised flow section located stream downward near the bottleneck location, stop-and-go waves can occur. Due to the self-compression of this synchronised flow, densities increase and speeds decrease leading to a spontaneous $S \rightarrow J$ breakdown necessary for these stop-and-go waves. This can be seen in Figure 2.5 by the narrow jam waves propagating upstream having small traffic speeds. The second effect is called the *catch effect*, meaning that if the MSP reaches a bottleneck location, it can cause a continuous traffic breakdown. This can cause a (second) SP to emerge with its upstream front fixed at the bottleneck location. This can also be seen in Figure 2.5.

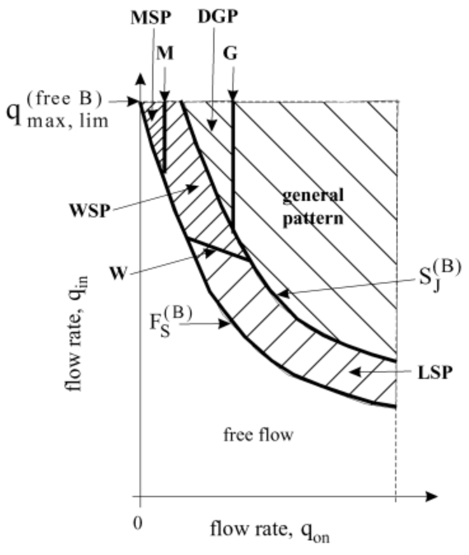


Figure 2.6: relative inflow (q_{in}) and flow (q_{on}) values at which different patterns occur. Adapted from [4].

General Congested Patterns

After traffic breaks down and an $S \rightarrow J$ phase change occurs, general congested patterns (GPs) can emerge at heavy bottlenecks. General patterns are characterised by a large synchronised flow area in which many stop-and-go waves occur, leading to lower speeds and flows compared to synchronized flow. A GP can be seen as a WSP which has higher traffic flows so more stop-and-go waves occur. In Figure 2.7, two empirical examples from GPs are given, one resulting from a WSP at an off-ramp transforming into a GP and one of an on-ramp bottleneck where a GP occurs due to higher flow rates. If traffic inflow rates increase more, so-called **mega-jams** can occur, which are wide stop-and-go waves which can be seen as one large stop-and-go wave which is not interrupted by sequences of synchronised flow patterns [25]. Therefore, traffic speeds are lower compared to GPs. Since they reduce the bottleneck capacity even stronger, almost zero, they can evolve into large sections of motionless vehicles.

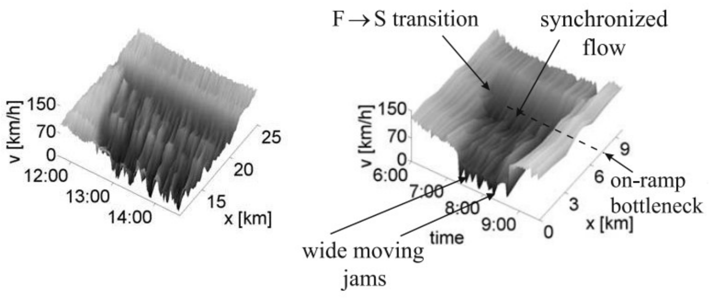


Figure 2.7: Speed data from GPs resulting from WSP (left) and near an on-ramp bottleneck (right). Adapted from [4].

2.3. Traffic Breakdown Processes

Besides predicting the *propagation* of congestion, which leads to predicting right congestion pattern type, it is also important to assess whether we can predict the *emergence* of a congestion pattern which is defined by the traffic breakdown process. In general, traffic breakdown and therefore congestion occurs when traffic demand exceeds supply. Within macroscopic traffic flow modelling, one generally assumes that when traffic flow exceeds a static capacity value, congestion will occur. However, this approach is rather simplistic compared to reality and is not suitable for short-term traffic forecasting. Therefore, this chapter will further zoom in on these traffic breakdown processes in a macroscopic context.

This chapter is structured as follows. First, traffic supply will be presented, including the variables influencing supply. Secondly, traffic demand will be analysed in a similar way. To conclude, several researches which try to capture elements of the traffic breakdown process will be presented.

Traffic *supply*, determined by the road *capacity* depends on a number of factors. It is mostly determined by the road layout, such as the (a) geometry, (b) number of lanes, (c) turbulence (by checking spacing between on- and off-ramps) and (d) other disturbances such as tight curve radii, high slopes or tunnels [20]. However, other parameters which can fluctuate over time can also influence road capacity. Vehicle composition can also influence road capacity, as a high number of trucks can decrease road capacity [20]. Brilon *et al.* [26] suggests that the road capacity depends on expect them to depend on control conditions, driver/vehicle populations and even travel purposes. van Lint [27] also suggests factors as ambient conditions (weather, road visibility), road works and traffic collisions.

Capacity should therefore not be seen as a fixed value, but an infinite number of free-flow capacities for the same road exists [3]. Several attempts have been made to define or model this probability, roughly using two approaches: either by analysing empirical data or by modeling traffic flow in new ways to better capture existing phenomena. Brilon *et al.* [26] attempted to capture an empirically observed probability distribution constructed of traffic breakdown observations on several roads on which clearly distinguishable bottlenecks were present. They fitted a Weibull distribution through this empirical probability function and analysed the parameters resulting from this fit. They concluded that the parameters describing this distribution vary widely between highways. However, the derived parameters had no physical meaning. Hoogendoorn *et al.* [28] created a mathematical model which tries to more accurately model the traffic breakdown phenomenon for macroscopic traffic flow models by including breakdown probabilities. They state that the breakdown probability changes based on traffic density (over space and time) and the characteristic curves of the kinematic wave model, which can be derived from the fundamental diagram. Their approach seemed to be successfully incorporate breakdown probabilities and showed it can be used to model several breakdown phenomena, as well as use it to create a stochastic first-order macroscopic model. This proves that seeing traffic breakdown as a probabilistic event better captures traffic breakdowns.

As capacity is not constant and most factors it depends on cannot be observed directly, predicting capacity can be difficult. Although road capacity can be observed during traffic breakdown and can be fitted to a probability distribution, there still is a lot of uncertainty on when traffic will exactly break down.

Traffic *demand* is the sum of the traffic demand between certain origins and destinations and the mode and route choice of these groups. It is also influenced by traffic information and network effects, which can depend the traffic state of potentially unobserved roads [27]. Traffic demand cannot be observed directly as one can only observe the product of supply and demand by looking at the traffic flows on roads. This can specifically pose problems when predicting traffic on a highway that has many on- and off-ramps. Due to model size constraints, only traffic data from the highway is used as input data, meaning that surges in traffic flow caused by on-ramps and off-ramps caused by the the road network to which these on- and off-ramps are connected cannot be predicted. To be more concrete, if traffic demand changes causing more people to enter or exit the highway at certain points, this cannot be observed from the data of this highway alone.

To summarise, it can be concluded that both traffic supply and traffic demand are difficult to predict. Although approaches to model a probabilistic capacity and traffic breakdown in a macroscopic way have been successful Brilon *et al.* [26], Hoogendoorn *et al.* [28], many challenges can be formulated with respect to correctly predicting traffic breakdown. Therefore, predicting the emergence of congestion is likely to be difficult.

2.4. Predictability of Congestion Patterns using Macroscopic Variables

To conclude this chapter, a short overview will be given on the predictability of congestion patterns using macroscopic variables. In this context, *predictability* is defined as whether it is possible to predict the traffic state, consisting of the macroscopic traffic variables, with a sufficient accuracy given the proposed prediction horizon. With *sufficient* is meant that the emergence or propagation of a traffic phase is captured correctly although the exact nature of it might deviate.

Based on this chapter, it can be concluded that there is plenty of theory to correctly predict the propagation of congestion patterns based on shockwave theory. However, an important shortcoming can be formulated which are not covered by this theory. As fluctuations in traffic demand are hard to predict, changes on in- and outflows from on- and off-ramps can disrupt the predictions quite severely. In case a longer road without discontinuities (i.e. on- and off-ramps) is considered, upstream and downstream traffic conditions can be used to predict traffic conditions. However, in case a shorter road is considered or there are many discontinuities or interchanges with a lot of merging and diverging traffic, this might not be the case.

However, the emergence of congestion patterns is hard to predict. As research shows that it is possible to capture road capacity in a probability distribution and that certain traffic breakdown processes can be modelled in a macroscopic, it is difficult to correctly predict traffic breakdown as supply and demand cannot be observed directly. Contrary to the *propagation* of congestion patterns, there is no clear theoretical framework which can be used to predict the *emergence* of congestion patterns. This can have serious consequences on the quality of the resulting traffic predictions.

3 Neural Network Theory

As explained before, there are several methods possible for traffic forecasting, which can roughly be divided in parametric and non-parametric approaches. Recently, the application of machine and deep learning within different domains has been booming and is standard for applications like image recognition, natural language processing, classification and prediction. For a quick overview of the position of neural networks and deep learning within the field of AI, which is an umbrella term to which machine learning belongs, the reader is referred to Figure 3.1. Specifically for time series forecasting, the application of a recurrent neural network (RNN), a specific variant on neural networks, is researched extensively, also within the context of traffic forecasting [29]. These methods often lead to much better performance compared to other traffic forecasting methods. This chapter will introduce the concept of neural networks and specifically their use within the traffic forecasting domain.

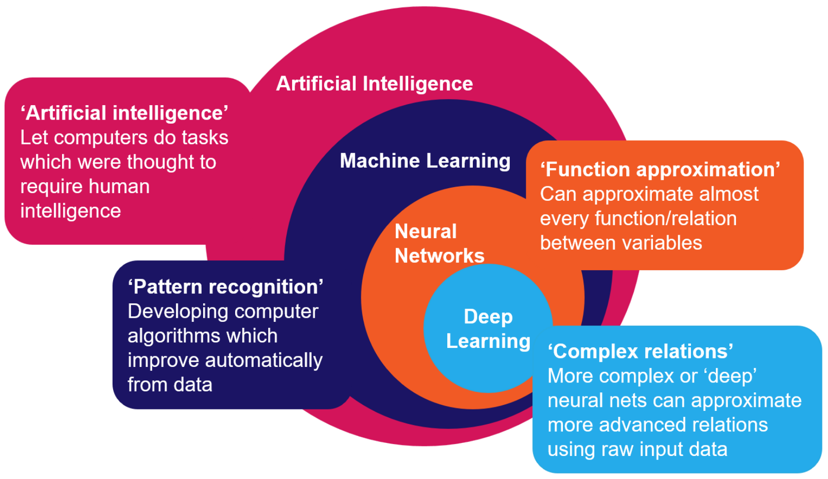

Artificial Intelligence
'Artificial intelligence' Let computers do tasks which were thought to require human intelligence
Machine Learning
'Function approximation' Can approximate almost every function/relation between variables
Neural Networks
'Pattern recognition' Developing computer algorithms which improve automatically from data
Deep Learning
'Complex relations' More complex or 'deep' neural nets can approximate more advanced relations using raw input data

Figure 3.1: An overview neural networks and deep learning within the AI landscape.

This chapter is structured as follows. First, the concept and workings of a simple feed-forward neural network is explained, zooming in on model training and testing. Secondly, a RNN is presented and its differences from a regular neural network are explained. Thirdly, we zoom in on the different RNN architectures and implementations when used for (multivariate) time series prediction. Lastly, a quick overview is given of the recent advantages of research on predicting traffic variables on highways using RNNs.

3.1. Feed Forward Neural Network

A standard feed-forward neural network can be seen as universal function approximator. Its goal is to approximate some function $f*$, such as a regression function $\mathbf{y} = f(\mathbf{x};\boldsymbol{\theta})$, which maps an input $\mathbf{x}$

to an output **y** by learning the parameters θ which together lead to the best fit of x to y [8]. They are called feed-forward since the values of the input **x** are passed forward through a series of functions which together form the complex function $f*$ and which in the end result in the output **y**. It is called a network since it links several functions $f^{(1)}, f^{(2)}, ..., f^{(n)}$ together to form the function $f*$. Each individual function is also called a hidden layer, which is called 'hidden' since its direct outcome is not read and interpreted but used as input for another function. It is known that for some activation functions, even nonlinear functions can be approximated.

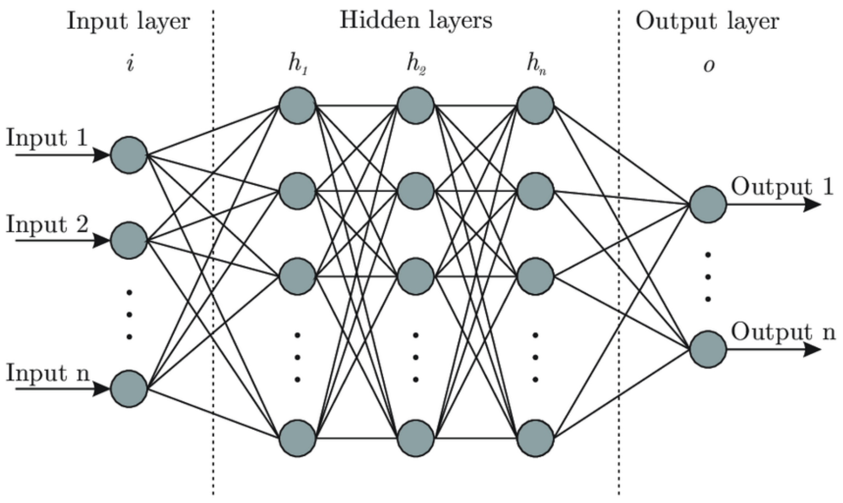


Figure 3.2: Graph representation of a neural network. Adapted from [5].

Although this sounds quite abstract, a neural network can be more intuitively represented by its computational graph, which is a directed acyclic graph as shown in Figure 3.2. In this graph, each node represents a function depending on its internal parameters and the resulted values of connected nodes. This computational graph consists of several horizontal layers, namely an input layer, which holds all the input parameters, an output layer, holding the (desired) output parameters and one or more hidden layers connecting the two. Together with the output layer, each hidden layer corresponds a function f^n, where n is the hidden layer index, which together form the function $f*$ the model should approximate. Each node is (usually) connected with all the nodes of the previous layer, making its function depend on all resulting parameters of the previous function. Goal of this neural network is that each unit of each layer sets its parameter values in such a way that ultimately, the right relation between the input nodes and an output node value can be found. The amount of hidden layers or chained functions is called the **depth** of a model, whereas the dimensionality of each layer is called its **width**. Both the width of each layer and the total depth of a neural network can be changed in order to optimise the model.

Let us now zoom in on the operations of only one node. As mentioned before, each node represents a function which can be formulated as as follows:

$$y = \phi\left(\mathbf{w}^T\mathbf{x} + b\right) \tag{3.1}$$

where ϕ is the activation function, **w** are the weights, **x** are the values of all connected nodes and b is the bias term. You can see that each node outputs a value using a linear function of its input values, including a bias term, after which on the resulting value a certain activation function is applied. When a linear activation function is used, which is the same as using no activation function, the relation between each input and output pair in the neural network will be linear too. However, when a nonlinear activation function is chosen in at least one of the layers, such as the rectified linear unit (ReLU), tanh and sigmoid function in Figure 3.3, a nonlinear function between the inputs and outputs can be formed. This essentially makes a feed-forward neural network a combination of chained linear regressions which by applying nonlinear activation functions can be made nonlinear.

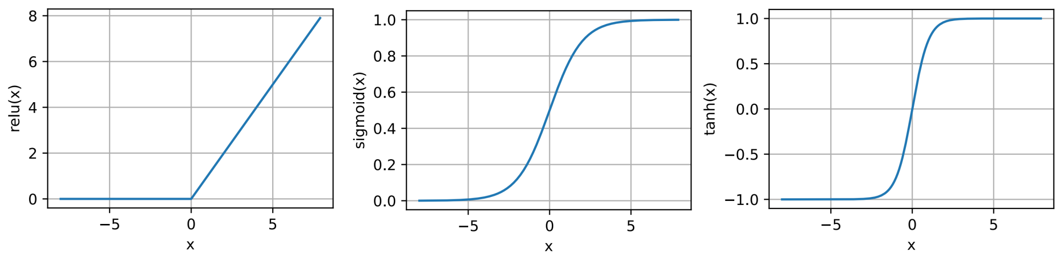


Figure 3.3: Examples of a ReLU (left), sigmoid (center) and tanh (right) activation function. Adapted from Zhang *et al.* [6].

3.1.1. Neural Network Training

When initialising a neural network, all weights and biases are chosen randomly according to a certain distribution or chosen as zero. Therefore, the neural network needs to be trained so it fits the right weights and biases to the data such that the correct relation between input and output values is found. This is done by using a labelled' training set which, besides the input values $\mathbf{x}$, also contains the true output values $\mathbf{y}$. The idea with training is that the model estimated output values $\hat{\mathbf{y}}$, retrieved from the output layer of the neural network, are compared with the true output value $\mathbf{y}$ and the model weights are changed so the difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$ is minimised. This is done using three steps, which will be explained in this paragraph. First, the input parameters are propagated through the neural network to retrieve $\hat{\mathbf{y}}$. Secondly, the loss function is calculated, which quantifies the difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Thirdly, the model parameters are updated, for example using backpropagation, to decrease the loss function value.

First, the input values $\mathbf{x}$ are propagated through the network to calculate the estimated output $\hat{\mathbf{y}}$ resulting from the activation functions and model parameters of each hidden layer (as in Equation 3.1). Secondly, the difference between the observed values ($\mathbf{y}$) and predicted values ($\hat{\mathbf{y}}$) has to be determined to assess the accuracy of the model. This can be done using a **loss function**, which formalises the differences between all values of ($\mathbf{y}$) and predicted values ($\hat{\mathbf{y}}$) into one single value [30]. One can imagine this loss function can be formulated in many different ways, depending on the goal of the model. When the goal of the model is to make a correct classification between groups, a loss function is required which focuses on increasing the probability of a right classification. When solving a regression problem, estimating the correct value, the difference between the ground truth and prediction should be minimised. As in this thesis a regression problem is solved, a few regression loss functions will be presented.

- mean squared error (MSE), which emphasises large (absolute) differences between prediction and ground truth values compared to small differences:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{3.2}$$

- mean absolute error (MAE), which does not differentiate between magnitudes of differences

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{3.3}$$

Thirdly, the value of this loss function should be minimised by changing the model weights, since a lower loss will result in a higher model accuracy. As the values of $\hat{\mathbf{y}}$ depend on all model parameters, one could see the loss function as a function with a single output value (the loss) which depends on all model weights. By finding the model weights corresponding to the (global) minimum of this loss function, a set of weights which lead to the highest model accuracy can be found. Using gradient descent, a minimum of this function can be found. A minimum of a function $f(\mathbf{x})$, where $\mathbf{x}$ = the set of weights and biases of a neural network, is situated at a location where the derivative of this function is zero $\left(f'(\mathbf{x}) = 0\right)$. In the case of multiple inputs, one can calculate the partial derivative $\frac{\partial}{\partial x_i} f(\mathbf{x})$

which separately calculates the derivative with respect to all inputs. All partial derivatives with respect to all input variables can then be combined into the gradient, as defined in Equation 3.4.

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d}\right]^\top \tag{3.4}$$

Due to this definition, the gradient is equal to the direction in which the function $f(\mathbf{x})$ increases the most. Therefore, by calculating the gradient and moving $\mathbf{x}$ in the opposite direction of this gradient, the value of $f(\mathbf{x})$, or the loss function, will decrease. This method is called **gradient descent** [8], and is formalised in the following equation:

$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x}) \tag{3.5}$$

where ϵ is the learning rate, which is equal to the step size with which $\mathbf{x}$ is moved towards the gradient. For a two-dimensional function, this might look like Figure 3.4. The higher situated, red areas have a high loss function value whereas the lower, blue areas have a low loss function value. The red circle shows the initial value and the black path the way a minimum is found, by finding the intermediate values denoted by stars and the spacing between those is equal to the learning rate. This figure also shows two different challenges using gradient descent. The first is that the resulting function is highly complex with different minima, making it hard to find the global minimum. However, often, a local minimum suffices too. The second problem is that it is very sensitive to the learning rate. If a learning rate is too small, it might not converge, but when it is too big, the algorithm might 'hop over' a minimum or not converge at all.

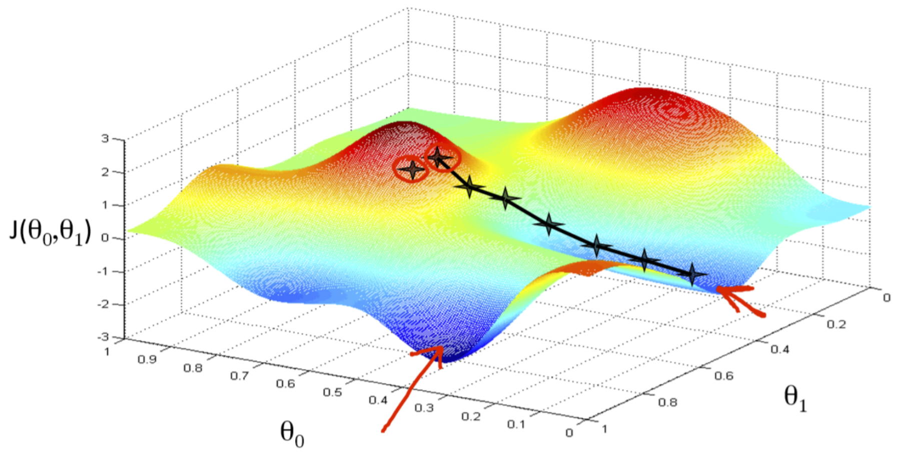


Figure 3.4: Representation of gradient descent using a two-dimensional example. Adapted from [7].

Deriving an analytical expression for the gradient is straightforward, but calculating this expression can be computationally expensive. Using the **backpropagation** algorithm [31], this calculation method can be simplified and made less computationally expensive, especially for multi-layered neural networks. By recursively applying the chain rule from the output layer to the (last) hidden layer up to the input layer, the gradient can be calculated. Since calculations start at the 'back' of the neural network and are propagated to the front, until all layers have passed, this step is called backpropagation.

Although gradient descent works really well, it is also computationally expensive, especially when using additive loss functions (such as MSE and MAE) which requires to calculate a gradient for each training step, making the computational cost $O(m)$ where m is the number of training steps. However, stochastic gradient descent sees the gradient as an expectation, which might be estimated using a smaller set of samples. This **minibatch** containing (m') samples, drawn uniformly from the dataset, can then be used to estimate the gradient, reducing the computational cost to $O(m')$. Therefore, stochastic gradient will decrease the computational cost of training a neural network.

3.1.2. Model Testing

After designing the neural network and training it on a dataset, assessing the training error, it is also important to test the model performance using another dataset with unseen input data. After all, we want our model not to just work using the data it was trained on, but also to be generalisable, meaning it also perform well on new, unseen inputs. In order to do this, we create a (smaller) test set and also look at the performance for this dataset. In the end, we want a model which both has a small training error as well as a small gap between the training and test error. Both terms depend on the **capacity** of the model, which is the extent to which a model can approximate a wide variety of functions. In order for the model to also perform on unseen data, we want the model complexity to match the underlying problem, i.e. the 'real' input-output relationships. This can be explained best using two terms, namely **overfitting** and **underfitting**, which are depicted in Figure 3.5 showing the error rates for a problem with respect to the model capacity. In case the model is overfitting, the training error is small but the gap between the training and test error is high. This happens when the model capacity is too high, either with respect to the chosen problem or the size or data in the training dataset. Since the model fits to present in the particular (limited) dataset and its capacity is high enough it can have a good fit on this dataset, its training error will be exceptionally low. The consequence is that some of these patterns might not be present in other datasets it will be used on, making it less generalisable and not useful in practice [32]. However, when the model is underfitting, its capacity is not high to capture the complex input-output relation present in the training data. Therefore, both the training and generalisation error will be relatively high. One could say that the neural network should be large enough to adjust its weights to the largest regularities but ignore the smaller ones, which might be due to noisy data.

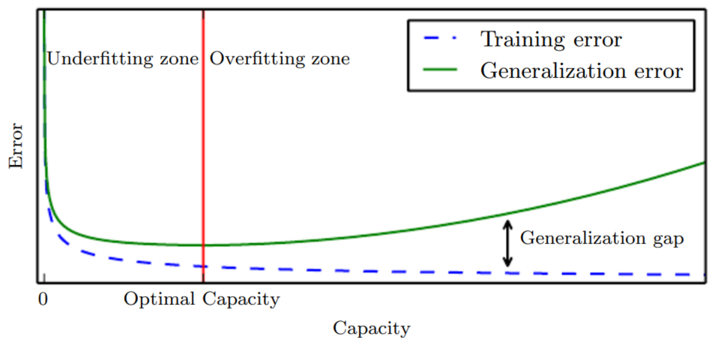


Figure 3.5: Overfitting and underfitting with respect to the model capacity. Adapted from [8].

3.2. Recurrent Neural Network

A recurrent neural network (RNN) is a type of neural networks specifically designed to process sequence data instead of one-dimensional inputs. Each input node can handle a sequence of arbitrary length instead of a single number, due to the introduction of recurrence. This recurrence is introduced by making the state (or value) of each unit in a recurrent layer not only dependent on its current input, but also on the state resulting from the input at the previous time step of the sequence. This results in two advantages when using a RNN over a FFNN with sequence data as input, (i) the number of parameters used for a RNN will be far less than a FFNN, and (ii) a RNN can (in theory) handle sequences of arbitrary length. This section is structured as follows. First, the concept of recurrence will be further explained and related to the standard ('vanilla') structure of an RNN, analysing why this leads to the aforementioned advantages when using sequence data. Secondly, the vanishing gradient problem will be explained, resulting in weaknesses of the 'vanilla' RNN structure. Thirdly, the gated recurrent unit (GRU) will be introduced to see how implementing GRU cells in an RNN structure can greatly improve accuracy.

3.2.1. Vanilla RNNs

Let us further explore the concept of recurrence by introducing the computational graph of a single RNN unit. By replacing each unit of a FFNN with a single RNN unit, one can create a RNN. This recurrence is depicted by a cyclic connection of the node (unit) with itself, making its updated hidden state (or value) not only dependent on its input, but also its previous hidden state. This can be captured by the following formula,

$$\mathbf{h}^t = f\left(\mathbf{h}^{t-1}, \mathbf{x}^t; \theta\right) \tag{3.6}$$

showing the dependence of the hidden state $\mathbf{h}^t$ at time step t depending on input $\mathbf{x}^t$, the previous value of the hidden state $\mathbf{h}^{t-1}$ and unit parameters θ. The computational graph of one RNN unit is shown in Figure 3.6, where the recurrence is captured by a cyclic connection of the hidden state h, meaning that there is a connection between the hidden state of the previous and current time step. By unfolding this computational graph, this becomes more clear, as this means that the hidden state contains information of all previous inputs, which is used for calculating the next hidden state. This unfolded representation of Equation 3.6 of hidden state $\mathbf{h}^t$ can be represented by Equation 3.7, where function g^t is introduced, which is an arbitrary function specific to hidden state h^t to define its value based on the previous input values. Instead of applying function f multiple times, passing historical input values through the previous hidden state, function g directly uses the inputs until time step t to calculate the output. Since they are a different representation of the same computational graph, both equations should hold the same. This is true since with this unfolding step, function g^t can be represented by chaining the function f repeatedly as shown in Equation 3.8, which is equal to the previously known Equation 3.9.

$$\begin{aligned}\mathbf{h}^t &= g^t\left(\mathbf{x}^t, \mathbf{x}^{t-1}, \mathbf{x}^{t-2}, \ldots, \mathbf{x}^2, \mathbf{x}^1\right) && (3.7)\\ &= f\left(f\left(\mathbf{h}^{t-2}, \mathbf{x}^{t-1}; \theta\right), \mathbf{x}^t; \theta\right) && (3.8)\\ &= f\left(\mathbf{h}^{t-1}, \mathbf{x}^t; \theta\right) && (3.9)\end{aligned}$$

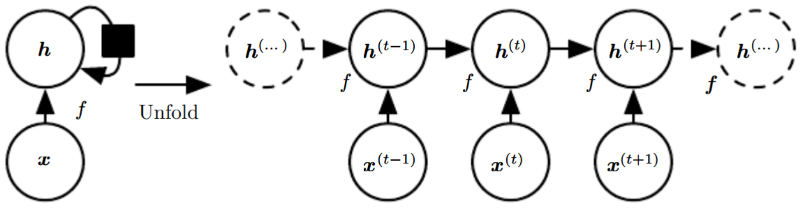


Figure 3.6: Graph representation of a regular (left) and unfolded (right) RNN unit. Adapted from [8].

So why is this recurrence introduced, and what are the advantages of having a recurrent neural network over a regular feed-forward neural network? Let us come back on the two advantages presented in the introduction. The alternative of having a RNN with a sequence per input unit is having a regular FFNN with a separate input node for each value of a sequence of each input parameters. Therefore, each hidden node consists of a function with as input all sequence values over time, similar to Equation 3.7, although all values of all t time steps would be present in one big vector. First, the input layer of the FFNN becomes very large, resulting in a high number of parameters that have to be learnt and possibly it would be very inefficient to learn it. Since a RNN applies the same function f over all values of a certain sequence time step, it essentially applies a parameter sharing technique where the same parameters are used within the sequence values belonging to a parameter. This is very valuable for time series analysis where the parameters do not change over time. Sometimes it can even be beneficial to not have separate parameters for each value of the time index, for example at natural language processing, where a desired piece of information can be at different locations in a sentence (which is the sequence). By encoding this information in the hidden state of a unit, it is assigned to an input sequence instead of an input belonging to a certain time step and input.

Secondly, for a FFNN, if the sequence length changes, a new function g has to be learnt since the old function can not handle the changing number of input parameters. Since a RNN uses the same parameters on each sequence time step, it has not trouble handling sequences of arbitrary length, which especially is the case with sentences at for example natural language processing.

Although this parameter sharing technique and handling of sequence data in RNNs is convenient, it does come with a price. First, as mentioned before, the parameter sharing technique assumes that the same parameters are applicable to different time steps. This also means that it requires the dataset to be **stationary**, meaning that the conditional probability distribution of the variables at time step $t+1$ does only depend on the variables at time step t and not explicitly on the value of t itself. Although the value of t could be explicitly present in the input data, mitigating this problem, it is important to keep this design choice in mind.

Secondly, the sequential processing of data makes it harder to optimise the parameters of the network [33], due to the recurrence introduced which can result in a deep and complex computational graph, which can lead to a **vanishing gradient** problem [34–36]. As mentioned before, when calculating the gradient using backpropagation, the chain rule is recursively applied by multiplying local gradients between layers with each other from the output layer back to the input layer. The length of the multiplication depends on the depth of the model. However, when training a RNN, this backpropagation step becomes more complex, due to several problems. First, the loss function is not only formulated per output unit but also per sequence element in this output unit, making the evaluation of the loss function more complex. Secondly, when using backpropagation on a RNN to adjust the model weights, gradients are not only propagated back between layers, but also through time due to the connections of the hidden states (backpropagation through time). As this extra unfolding step makes the computational graph more complex, this also has influence on the backpropagation algorithm. When more hidden states from time steps far ago influence the outcome of a sequence, this results in a deeper backpropagation step, making the amount of elements multiplied by the chain rule larger than with a FFNN. If there are many values in this chain rule product near-zero, the gradient term will go to zero quite quickly, making it hard to change the model weights. Although in theory, this hidden state can capture long-range temporal dependencies and contextual information, this vanishing gradient problem makes this harder for standard RNNs in practice.

3.2.2. Gated RNNs

In order to solve the problem of vanishing gradients, gated RNNs such as the long short-term memory (LSTM) and the gated recurrent unit (GRU) were introduced. By introducing *gates* in RNN units, mechanisms are inserted for when a hidden state should be updated or be reset. Since the model can learn at which points these gates should be enabled, it can better regulate the flow of information over a sequence. Essentially, by learning which data is most relevant, these gates can reduce the actual model depth making the probability a vanishing gradient occurs smaller [37]. The LSTM, introduced by Hochreiter and Schmidthuber, is the oldest gated RNN and contains three different gates. The GRU, first used by Cho *et al.* [38], has a simpler design compared to the LSTM as it has only two gates, resulting in less parameters which have to be fit. Since it is suggested its performance is similar to LSTM [38], the GRU is used for this thesis and will be explained in this section.

As explained before, a GRU has two different gates. The reset gate ($\mathbf{R}_t$) will reset (parts of) the hidden state, which can be relevant if there is a break between values of a sequence where values are not related to each other. The update gate ($\mathbf{Z}_t$) will determine which (parts of the) hidden state will be updated by the new input data, which can prevent irrelevant input data from affecting the hidden state, or makes it possible to keep relevant sequence data which was given in an early stage. The equations determining the weights of both gates are given in Equation 3.10 and Equation 3.11, showing that both weights depend on the input data $\mathbf{x^t}$ and the hidden state value $\mathbf{h^{t-1}}$, having a sigmoid activation function.

$$\mathbf{R}_t = \sigma\left(\mathbf{X}_t\mathbf{W}_{xr} + \mathbf{H}_{t-1}\mathbf{W}_{hr} + \mathbf{b}_r\right) \tag{3.10}$$

$$\mathbf{Z}_t = \sigma\left(\mathbf{X}_t\mathbf{W}_{xz} + \mathbf{H}_{t-1}\mathbf{W}_{hz} + \mathbf{b}_z\right) \tag{3.11}$$

In Figure 3.7, a GRU cell is depicted, which would replace one RNN unit in the unfolded compu-

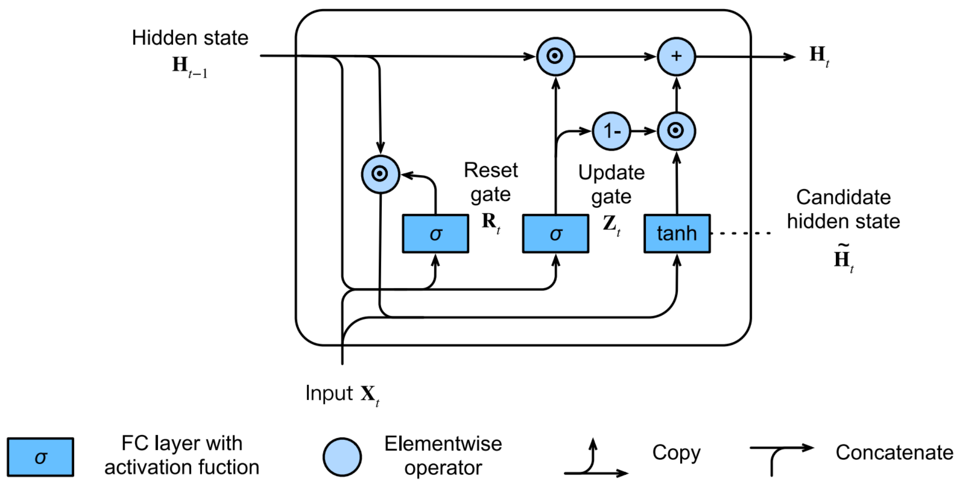


Figure 3.7: Internal representation of GRU cell. Adapted from [6].

tational graph of Figure 3.6, showing all computational elements present in the cell. Calculating a new value of the hidden state for a GRU consists of the following consecutive steps, namely:

1. a new candidate hidden state $\tilde{\mathbf{h}}^t$ is calculated, where the reset gate determines whether the previous hidden state $\mathbf{h}^{t-1}$ should be kept or 'reset' using the new input $\mathbf{x^t}$;

2. the new hidden state $\mathbf{h}^t$ is calculated, where the update state determines whether it consists of the candidate hidden state $\tilde{\mathbf{h}}^t$ or the previous hidden state $\mathbf{h}^{t-1}$.

First, we will explain the function of the reset gate, which generates a candidate hidden state $\tilde{\mathbf{h}}_t$ by point-wise multiplication ($\odot$) of the previous hidden state weights of with the reset gate weights. When the reset gate weights are close to 1, the hidden state is updated as in a regular RNN, when they are close to 0, the hidden state only depends on the input $\mathbf{x^t}$, 'resetting' the hidden state. To implement this, the previous hidden state weights of the candidate hidden state in a GRU cell (Equation 3.13) depend on the reset gate weights, compared to updating the hidden state of a regular RNN (Equation 3.12),

$$\mathbf{H}_t = \tanh\left(\mathbf{X}_t\mathbf{W}_{xh} + \mathbf{H}_{t-1}\mathbf{W}_{hh} + \mathbf{b}_h\right) \tag{3.12}$$

$$\tilde{\mathbf{H}}_t = \tanh\left(\mathbf{x}_t\mathbf{W}_{xh} + \left(\mathbf{R}_t \odot \mathbf{H}_{t-1}\right)\mathbf{W}_{hh} + \mathbf{b}_h\right) \tag{3.13}$$

Secondly, the update gate is explained, which determines in which way the new hidden state $\mathbf{h}^t$ is combined from the previous hidden state $\mathbf{h}^{t-1}$ and the candidate hidden state $\tilde{\mathbf{h}}_t$, using the update gate weights $\mathbf{z}^t$. If an update gate weight is close to 1, the new hidden state will be equal to the previous hidden state and input $\mathbf{x^t}$ will essentially ignored. If an update gate weight is close to 0, the new hidden state will be equal to the candidate hidden state. This will also influence the backpropagation process, since the model weights will not be updated when a time step has no influence on the cell outputs. In short, the reset gates help to capture short-term dependencies by wiping the internal state, whereas the update gates help to retain longer-term dependencies in the data [6, 39]. This results in Equation 3.14 for calculating the new hidden state values.

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t \tag{3.14}$$

3.3. RNN Architecture for Time Series Prediction

Although in the previous chapters, the workings of one unit or layer have been explained, the general architecture of the neural network is important too, consisting of the number of layers, the layer dimensionality (i.e. units per layer) and the way several layers are connected. Since in this thesis RNNs will be used for time series prediction, their architecture should be fit to process these temporal sequences. One can think of several questions when designing an architecture. Although the input is a sequence, should the output also consist of a sequence or rather give only one value? And how should the results of one or multiple GRU layers be combined into an output? How many hidden layers should be used and of how many units should each hidden layer consist?

In this chapter, neural network architecture design choices will be elaborated with respect to time series analysis, since predicting traffic essentially is a time series problem. First, general design problems are addressed such as the dimensionality and number of layers. Secondly, two RNN-specific design choices regarding the output dimension are elaborated, namely having a one-shot or a sequence-to-sequence model.

3.3.1. Layers and Layer Dimensionality

Both the amount of layers and the dimensionality of each layer are the main parameters when designing a neural network. The dimensionality of the input and output layer are prescribed by the input features and output labels of the input and output dataset. However, the number of hidden layers and units per layer are often chosen using preliminary data assumptions and tuned using trial and error. This hidden layer choice represents the model complexity: the higher the model complexity, the more complex the input-output patterns the model can capture. The universal approximation theorem [40] prescribes that a neural network with at least one layer (having a non-linear activation function) can approximate any continuous function where the inputs are within a certain range. The number of hidden units in this layer should be large enough so the function can be approximated. Although this is the case in theory, in practice the learning algorithm may fail to fit the correct model weights, or the absence of enough data may result in overfitting of the model. In practice, using deeper models (having more hidden layers) with fewer units can result in a better approximation of complex functions and reduce the generalisation error of the model [8].

Although deeper models in general can fit more complex functions, they are also more complex to fit. Often, complex model structures are required and hyperparameter tuning can become more complex. There is no clear rule of thumb on the model complexity required for a certain problem, as this is strongly problem dependent. In practice, model structures are chosen based on trial-and-error, by selecting a variety of different layer combinations and assessing which one performs best.

3.3.2. Model Output Dimensionality

In general, two different output dimensionalities can be considered for a RNN. The simplest output is a single-step prediction, in which all outputs are generated at once from the input sequence. Secondly, a sequence-to-sequence architecture is possible, which outputs a sequence per output unit instead of a node. Since both have their advantages and disadvantages, both possibilities will be explained and assessed.

Single-Step Prediction

In its simplest form, a RNN layer will take a sequence as input and will generate an output sequence of the same length. When predicting one step ahead with a sequence length of s, the input will consist of a sequence $[\mathbf{x}^{t-s}, \mathbf{x}^{t-s+1}, ..., \mathbf{x}^{t-1}, \mathbf{x}^{t}]$, and the output consists of a sequence $[\mathbf{y}^{t-s+1}, \mathbf{y}^{t-s+2}, ..., \mathbf{y}^{t}, \mathbf{y}^{t+1}]$. However, in time series analysis, we are in general not interested in estimating known values (between time steps $t-s+1$ and t), but only in the unknown time step $t+1$. This is graphically shown in Figure 3.8, where in this case, $s = 10$. Therefore in practice, only the last output sequence value is used. In practice, it was noticed that a dense layer is used as an output layer, which connected to the last RNN layer.

In theory, it is also possible to output a 'sequence' of values using the single-step prediction method, simply by increasing the amount of units in the output layer. However, if the model output should hold n variables over t time steps, the number of units in the output layer is equal to nt, meaning that when the amount of output variables is high, the output dimension can increase greatly with an increase in prediction horizon. Besides, since the correlations between the variables often are stationary over time, it might be smart to share parameters belonging to an output variable over time. This leads us to the introduction of a sequence-to-sequence architecture.

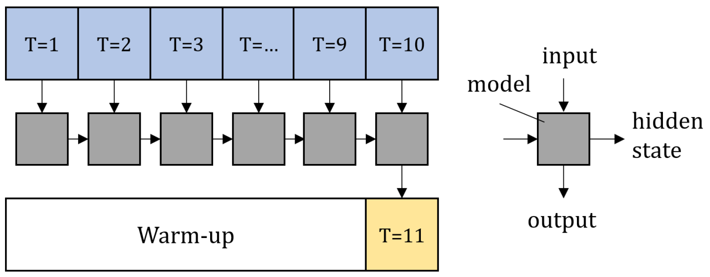


Figure 3.8: Representation of input-output connections for a one-shot model.

Sequence-to-Sequence Architecture

A sequence-to-sequence RNN model architecture was introduced independently by Sutskever *et al.* [41] and Cho *et al.* [38] to overcome the problem that for classic RNN model architectures, the output either consisted of a single timepoint or had the same sequence length as the input. Originating in natural language processing, a sequence-to-sequence architecture enables to output sequences with different lengths than the input sequence. This can be useful in for example machine translation, as a sentence in English might have a different amount of words compared to Dutch. This is done by splitting the model into an encoder and a decoder part, which both are separate RNN structures. The encoder RNN processes the input data, which results in a final hidden state. This hidden state is then fed into the decoder, together with the last prediction value, to generate the predicted sequence. The encoder and decoder can be designed separately to allow both the encoder and decoder sequence length to be chosen independently. By this hidden state sharing between the encoder and the decoder, the decoder can generate its predictions according to the information held in this hidden state. This can be captured by the following equations for the encoder calculating the hidden state at the last encoder time step t (Equation 3.15) and the decoder predicting the output at the the first (Equation 3.16) and second (Equation 3.17) decoder time step:

$$\mathbf{h}^{t} = f\left(\mathbf{h}^{t-1}, \mathbf{x}^{t}; \theta\right) \tag{3.15}$$

$$\tilde{\mathbf{y}}^{t+1} = f\left(\mathbf{h}^{t}, \mathbf{x}^{t}; \theta\right) \tag{3.16}$$

$$\tilde{\mathbf{y}}^{t+2} = f\left(\mathbf{h}^{t+1}, \tilde{\mathbf{y}}^{t+1}; \theta\right) \tag{3.17}$$

Figure 3.9 also shows this graphically, where both the input sequence length are 10 and the model predictions are fed back as input for the next value of the sequence. An important advantage of this specific sequence-to-sequence architecture, is that it can be used as an autoregressive time series model. As can be seen in Equation 3.17, the decoder uses the prediction of the previous time step as input to predict the next time step. A big advantage from setting up models in an autoregressive way is that it breaks down the prediction process in several steps. When using single-step prediction, the model instantly predicts all parameter values specified in the prediction horizon, not using parameters predicted from previous time steps to predict next time steps. When using an autoregressive model, this model is trained to predict just one step ahead but by looping this one step ahead prediction several times, using previously predicted values as input, it can generate a prediction of arbitrary length. It both uses information from the input sequence, using the hidden state of the trained encoder, and a trained decoder to estimate the right sequence from this.

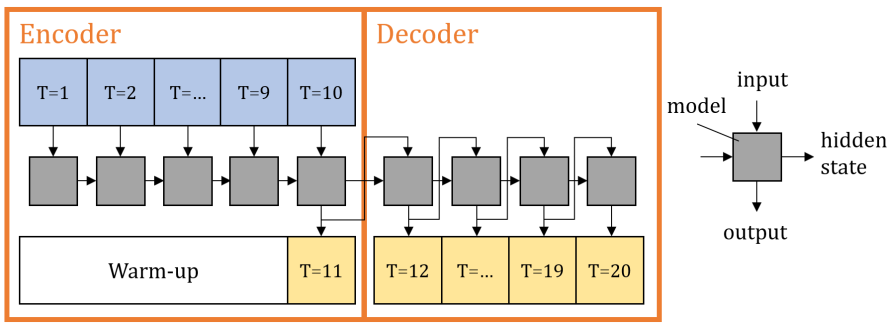


Figure 3.9: Representation of input-output connections for a sequence-to-sequence model.

However, this sequence-to-sequence approach also has two downsides. First of all, since the decoder uses previous outputs as an input for the next prediction, both the input and output have to consist of the same parameters, although shifted in time. For example: let's assume one wants to only predict traffic speed from a large variety of data (such as historical traffic flow, traffic speed, traffic density, weather predictions, public transport ridership, AEX-index, etc.) the output of an autoregressive model should also consist of this complete dataset, therefore also predicting all other values besides flows. This can lead to a larger model structure, since both the output dimension grows and the model complexity increases as these other values also have to be captured. Since the number of model parameters to be fit grows too, the model might be hard to train. As a second downside, there might be problems of error propagation when values are predicted based on previous predictions. If in an early prediction step an error occurs, the values predicted in the next steps will be based on this erroneous prediction. Therefore, a small prediction error in the begin of a sequence can grow to a larger prediction error in the end.

3.4. RNNs in Traffic Forecasting

Short-term traffic forecasting models concern making predictions ranging from seconds to hours in the future based on current and past information [42]. These models are concerned with forecasting either traffic variables, such as traffic flow and speed, or travel time [42]. Recently, short-term traffic forecasting using neural networks has sparked interest, leading to a spike in papers published on this topic from 2016 Ermagun and Levinson [29], Lana *et al.* [43], Do *et al.* [44]. This is likely the result of an overall increase of interest in artificial intelligence and machine learning, which also lead to the creation of packages like PyTorch, Keras and Tensorflow, making it relatively easy to build these complex neural network structures. Together with the increase in computational power and the possibility to train complex neural networks on consumer-grade GPUs, it became much easier to create and train neural networks. This 'democratisation' of AI lead to the application of neural networks on domains outside the traditional AI-based fields such as object recognition and natural language processing. In case of short term traffic forecasting, this also lead to a large variety of papers all having different approaches to use neural networks for traffic forecasting. Therefore, an exploratory literature study has been executed. All papers were retrieved in in Scopus on January 13th, 2021, using the following search term: `("LSTM" OR "GRU" OR "RNN" OR "recurrent neural network" OR "gated recurrent unit" OR "long short-term memory") AND (‘‘traffic flow prediction’’ or ‘‘traffic speed prediction’’) in TITLE-ABS-KEY`. An overview of all assessed literature, based on the criteria which will be mentioned below, can be found in

Based on this literature survey, several factors could be formulated on which the modelling structures differed. In this section, they are presented from coarse to fine, starting with more high-level design choices and working towards more low-level, specific design choices. Keeping this order in mind, the most important factors identified were:

1. **temporal structure** of input data, as there are two options concerning the selection of times at

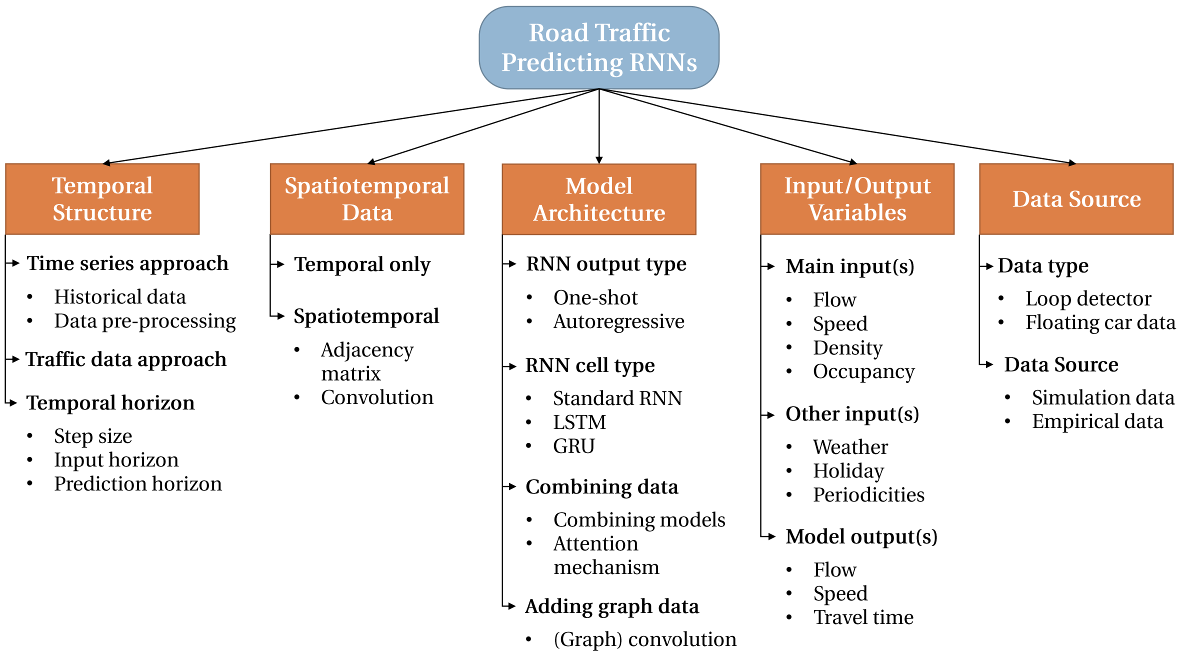


Figure 3.10: Taxonomy of the relevant factors found during the literature analysis.

which spatiotemporal traffic variable data should be included and since the prediction horizon heavily influences the model accuracy;

2. **spatiotemporal data handling**, as, depending on the feature engineering approach, the spatiotemporal dataset can be aggregated over space and time in various different ways;
3. **model architecture**, as some approaches combined other neural network structures with a RNN;
4. the model **input and output variables**, since different traffic variables can be included, each having their advantages and disadvantages;
5. the **data source** of the model, as the data type influences the accuracy of the model

An overview of the most important choices made for each different factor can be found in Figure 3.10. In this section, the components in this figure will be elaborated further.

Since in this master thesis only loop detector data retrieved from a simulation is used, the last factor, 'Data Source' will not be assessed in depth. Therefore, this chapter is structured as follows. First, the temporal structure will be elaborated, explaining both the differences between both approaches as well as the specific choices made on the temporal horizon. Secondly, the difference between temporal and spatiotemporal RNNs will be elaborated. Thirdly, different RNN model architectures will be explained. Fourthly, differences in input and output parameters used in spatiotemporal traffic forecasting neural networks will be explained, as well as the possibility to include other data of road networks in the model inputs. Finally, a summary will be given of the main findings and research gaps found upon analysing the literature.

3.4.1. Temporal Structure

Within the literature analysed, two different approaches could be identified for the selection of the times at which data, namely a more statistical *time series* approach and a more *traffic data* oriented approach.

The statistical time series approach relies on inputting historical values, which are then used to extract periodicities in the data and use these for the prediction task. This is similar to how (statistical) time series analysis models like ARIMA work, which have also been used on traffic forecasting tasks and are often used as benchmark models for AI-based approaches. Methods using this approach (i.e. [16, 45–47]) used historical data belonging to the same time as the prediction, often a combination of data from previous days or from the same weekday but different weeks. For example, if traffic was predicted for a Monday on 09:00, they used traffic data on 09:00 from both the last 7 days as well as all Mondays of last month. Some authors only included historical data from the same weekday, to compensate for daily differences in traffic flow. Besides inputting historical data, some approaches also included time series decomposition or detrending. Zhao and Zhang [48] first detrended flow data (using PCA) and trained an LSTM model on the resulting residuals. Combining the residual data with the trends then resulted in the flow prediction. Asadi and Regan [49] used time series decomposition to split the data in residuals, long term patterns and periodic patterns and trained a different model structure on each of the data types, after combining the results to get the final prediction.

For the traffic data oriented approach, only short-term traffic data is used as input and the goal of the model is to extrapolate this data by fitting functions similar to traffic flow theory. The input of the data consists of a spatiotemporal dataset of one or more traffic variables over a shorter time period, often between 30 minutes and a couple of hours, and will predict traffic from 5 to 60 minutes in the future. The idea is that shorter-term flow and speed changes are a better predictor for future traffic than historic traffic variables. For a RNN, the idea is that by training the model, it will implicitly learn the relation between input data and the correct prediction. One could almost draw parallels with a traffic model approach, where you explicitly feed traffic data and road characteristics in the model which are used in combination with a traffic flow theory based modelling approach to model what the traffic will be in the network. The big difference is that in a RNN, both are not explicitly formulated.

Polson and Sokolov [50] data analysis to assess correlations of both historical data as well as more recent observations. They concluded that recent observations, obtained within 40 minutes, are stronger predictors than historical values in the range of one day, stating that a more powerful model can be obtained when using recent observations. Therefore, this traffic data oriented approach will be used within this thesis instead of a time series approach and the next sections will therefore elaborate more on the choices which can be made using this approach.

After discussing the two main general temporal approaches, let us focus on the *temporal horizon*, which mainly depends how far in the future one wants to predict traffic variables. Although one wants to make predictions over very long time interval, there are some theoretical limitations which prevent doing so. First, there are theoretical limitations on the predictability of traffic as missing (input) information and uncertainties in the traffic process prevents the creation of accurate predictions over long temporal horizons. Therefore, one needs to assess what data is available for making a prediction and which prediction horizon will guarantee a reasonably accurate prediction given this dataset. Secondly, some limitations are governed by the neural network structure, as model limitations might prevent the model to fit the correct function which might be theoretically possible.

On a more practical note, there are three different parameters which define the temporal dimensions of the model input and output, namely the temporal interval of the dataset, the input sequence length and the output sequence length. The **temporal interval** of the dataset is the difference of time between two consecutive measurements. The interval ranges from 30 seconds to 1 hour, although interval sizes of 2 or 5 minutes are most common [29, 43, 44]. Most of the time, this is governed by the source of the data. In many researches a PeMS dataset is used, which consists of highway loop detector data aggregated over 5 minutes from highways in California. Studies which rely on GPS data of probe vehicles (such as taxis) tend to have higher temporal intervals, mostly in the range between 15 to 60 minutes, as aggregating data on a smaller interval can result in very noisy input data. The input and output sequence length depend on the chosen prediction horizon and the interval of the data, as

$$l_{seq} = \frac{horizon}{\Delta t} \tag{3.18}$$

In general, one only wants to use the most relevant data as input data, as more recent data is far more relevant for making a prediction compared to older data. Having more data than necessary can create an unreasonably large sequence length in which it is hard to capture temporal dependencies, making it hard to train the model. Another disadvantage is that larger sequence lengths lead to fewer data samples when using the same dataset, which can reduce the dataset available to train and evaluate the model. Therefore, the input sequence length has to be chosen carefully.

The first RNN models trained for traffic forecasting (for example [51–55]) used one step ahead prediction, resulting in an output sequence length of 1. Many other RNN models make use of a fixed input and output sequence length. However, not many papers explicitly assess their choice in input and output sequence length. However from a traffic management point of view, Oh *et al.* [15] suggest that a prediction horizon of 15-30 minutes is needed in order to anticipate on the results of the forecast. Xue and Xue [56] used an LSTM RNN and data with a temporal interval of 5 minutes, input sequence lengths ranging from 2 (10 minutes) to 12 (60 minutes) and a one-shot output with a shift ranging from 1 (5 minutes) and 4 (20 minutes). For the input sequence length, longer input sequences generally resulted in better predictions, although results between lengths were fluctuating. For the output shift, a 15-minute ahead prediction proved to be a clear optimum and surprisingly, the 5-minute ahead prediction performed worse compared to the 10-minute ahead prediction. On these aspects, no clear trends could be observed. Li *et al.* [57] used input sequence lengths of 20, 30, 40 and 50 entries, having a dataset with a 5 minute interval. They stated that a sequence length of 40 entries lead to the best results. Mackenzie *et al.* [46] predicted traffic conditions 1, 3, 6, 9 and 12 steps ahead, using a data interval of 5 minutes. However, no clear trend could be found assessing the results with a varying prediction horizon for the two analysed detector locations and the errors varied greatly between the two locations. Zhang *et al.* [58] also used data with an interval of 5 minutes, used a 12-step input sequence (1 hour) and a 3, 6 and 9 step ahead prediction. In this case, one could clearly see predictive performance deteriorating with a higher prediction horizon as the 3-step prediction performed much better than the 6 and 9 step prediction. This has also been confirmed by Cao *et al.* [45], which also showed that increasing output sequence lengths resulted in a decrease in prediction performance.

3.4.2. Spatiotemporal Data Handling

As has been mentioned before, predicting traffic flow or speed can be seen as a spatiotemporal prediction task, since traffic flows and speeds propagate over a road network with certain speeds over both space and time. Since the future traffic flow or speed on a road segment therefore depends on the previously observed traffic flow or speed on adjacent links, it is important that a model can consider both spatial as well as temporal correlations in the data for its prediction task. Many traffic forecasting papers using RNNs only focus on temporal correlations (i.e. [51, 52, 59–63]) and therefore have as input flow and/or speed data of one road segment as input data.

However, others also focus on incorporating spatial correlations [64] by including data from related road segments, and do so in several ways. Kang *et al.* [65] predicted traffic flows at a single detector location using several traffic variable combinations from several detector location combinations as input, ranging from zero to three upstream and downstream road segments. They found the highest prediction accuracy was achieved using data from all three upstream and downstream road segments. [45] used both speed data from one upstream and downstream road segment, as well as speed data from alternative routes to predict traffic speeds on one road segment. Others use input data from several road sections and also predict traffic on all the input sections. Zhang *et al.* [58] and Mihaita *et al.* [66] both used a Speed-to-Vector (Speed2Vec) data embedding mechanism, in which temporal sequences of several spatially related nodes can be encoded. By building a two-dimensional matrix, where each matrix entry consists of a speed sequence of one segment at one time step, the spatiotemporal data is coded in an efficient way. Among others, Wu *et al.* [16], Polson and Sokolov [50], Cui *et al.* [67], Yu *et al.* [68], also use a similar data encoding mechanism to predict traffic flow/speeds on several sections using spatiotemporal input data and a RNN. Since their neural network configurations outperformed other neural networks, one can conclude that RNNs can benefit from using spatiotemporal input data.

Besides only using spatiotemporal data, there are also modelling approaches which try to incorporate the road network graph structure into their neural network and therefore explicitly encode the spatial relationship between the spatiotemporal inputs. By encoding the graph structure in the network, the neural network does not have to derive the spatial structure of the nodes implicitly based on the input-output pairs. This would also make it possible to more accurately predict traffic on road networks with a more complex network topology instead of one single road. This can be done using a combination of an adjacency matrix, which depicts all adjacent road sections of a certain road section, and convolutional operations. Specific methods will be considered in the next section.

3.4.3. Model Architecture

Based on choices made regarding the data in the previous sections, several choices can be made regarding the specific model architecture. Firstly, the *output dimensionality type*, as discussed in section 3.3, was found to be important. For the output dimensionality type, one can choose between the one-shot and sequence-to-sequence model. Most models analysed had a one-shot model structure, although sequence-to-sequence models have also been used. Lu *et al.* [69] compared many different neural network forms for predicting traffic speeds, such as a regular LSTM, a sequence-2-sequence model and a graph LSTM, in which they found that the sequence-2-sequence model and graph LSTM both performed much better. However, no clear assessment has been made on what the causes are of some methods outperforming the others.

Secondly, the RNN cell type used also varies significantly. Most frequently, a LSTM cell is used, sometimes even the bi-directional variant which can use information from both the past and the future. However, since traffic prediction is a linear process, contrary to NLP, it is not expected to benefit from this bi-directional LSTM. A GRU cell is also often used and is expected to perform similar to LSTM [38]. Thirdly, one could also use the 'vanilla' RNN cells, which are expected to suffer from the vanishing gradient problem.

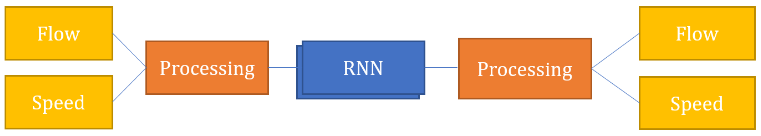
Although in general, the GRU is expected to perform better than LSTM and RNN, this was also confirmed for traffic forecasting. Chattha *et al.* [70] concluded that the errors are lowest for a GRU compared to the other two methods, making it the state-of-the-art RNN cell for traffic prediction.

Thirdly, exotic model combinations can be created when different data have to be merged. This needs to be done when spatiotemporal data from different (fundamental) traffic variables have to be combined, but also when other variables such as historical data or non-traffic variables (i.e. weather, holiday, ...) are added.

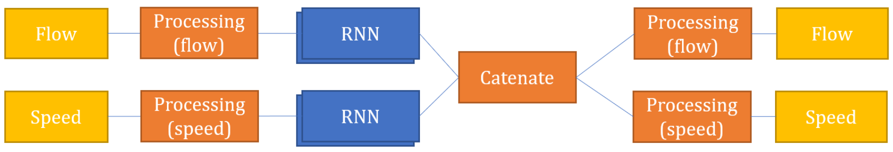
When *multivariate spatiotemporal data*, from both traffic flow and speed, need to be combined, it is important to assess how to correctly integrate these into the model structure. There are little RNN traffic prediction models which combine both datatypes, but one can distinguish two different methods to do this. The first method, shown in Figure 3.11a is to combine both flow and speed data into one input layer which are then processed by the same RNN layers, as has been done by Ma *et al.* [71] and Han *et al.* [72]. Although Li *et al.* [9] does not include flow and speed data, he also proposes that his network structure can handle two or more spatiotemporal data types by including them into the same RNN structure. A big advantage of this method is that it can also make use of correlations between flow and speed data over space and time. The other method, depicted in Figure 3.11b is by processing both data types using separate RNN layers, catenating both results and using a dense layer to combine both results for the prediction, as has been done by [16] an

[73]. Using this method, spatiotemporal correlations can only be detected by the dense layer used on the catenated RNN outputs, possibly reducing the ability of the model to detect more complex spatiotemporal patterns.

A method which is also often used when multivariate data is included is an attention mechanism. This attention mechanism increases the importance of relevant variables whereas it decreases the importance of non-relevant variables. This can especially be relevant if many different parameters have to be incorporated, as it helps prioritising or selecting features in case the feature space is particularly large.



(a) Neural network design with one RNN in total.



(b) Neural network design with one RNN per parameter type.

Figure 3.11: Variants of neural network design for multivariate input.

When *other variables* have to be combined into a neural network, different options also exist. Lv *et al.* [47] used holiday and weather data as inputs, and simply catenated its results with the output of the RNN analysing the spatiotemporal traffic data, using a fully-connected layer to generate the final prediction based on the two data sources. However, Wu *et al.* [16] used an attention mechanism for this purpose, which weighs parameters differently comparing to the values of these parameters.

Lastly, the model structure also changes if the spatial road network structure is encoded explicitly in the neural network, as mentioned in the previous section. There are two methods in order to do this. Wu *et al.* [16], Ma *et al.* [71], Han *et al.* [72] used the convolution operations in a CNN to 'mine' spatial features from the dataset, while using an RNN to extract temporal relations in the data. They either combined both networks serially, feeding the CNN output into the RNN network, or parallelly, where the RNN and CNN results were combined afterwards and processed (by a dense layer) to retrieve the final results. However, since road networks are sparse, using a CNN might be suboptimal as this results in a lot of unused matrix entries. Zhang *et al.* [58] proposes adding an *adjacency matrix* and using a graph attention network on the input data before passing it to the LSTM layer, in order to encode this spatial data. Li *et al.* [74] proposed a Diffusion Convolution Recurrent Neural Network which uses graph convolution in combination with this adjacency matrix to extract spatial dependencies to predict traffic. As road network graphs are often extremely sparse directed graphs, this network structure can better capture spatial dependencies in traffic than a regular CNN can. Variants of these techniques are also used by Cui *et al.* [75] and Bogaerts *et al.* [76].

A promising other method of implementing this graph data in traffic forecasting is Dynamic Graph Convolution Recurrent Neural Network (DGCRNN), as proposed by Li *et al.* [9]. The model makes use of Dynamic Graph Convolution (DGC) to embed the highway graph structure in the neural network so the spatial propagation of traffic-related information can be captured. Introducing a 'receptive field' by embedding a k-walk aggregator within the convolution, information of neighbouring nodes is used when predicting traffic variables on the current node. The steps involved in a DGC module, as depicted in Figure 3.12, are (I) computing the convolutional kernels from several inputs, (II) applying these kernels in a graph convolution and (III) adjusting the output dimension. Smart parameter sharing within the DGC module makes sure traffic properties are captured while simultaneously minimizing the number of trainable parameters. Using this parameter sharing tech-

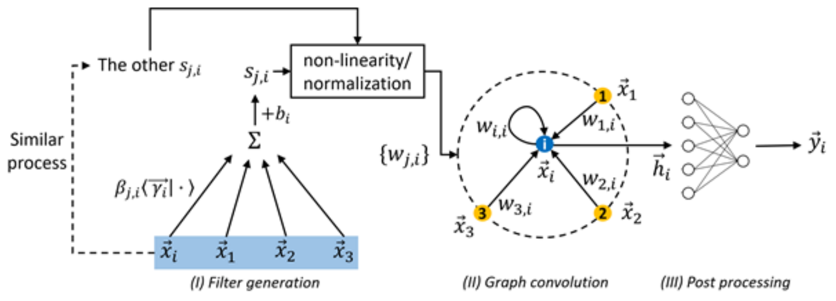


Figure 3.12: Steps in Dynamic Graph Convolution module. Adapted from [9].

nique, the model was able to correctly predict the propagation of shockwaves through a ring road network and can potentially also be used on more complex road networks.

3.4.4. Input and Output Variables to Include

Based on the literature researched, three different categories of input and output variables were identified, which will be treated in this section. First, the traffic related input- and output variables will be elaborated. Afterwards, other (non-traffic) related inputs will be mentioned.

It is important to select the right traffic variables to include in the model inputs and outputs, as these are the main data on which the model is being trained. In chapter 2, both traffic flow and speed were identified as important variables, although it was found easier to detect and classify congestion based on speed data. Since traffic flows could remain quite constant during congestion, traffic speeds are a better indicator for congestion. Besides speeds being more relevant for the classification and analysis of congestion patterns, it can also be used for calculating travel times and delays. Therefore, one could say that predicting speeds is more relevant compared to predicting flows. However, most models developed for short-term traffic prediction were used to predict traffic flow [29, 43, 44] and within this literature study, only 11 of the papers analysed were predicting traffic speeds. Traffic speed prediction might also be harder than traffic flow prediction, as from chapter 2 follows that traffic speeds can increase and decrease rather quickly compared to traffic flows due to these abrupt traffic phase changes. This has also been noted by Polson and Sokolov [50], as they resort to using tanh-layers and l1-regularisation to account for these abrupt changes. Although predicting speeds might result in a lower accuracy compared to flow, the added value of having speed data greatly outweighs the greater complexity of this problem. Travel time would also be a good model output, as proposed by van Lint [27], especially when used in an ITS. However, compared to flow or speed predictions, it would be much harder to prove the validity of the predictions.

Based on the model outputs, model inputs have to be chosen which would be good predictors for these output variables. In most of the proposed neural networks, the input variables are equal to the output variables. However, Kang *et al.* [65] found that when predicting traffic flows, using flow, speed and occupancy data as inputs resulted in a higher model accuracy. However, it is unlikely that adding occupancy data could result in a higher accuracy, as this raw data is used to derive flow and speed data. Other researchers also incorporated flow and speed as input data (i.e. [16, 27, 73]) but did not comment on whether this resulted in an increased accuracy compared to using only one of them.

Besides only using traffic related variables such as flow and speed as input features, some models also use other data sources as input which are believed to affect traffic supply and demand. Zheng *et al.* [77] also used weather data and route structure data, by combining them into an embedding layer. Lv *et al.* [47] also encoded weather data and whether the data were retrieved from a peak hour or holiday. As rainy weather can result in more car traffic demand or a lower road capacity, the availability of alternative routes can prevent congestion from getting worse and traffic during peak hours and holidays differs from regular traffic conditions, the idea is that the neural network can use these correlations between the input and output data to have a better prediction. Although adding this data explicitly has not been researched in this thesis, this might help creating models with a higher generalisability and even are transferable to other roads, as this enables the model to explicitly make a distinction between different roads based on the input variables.

3.4.5. Main Findings and Research Gaps

As within this literature analysis many different smaller conclusions were drawn and research gaps were found, let us conclude with a quick recap containing the main findings and research gaps identified.

As for the *temporal structure*, it was found that the traffic data oriented approach leads to better results compared to the (statistical) time series approach: short-term traffic predictions should only be based on more recent flow and speed data. both the input horizon and prediction horizon were found to be important properties, although little attention is given to its choice. In general, the predictive accuracy decreases with a higher prediction horizon and output sequence length, which is as expected. However, for the input sequence length, no clear trend could be observed. Traffic flow theory might be helpful for determining the right input horizon, as this can lead to recommendations on the temporal horizons which might be relevant from a traffic point of view. A research gap found was a good substantiation on which temporal horizon parameters should be chosen when preparing the dataset.

When assessing the *spatiotemporal data* properties, using a spatiotemporal dataset and model structure greatly improved the accuracy. Adding spatial information in terms of an adjacency matrix or by using convolution operations can be very helpful in case the network structure is complex, such as a dense urban network. This can provide additional information on the way traffic flow and speed information of individual road segments are correlated to each other. However, for simple networks such as single highways, this information has little added value as it can be learnt easily by the neural network.

Assessing the *model architecture* gave more insight into the design choices which can be made when designing these neural networks. GRUs were found to result in the highest accuracy, as well as the autoregressive output type. Several methods were used to combine data from different sources. Overall, it was found that many different model architectures are proposed, but the choice for a particular model architecture often is not substantiated. When many different model types are combined and an attention mechanism is used, it is unclear what the purpose is of these often complex model structures. When CNNs are used to mine spatial features, no substantiation is given on how it will actually do this on extremely sparse network data. Therefore, it would be helpful to break-down a model architecture and test the influence of different design choices on the model accuracy. This incremental improvement of a neural network architecture would lead to a better substantiation of design choices instead of only proposing a new and complex model structure.

When looking at the *input and output variables*, it was found that predicting speeds is more valuable than predicting flows. In order to do so, flow and speed data would be valuable. However, a research gap found was that no research has been done on whether adding flows would result in a better speed prediction compared to only using speed as input data. Mostly, a choice is made regarding the input data, but no assessment on the validity of this choice is given. This is also the case when adding other inputs, such as weather and holiday data. Although the models are compared with benchmark model, no explicit assessment on the added value of including this extra data is given.

Another major research gap that could be identified was that a very limited assessment of the model error was done. Some authors used a boxplot to assess the variation of errors for different hyperparameter combinations (e.g. [67, 72, 78]) or line plots or heatmaps containing predicted flows or speeds (e.g. [58, 75, 79, 80]). However, limited attention is given to interpreting the errors over space and/or time, or to assess the variability of the error over the different days on which traffic is predicted.

Table 3.1: Literature table containing an overview of all assessed literature based on the criteria defined in the literature analysis (part 1).

	Temporal Structure				**Spatiotemporal Data**		**Model Architecture**				**Input/Output Variables**			**Data Source**	
	Time Series	*Step Size (min)*	*Input Horizon (min)*	*Output Horizon (min)*	*ST Data*	*Graph Data*	*RNN/other Type*	*RNN Struct.*	*Attention Mechanism*	*Model Struct.*	*Traffic Input*	*Exogen. Input*	*Model Output*	*Data Type*	*Simulation Data*
van Lint [27]	✗	1	1	1	✓	✗	RNN	1s	✗	Serial	q, u	✗	tt	Loop	✓
Ma *et al.* [81]	✗	2	1	1	✓	✗	LSTM	1s	✗	Serial	q, u	✗	u	Loop	✗
Tian and Pan [51]	✗	5	5,10,...,60	15,30,45,60	✗	✗	LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Fu *et al.* [52]	✗	5	30	5	✗	✗	LSTM/GRU	1s	✗	Serial	q	✗	q	Loop	✗
Guo *et al.* [59]	✗	5	30	30	✗	✗	GRU	AR	✗	Serial	q	✗	q	Loop	✗
Kang *et al.* [65]	✗	5	1	1	✓	✗	LSTM	1s	✗	Serial	q, occ, u	✗	q	Loop	✗
Li *et al.* [74]	✗	5	5	15,30,60	✓	GCN	RNN	1s	✗	Serial	q	✗	q	Loop	✗
Liu *et al.* [53]	✓[a]	0.5	0.5	1	✗	ConvLSTM	Conv+ BI-LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Shao and Soong [54]	✗	5	5	5	✗	✗	LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Wang *et al.* [82]	✗	5	5	15,20,25,30	✗	✗	Bi-LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Yu *et al.* [83]	✗	2	30	2,4,6, 20,40,60	✓	CNN	CNN+LSTM	AR	✗	Serial	u	✗	u	Loop	✗
Zhao *et al.* [78]	✗	5	1	15,30,45,60	✗	✗	LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Yu *et al.* [68]	✗	5	60	15,30,45	✓	GCN	STGCN	1s	✗	✗	u	✗	u	Loop	✗
Chen *et al.* [60]	✗	5	30,60,120, 180,240,300	30,60,120, 180,240,300	✗	✗	LSTM	1s		Serial	q	✗	q	Loop	✗
Cui *et al.* [67]	✗	5		5	✓	✗	LSTM	1s	✗	Serial	u	✗	u	Loop	✗
Lv *et al.* [47]	✓[b]	10,15, ...,30	100,150, ...,300	100,150, ...,300	✓	LC	LSTM	1s	✗	Parallel	u	✓[1]	u	FCD	✗
Qu *et al.* [73]	✗	15	90	15	✓	✗	LSTM	1s	✗	Parallel	q, u	✗	q	Loop	✗
Tian *et al.* [84]	✗	15, 60	15, 60	15,60	✗	✗	LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Wu *et al.* [16]	✓[c]	5	105	5,10,...,45	✓	CNN	GRU+CNN	1s	✓	Parallel	q, u	✗	q	Loop	✗
Xue and Xue [56]	✗	5	10,15,..., 60	5,10,15,20	✗	✗	LSTM	1s	✗	Paralel	q	✗	q	Loop	✗
Zhao and Zhang [48]	✓[d]	3	2880	1440	✗	✗	LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Dai *et al.* [85]	✗	5	(corr.)	5,10,15,20	✓	✗	GRU	1s+AR	✗	Serial	q	✓[2]	q	Loop	✗
Du *et al.* [86]	✗	15	45,90	45,90	✗	✗	LSTM	s2s	✓	Serial	q, u, tt	✗	q	Loop	✗
Essien *et al.* [87]	✗	5	60	120	✗	✗	Conv- LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Guo *et al.* [88]	✗	5	5-100	5	✗	CNN	CNN+GRU	1s		Serial	q	✗	q	Loop	✗
Han *et al.* [72]	✗	5	1440	5	✓	CNN	CNN+LSTM	1s	✗	Parallel	q	✗	q	Loop	✗

Variables: occ = detector occupancy, q = flow, tt = travel time, u = speed.
RNN Structure: 1s = one-shot, AR = autoregressive, s2s = sequence-to-sequence.
Time Series: [a] only periodicity (day/week); [b] last week, last day; [c] statistical processing; [d] residuals, long-term patterns, periodic patterns
Exogenous Input: [1] periodicity (day/week), weather, holiday; [2] correlation analysis

Table 3.2: Literature table containing an overview of all assessed literature based on the criteria defined in the literature analysis (part 2).

	Temporal Structure				**Spatiotemporal Data**		**Model Architecture**				**Input/Output Variables**			**Data Source**	
	Time Series	*Step Size (min)*	*Input Horizon (min)*	*Output Horizon (min)*	*ST Data*	*Graph Data*	*RNN/other Type*	*RNN Struct.*	*Attention Mechanism*	*Model Struct.*	*Traffic Input*	*Exogen. Input*	*Model Output*	*Data Type*	*Simulation Data*
Kim *et al.* [89]	✗	15	150, 225	15	✓	GCN	LSTM	1s	✗	Parallel	u	✗	u	Loop	✗
Li *et al.* [62]	✗	5	5-45	5	✗	✗	LSTM	1s	✗	Serial	q	✗	q	ANPR	✗
Li and Wu [90]	✗	5	5	5	✗	✗	LSTM	1s	✗	Serial	q, occ, u	✗	q, occ, u	Loop	✗
Li *et al.* [57]	✗	5	100,150, ..., 300	5	✓	GCN	GCN+LSTM	1s	✓	Serial	q	✗	q	Loop	✗
Lu *et al.* [69]	✗	5	24	12	✓	GCN	GNN+LSTM	s2s	✗	Serial	u	✗	u	Loop	✗
Luo *et al.* [91]	✗	5	30	5	✓	KNN	KNN+LSTM	1s	✓ (KNN)	Serial	q	✗	q	Loop	✗
Mihaita *et al.* [66]	✗	3	3-90	3, 6, . . . , 30	✓	✗	LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Mou *et al.* [92]	✗	2	20, 40, . . . , 160	2	✗	✗	LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Peng *et al.* [93]	✗	2	2	2	✗	✗	LSTM	1s	✓	Serial	q	✗	q	Loop	✗
Wang *et al.* [94]	✗	5	5	5	✗	✗	LSTM	1s	✗	Serial	q	✗	q	Loop	✗
Wei *et al.* [95]	✗	5	5	5	✓	✗	LSTM	1s	✓ (AE)	Serial	q	✗	q	Loop	✗
Xiangxue *et al.* [96]	✓[e]	5	5	5	✓	✗	LSTM	1s	✗	Serial	u	✗	u	FCD	✗
Yang *et al.* [97]	✗	15	480	15,30,45,60	✗	✗	LSTM	s2s	✓	Serial	q	✗	q	Loop	✗
Zhang *et al.* [58]	✗	5	60	15,30,45	✓	✗	LSTM	1s	✓ (GAT)	Serial	u	✗	u	Loop	✗
Zheng *et al.* [98]	✗	5,15	15,45	5,15	✓	CNN	LSTM+ DAE+CNN	1s	✗	Serial	q, u	✗	q	Loop	✗
Zheng *et al.* [77]	✗	20	120	20	✓	CNN	Embed+ CNN+LSTM	1s		Serial	tt	✓[3]	tt	Loop	✗
Asadi and Regan [49]	✗	5	30	15,30,45,60	✓	CNN	LSTM+ CNN+AE	s2s	✗	Serial	q, u, occ	✗	q	Loop	✗
Bogaerts *et al.* [76]	✗	5	20	25,50, . . . , 225,240	✓	CNN	CNN-LSTM	AR	✗	Serial	u	✗	u	FCD	✗
Chen *et al.* [99]	✗	15	15	150,300, 450,600	✗	✗	AE GRU	1s	✓ (AE)	Serial	q	✗	q	Loop	✗
Chen *et al.* [100]	✗	5	50	15	✗	✗	LSTM	1s	✓	Serial	q	✗	q	Loop	✗
Chen *et al.* [101]	✗	5	5,10,. . . ,25	5	✓	CNN	CNN+LSTM	1s	✗	Serial	q, occ	✗	q	Loop	✗
Lee and Lin [102]	✗	1	1	1	✓	✗	LSTM	1s	✗	Serial	u	✓[4]	u	Loop	✗
Li *et al.* [9]	✗	2,5	30,60	20,30	✓	DGC	GRU	AR	✗	Serial	u	✗	u	Loop	✗
Lu *et al.* [103]	✗	5	120	60	✓	GCN	GRU	AR	✗	Serial	u	✗	u	Loop	✗
Ma *et al.* [71]	✗	5	10,15,. . . ,55	10,15,. . . ,55	✓	CNN	CNN+LSTM	1s	✗	Serial	q, u	✗	q, u	Loop	✗
Zhou *et al.* [104]	✗	15	300	15	✓	✗	LSTM+SVR	1s	✗	Serial	q	✗	q	Loop	✗
Zhu *et al.* [105]	✗	15	15	15	✓	GCN	RNN	1s	✗	Serial	q, u	✗	q	FCD	✗

Variables: occ = detector occupancy, q = flow, tt = travel time, u = speed.
RNN Structure: 1s = one-shot, AR = autoregressive, s2s = sequence-to-sequence.
Time Series: [e] residuals, long-term patterns, periodic patterns
Exogenous Input: [3] precipitation, temperature, route structure (embedding); [4] temperature, precipitation, holiday, day, reachability

4 Preliminary Data Analysis

The goal of this preliminary data analysis is to gain more insight on traffic characteristics by only looking at the data of a road. Since this is the only information a neural network can base its predictions on, including maybe a graph structure, it is important to have an idea what data contains relevant information for traffic prediction. Besides, this is an important step to formulate design requirements and choices for the neural network. By doing a correlation analysis and determining data aggregation levels, design choices regarding input data and node specification can be made. If two roads with similar layouts have big differences in traffic-related parameters (such as the fundamental diagram) which influence predictions, it might be relevant to feed these into the neural network directly. Since the model should be able to estimate traffic on various similar roads, the input data should also contain information distinguishing location specific properties. Roughly, the following requirements can be formulated:

- the data should contain information which can predict at least traffic speeds correctly
- the data provided should capture location specificities in traffic so it can correctly estimate traffic on a wider variety of roads

For the first requirement, it is important to look at the relation between flow and speed over space and time and see if there are patterns and relations which can help predicting traffic. For the second question it is important to research which data can be used to accurately predict traffic on a range of roads.

In order to meet both requirements, the goal is to find meaningful information and correlations over space and time for flow and speed data. When looking over space, spatial regions with similar traffic conditions might also have a a similar spatial correlation and may be a relevant spatial aggregation level for the neural network. It would be likely that these coincide with the location of discontinuities, as these discontinuities are likely to change traffic conditions and the prediction results. When looking over time, it might be useful to aggregate temporal information to both reduce the amount of data needed for depicting the underlying pattern as well as preprocessing the data so the predictions are more reliable. Another important goal of this data analysis is to assess the accuracy of the data and the viability of a data-based approach, by checking if the data quality is high enough.

For this preliminary data analysis, data was retrieved using the Mirrors-NDW tool from DiTTlab which provides traffic data from loop detectors on Dutch highways maintained by Rijkswaterstaat. This tool filters the raw data using the adaptive smoothing method (ASM) and provides flow *per lane* (in veh/h) and speed (in km/h) averages (in integers) over an arbitrary space and time with a resolution of $\Delta x = 100$ m and $\Delta t = 0.5$ min. With the elementary traffic relation $q = k * u$, density can be calculated. For this data analysis, two roads have been analyzed using data from Monday 18th of November to Friday 22th of November between 14:00 and 19:00, to capture both congested and uncongested situations. Thursday 21th of November has been excluded for both roads, due to '99999'-values probably resulting from a system-wide malfunction. Two highway sections have been reviewed, namely A20 and A4. Firstly, the results of the A20 will be thoroughly assessed. Secondly, the A4 analysis will be used to validate the previously acquired results. Lastly, conclusions will be drawn from the data analysis.

Pleas note that in the heatmaps, the driving direction if from bottom to top. In general, 0 km corresponds with the most upstream segment of the road.

4.1. A20 (knp. Terbregseplein to knp. Gouwe)

This stretch of the A20 highway is 12.3 km long and runs from Terbregseplein interchange in Rotterdam, connecting the A20 with the A16, to Gouwe interchange in Gouda. It consists of four exits, of which one leads to a service station, and a lane merge where the number of lanes changes from three to two lanes. A schematic view of the road layout and discontinuities (such as on-ramps, off-ramps and lane merges) can be seen in Figure 4.1 and a satellite image with its surroundings in Figure 4.2. It starts off as a three-lane road, but continues as a two-lane road after a lane merge in the middle of the section. The speed limit is 100 km/h for the first 1300 m, after which it increases to 120 km/h. Congestion occurs frequently at the lane merge point, which is situated 100 m away from the off-ramp of exit 17. Another known bottleneck is exit 18 (Moordrecht) and the A12 itself, which cause congestion on one and two days respectively of the four analyzed days. Since Gouwe interchange is a partial interchange, traffic entering from A20 can only go east on A12 (or has to use exit 18), so no turbulence due to lane sorting can be expected on the last kilometers. However, between exit 18 and knooppunt Gouwe, the road an S-curve with decreasing radii, which could create a change in driving behaviour.

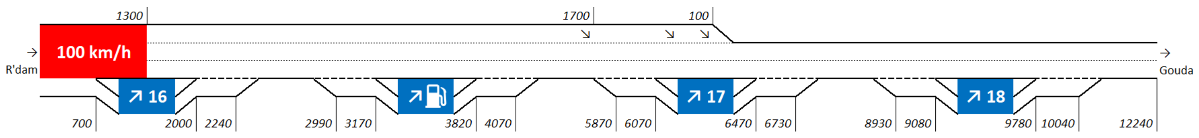


Figure 4.1: Schematic overview of A20 (with distances in m)

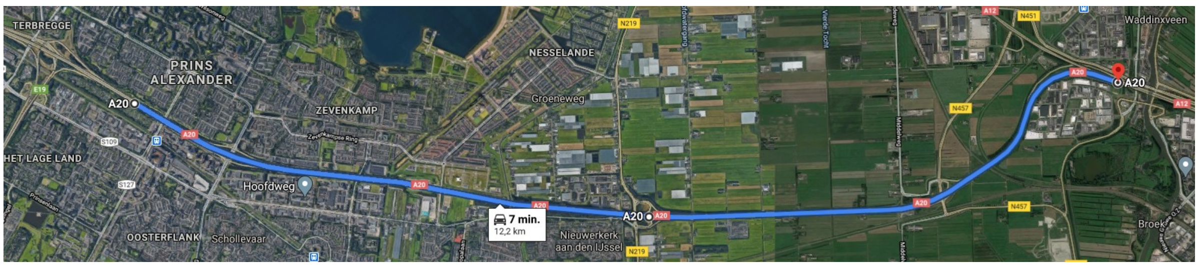


Figure 4.2: Aerial view of A20 (driving direction: left to right)

This data analysis is structured as follows. First, traffic variables were aggregated and analysed over space and time as well as heatmaps of the variables. Secondly, these variables were used in standard deviation and difference plots, as well as to showcase the empirical fundamental diagrams of the road. Lastly, the results of a correlation analysis will be discussed.

4.1.1. Traffic variables over space and time

First, to get an indication of the road characteristics, plots were made which showed the mean flow, speed and density per spatial road segment, both for congested and free-flow traffic conditions. Density has been calculated with the equation $q = k * u$. As this equation only holds for homogeneous and stationary traffic conditions and should be interpreted with caution when many accelerations and decelerations happen.

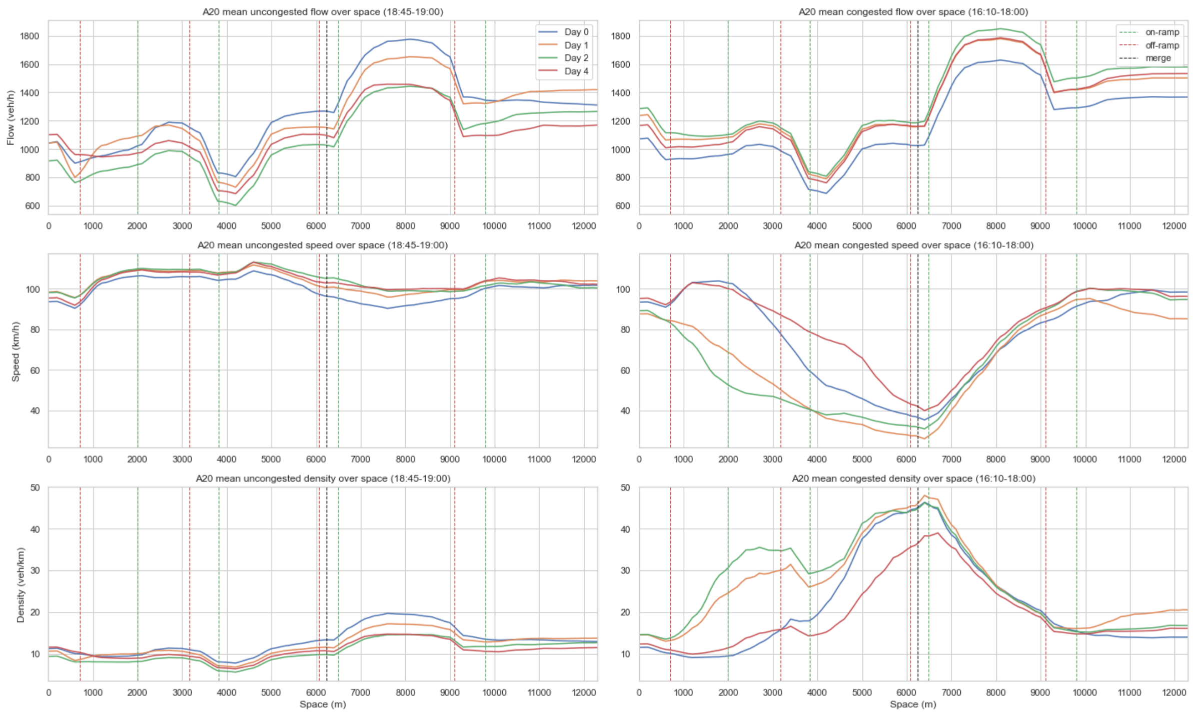


Figure 4.3: Average traffic flow (top), speed (middle) and density (bottom) over space for A20 during free-flow (left) and congested (right) conditions

When looking at the mean values during **free-flow** situation, which consists of all pictures on the left of Figure 4.3, one can see that each day approximately follows the same trends albeit the magnitude of the values differs. In general, off-ramps cause dips in flow and density whereas on-ramps causes traffic flow to increase. The peak in flow and density after the lane merge is approximately 1.5 times higher than before the lane merge, as expected, and results in a slightly lower speed. However, there are two phenomena which cannot be understood when comparing the patterns with the road layout. Firstly, at 4 km, there is a sudden drop in flow and density which cannot be explained since (a) the on-ramp at this location should cause an increase in traffic, (b) the magnitude of this dip is higher than expected at a rest area and (c) since it is not caused by a drop in speed, as density decreases too. Secondly, one can see that lane density is the highest at 7.5 km, which is well after the lane merge (6.2 km) and even the on-ramp (6.7 km), which would be the two main contributors to a flow increase.

When looking at the mean values during **congestion**, one can see that, compared to free-flow conditions, the flow values remain similar, but speeds decrease and densities increase. The lowest speed and highest density is measured between the lane merge and on-ramp (approx. 6.4 km). Looking upstream of this point, speeds are slowly decreasing, as further upstream congestion is observed over a smaller period of time, causing a higher mean speed there. After inspection of the heat maps, it was concluded the jam waves resulting from the lane merge spread not further than the off-ramp of exit 16 (at 2.2 km), so the temporary decrease in speed here can also be caused by the queue tail signalisation. Looking downstream of this point, one can see speeds increase and densities decrease as traffic is slowly driving out of the jam wave due to the high density. After the on-ramp of exit 18, at 9.8 km, flow, density and speed start to stabilise again. Meanwhile, in the density plots, at 4 km, the same drop in density as in the uncongested situation can be seen.

After analysing the aggregated graphs, a heatmap provides a good way to research flow or speed data for a whole day in more detail. Since comparing heatmaps of four days would make this chapter quite lengthy without adding much information, we zoom in on the third day of the dataset, the 20th of November 2019. This day shows a simple and typical congestion pattern for the road with jam waves originating from the lane merge bottleneck, although congestion is quite high compared to the previous days. When looking at the traffic flows in Figure 4.4, one can observe the general patterns captured before, such as the higher flow between the lane merge and the next off-ramp (6,600-9,200 m) and the lower flow between 3,700m and 4,700m. However, flows are strongly fluctuating over time, creating a fine wave pattern which is propagating with free-flow speed (when uncongested) or a shockwave speed of approximately -20 km/h (when congested). Looking at the speed heatmap in Figure 4.5, the congestion patterns become even more clear. One can also see that both after 1,000 m (when the maximum speed increases) and after the on-ramp at 10,000 m, traffic speeds increase. Indeed, traffic drives slowly out of the congestion after passing the lane merge bottleneck and the jam waves spill back approximately until the most upstream off-ramp at 700 m.

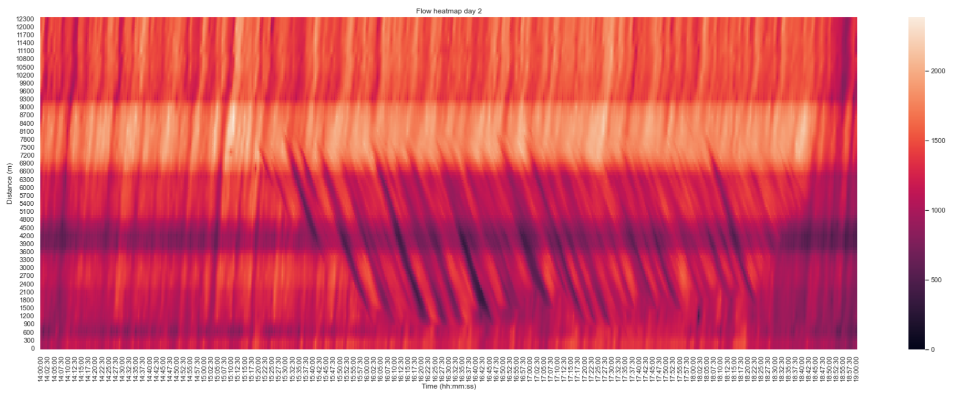


Figure 4.4: Heatmap showing flow over space and time for A20 on 20th of November.

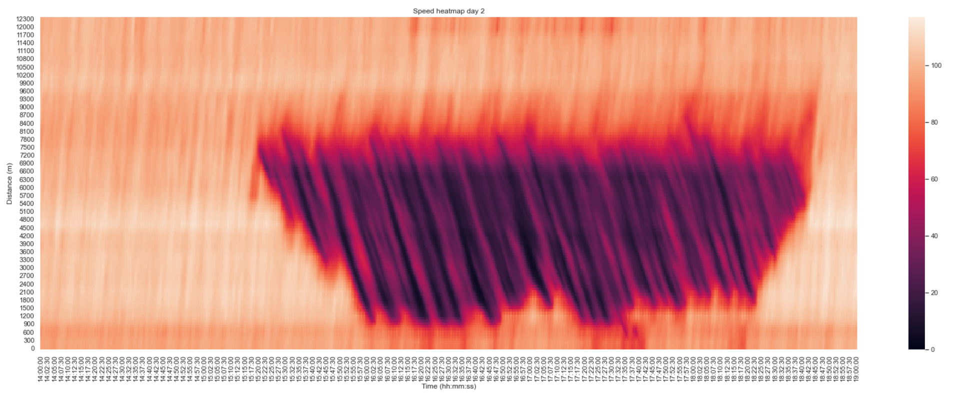


Figure 4.5: Heatmap showing speed over space and time for A20 on 20th of November.

4.1.2. Fundamental Diagram

The relationship between flow, speed and density, also called a fundamental diagram, contains a lot of information on traffic. Since the fundamental diagram shape can only be reproduced using stationary and homogeneous traffic conditions, the graphs here contain a lot of 'noisy' data due to accelerations and decelerations. In an attempt to give cleaner relations, both the three-lane road section and the two-lane road section have been plotted separately. In this paragraph, the scatter plots of flow and density, as shown in Figure 4.6, will be analyzed. Density is calculated using $k = q/u$. When looking at the three diagrams, one can see the points are ordered on straight lines, which is because speeds are stored at integers. The maximum flow (approx. 2300 veh/h), between the free-flow and congested branch, and jam density (approx. 140 veh/km) fall within expectations, as well as the general shape of the fundamental diagram and the resulting shockwave speed (-16 km/h (3-lane section) and -19 km/h (2-lane section)). However, the maximum flow per lane on the three-lane is significantly lower (1791 veh/hr), presumably due to the lane merge bottleneck just before the two-lane section which causes traffic to break down earlier. One can also see that on the three-lane section, much lower flows and densities are measured compared to the two-lane section and the flows and densities on the 'congested' branch are much more spread out. This means there is far

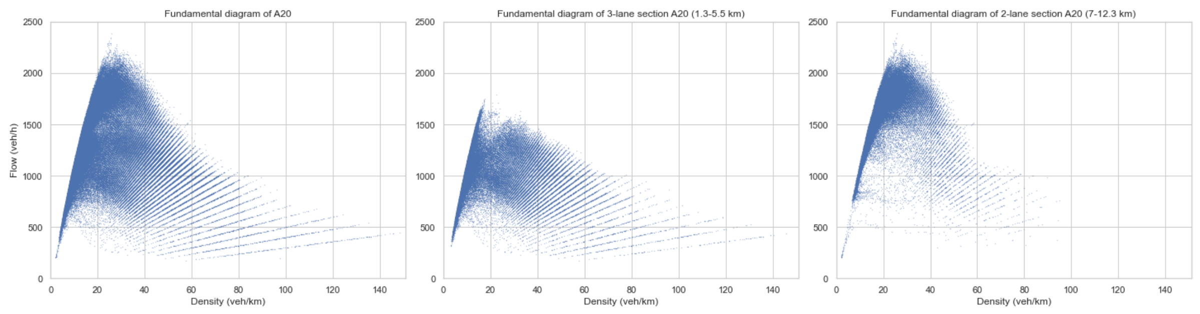


Figure 4.6: Fundamental diagram (u,q) for A20 plus two sections

more congestion on the three-lane section, which makes sense as most congestion occurs upstream of the bottleneck. When analyzing speeds, which equal to slopes in the fundamental diagram, the two-lane section has a lower top speed than the three-lane section, which probably is caused by a higher traffic flow in combination with the lane merge and curves at the end of the section. Within the combined fundamental diagram of the whole road, the shapes of both the two-lane and the three-lane section are visible. One can see a higher point density near the road capacity, from the on-capacity nearly-congested flow after the bottleneck on the two-lane road, but also a higher point density on the congested branch around 1000-1500 veh/h flow.

4.1.3. Correlation analysis

As explained in the introduction, a neural network essentially 'learns' (non-linear) correlations between its inputs, which it then uses to predict future values. Therefore, a preliminary correlation analysis can give an idea of the correlations are relevant as parameters for the fundamental diagram. Furthermore, it also gives an idea of the aggregation level of the input data: when correlations are stable over a period in space and time, this period will be a good aggregation level for the neural network.

For the correlation analysis the Pearson correlation coefficient (r) has been calculated, see Equation 4.1, which is widely used and simple to calculate and interpret.

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}} \tag{4.1}$$

Although this correlation coefficient can only test linear relations between two variables, it is a good first measure for correlations. A correlation of +1 indicates a perfectly *positive* linear correlation, a correlation of -1 a perfectly *negative* one. For easy analysis, correlation coefficients have been calculated for the whole dataset at once and plotted on a heatmap. Correlations over space or time were found by rolling over the respective variable using various window sizes, to detect finer correlations as well as filter out effects due to noise. Window size 5 (500 m/2.5 min) was found to give the best interpretable results, depending on the purpose. When interpreting the heat maps, the correlation coefficient over the space/time interval $[x, x+a]$ (with a = window size) is shown at point $x+a$, as this coincides how neural networks use input of previous points to calculate the next one. In this paragraph, first the correlations over space for both flow and speed will be discussed, followed by the correlations over time. The data used was from the A20 at the 20th of September 2019.

Correlation over space

The correlation coefficients over space in general have a higher percentage of significant values ($p < 0.1$, two-sided) than over time, with 69% (speed) and 74% (flow) significant values using a window size of 5 measurements. This indicates useful and strong correlations exist of which a neural network can make use of.

Flow

In general, the flow patterns seem to be quite persistent, with boundaries in space where flow increases or decreases being consistent over time. The window size was kept at 5 steps, as this resulted in noise-free observations, which can be seen in Figure 4.7 During free-flow conditions, which can be noticed on the edges of the heatmap (15:00-15:20 and 18:40-19:00), many fine patterns emerge which often cannot be explained well. However, in general, several 'zones' can be found which are separated by changing correlations at the following points:

1. *800 m:* represents a flow increase at approx. 600 m, although it coincides with the off-ramp of exit 16, a flow increase would not be expected.
2. *3,000 m*: represents a flow decrease at approx. 2,800 m, which is shortly before the off-ramp of the rest area.
3. *4,300 m*: represents to a flow increase at approx. 4,100 m, this coincides with the on-ramp of the rest area.
4. *6,000 m*: this border is less clear, but the flow decrease at approx. 5,800 m coincides with the off-ramp of exit 17; turbulence from the lane merge downstream could make this border less clear
5. *6,500 m*: represents flow increase at approx. 6,300 m is probably caused by the lane merge, although it is located slightly more upstream
6. *7,500-8,400 m:* border of decreasing flow of which the location fluctuates strongly and is situated between the off-ramp of exit 17 and the on-ramp of exit 18.
7. *9,700 m*: represents a flow increase at approx. 6,500 m, which is a bit upstream of the on-ramp of exit 18.

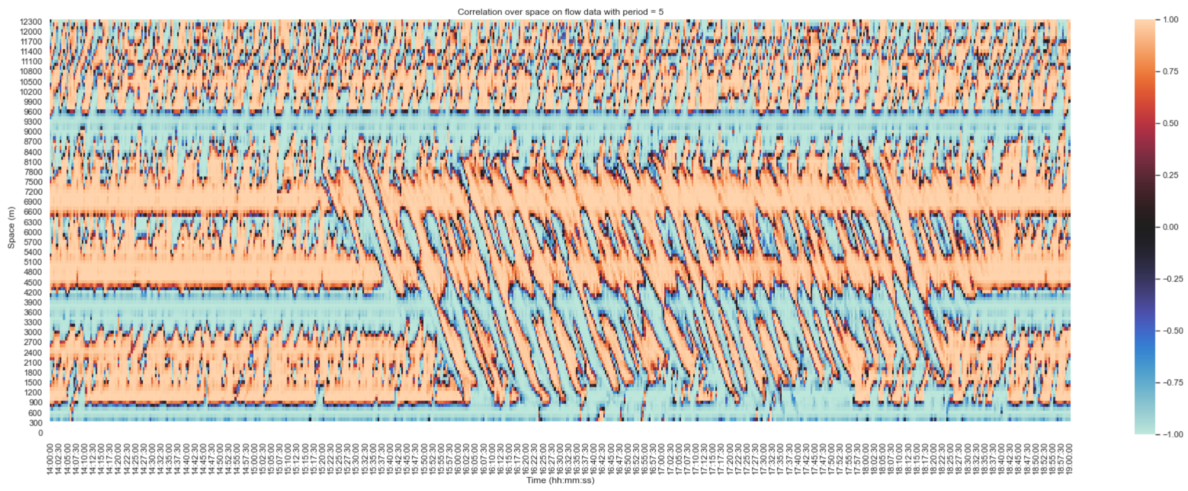


Figure 4.7: Heatmap showing Pearson correlation coefficients of flow over space (window size = 5) for A20 on 20th of November

One can conclude that in general, flows decrease after off-ramps and diverging points and increase after on-ramps and merging points. However, there are a few peculiarities in the data. Firstly, at the off-ramp at 800 m, flow is increasing instead of decreasing. Since the merging section of Terbregseplein interchange also leads to this off-ramp, the increase in flow might be due to a decrease in turbulence after this point. Secondly, there is a very clear change in flow caused by the rest area, which is quite peculiar; as mentioned earlier, there is a very sharp decrease and increase in flow around this point, although it is uncommon for a rest area to attract this much traffic. Thirdly, there is a very irregular flow pattern after 10,000 m, which is probably explained by the sharp S-curve which causes people to decelerate. Lastly, some borders between opposing correlations are highly irregular, for example at 6,000 m and especially between 7,500-8,400 m, from which the last one cannot be explained correctly by a discontinuity on the road.

During congestion, a pattern which looks like congestion waves emerges, although the general flow pattern during uncongested situations still is present. Although congestion clearly influences the spatial correlations, the pattern remains more stable than could be expected as it is only incidentally interrupted by these stop-and-go waves. This could mean that flow increases and decreases remain stable but only their absolute values are influenced by congestion.

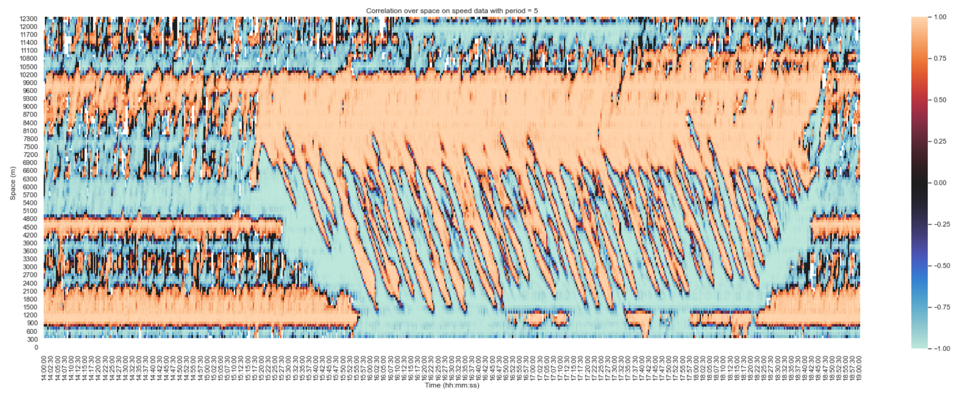


Figure 4.8: Heatmap showing Pearson correlation coefficients of speed over space (window size = 5) for A20 on 20th of November

Speed
In general, the correlations found in the speed data are less stable over time as the patterns drastically changes when congestion occurs. Maybe this is since speed is more strongly related to congestion. During congestion, speed upstream of the bottleneck will initially drop and then fluctuate due to the fluctuating speed in the to stop-and-go waves. Downstream of the bottleneck, speed will increase slowly as traffic drives out of the bottleneck with a critical flow. This completely overrides the structure during free-flow compared to flow. The few white (NaN) values might have something to do with the fact that speeds are measured on integer levels. Under stable traffic situations, having five consecutive equal measurements gives a division by 0 error when calculating the correlation coefficient.

During uncongested situations, several borders emerge at the following locations:

1. *800 m, 6,600 m, 10,000 m:* have similar situations explained in the 'flow' part
2. *2,200 m*: which represents a speed decrease at 2000 m, which is the location of the on-ramp of exit 16
3. *Between 2,200 and 4,200 m:* a highly irregular pattern occurs, which is just as hard to explain as the traffic situation in this section
4. *4,200 m:* traffic speeds increase from approx. 4000 m, which coincides with the on-ramp of the rest area although a speed reduction due to extra flow would be expected,
5. *4,900 m:* speeds start to decrease from approx. 4700 m, which does not represent any discontinuity on the road
6. *7,500-9,000 m:* irregular pattern, just downstream of the lane merge.

Several situations, namely situation 4 to 6, can be explained by looking at flow and speed data, as situation *4* and *5* coincide with a decrease and increase in flow respectively and in situation *6*, there is a sharp increase in flow. However, as mentioned before, they occur at places where there are no discontinuities on the road although they occur constantly during free-flow.

Correlation over time
As mentioned before, the number of significant correlations is over time less compared to over

space. With the same conditions ($p < 0.1$, two-sided, window = 5), only 52% (flow) and 54% (speed) of the observations are significant. This is probably since over time, there are much more 'borders' between positively and negatively correlated zones which show no significant correlation. However, since the correlations found within each wave are significant, this may still contain useful information. When looking at correlation over space and time, one can see that this correlation does not

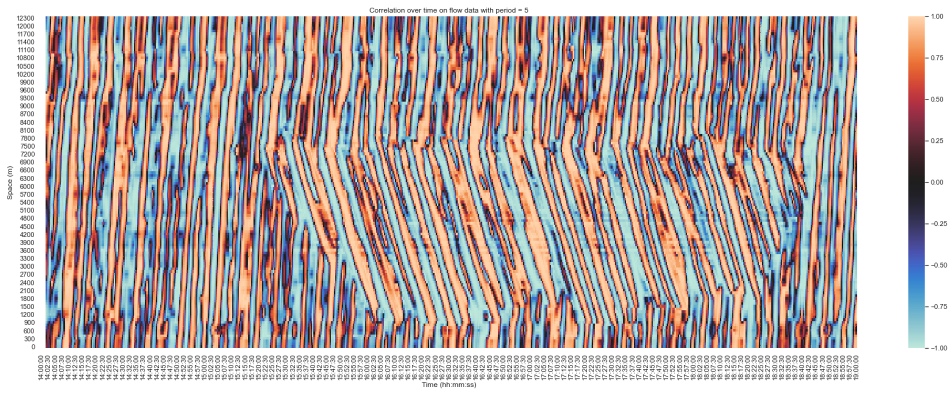


Figure 4.9: Heatmap showing Pearson correlation coefficients of flow over time (window size = 5) for A20 on 20th of November

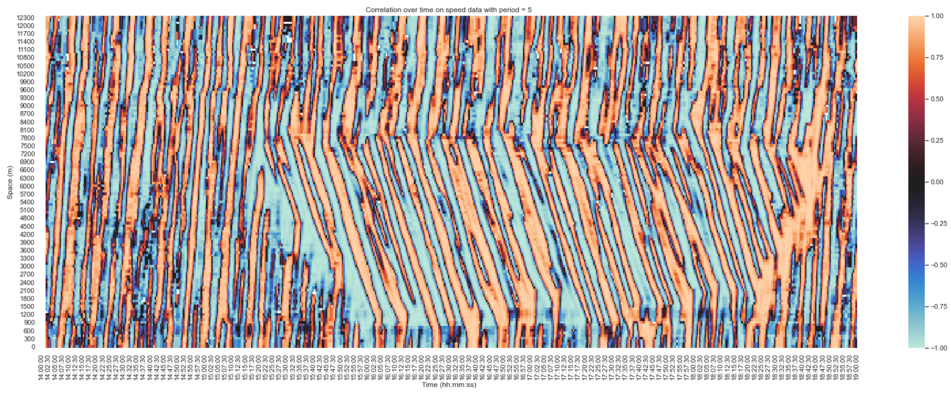


Figure 4.10: Heatmap showing Pearson correlation coefficients of speed over time (window size = 5) for A20 on 20th of November

follow patterns which are related to traffic conditions. Instead, flows and speeds seem to be fluctuating regularly, where values within the waves propagate through space and time with free-flow speed (in uncongested conditions) or shockwave speed (in congested conditions). The fluctuating patterns over space and time look quite similar and seem to be oscillating at the same frequency. This frequency might tell something about the minimum timestep size necessary to correctly capture the fluctuations present in flow and speed. One can detect the minimum sampling frequency for detecting the fluctuations by doing a Fourier transform on the data, from which the results can be seen in Figure 4.11, and finding the peak which has the highest frequency. This has been done on flow data (due to the absence of NaN values) for uncongested situations, as the smallest frequency has been detected here. The periodicities of the found frequencies were 2.8 min (using 5 consecutive points for the correlation calculation) and 3.9 min (using 10 consecutive points for the correlation calculation). Rounding down, one can conclude that the lowest timestep size which still captures these fluctuations is 2.5 min.

4.1.4. Data Quality Assessment

When looking at the flow and speed data, a mismatch was observed between the data itself and what could be expected from the road geometry. There are several reasons that could cause these two phenomena, which mainly are a mix of erroneous data and shortcomings of the implementation

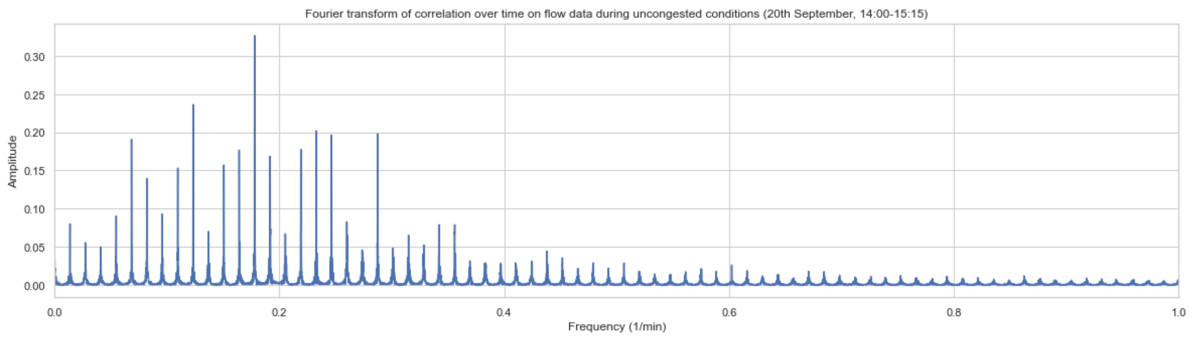


Figure 4.11: Results of Fourier transform of correlation over time on flow data during uncongested conditions (20th September, 14:00-15:15)

of the ASM. The dip in flow data around the service station could be due to two malfunctioning detectors as there are detectors situated at 3800 and 4200 m, both positions within the dip. Since flow drops with 2/3, it could be that traffic on one of the three lanes was not captured correctly. As downstream of this dip, there is a higher detector density than upstream of this dip, the upstream decrease has a much higher slope compared to the downstream increase in flow.

However, the shift in data at the lane merge is a much stranger phenomenon. The previously mentioned dip was exactly at the position of two (faulty) detectors, so it is unclear where this shift came from. When analyzing images at the detection loop positions, one can see that for some detectors around the lane merge only one detector is present for 2 or 3 lanes only which may cause erroneous data. Besides, there is a higher detector density before the lane merge compared to afterwards, which might cause a delay in the increase in flow data after the ASM has been applied. Therefore, several effects might result in the spatial mismatch in this data.

4.2. Validation of Results (A4 Zweth to exit 12 Den Haag Zuid)

The section of A4 will be quickly assessed to see if the results found on A20 can be validated using a different dataset. The A4 between Rotterdam and The Hague is quite similar to A20, as it also lies between to cities (Rotterdam and Delft), crosses a rural area with no on- and off-ramps and also has a lane merge. However, the whole section is a bit shorter with less on- and off-ramps. The schematic representation can be seen in Figure 4.12 and an aerial view in Figure 4.13.

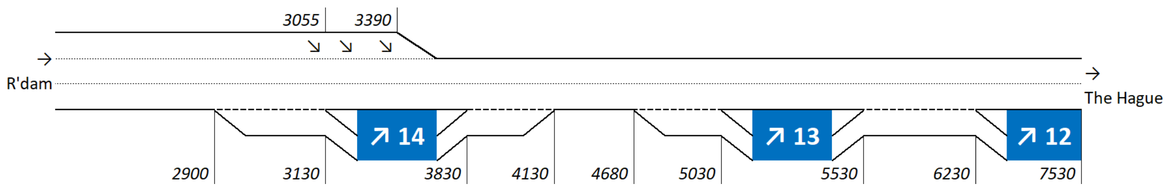


Figure 4.12: Schematic overview of A4 (with distances in m)

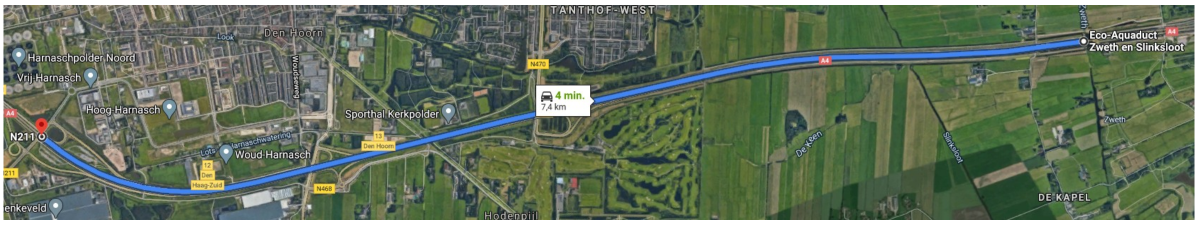


Figure 4.13: Aerial view of A4 (driving direction: right to left)

When looking at the mean data from the A4, as can be seen in Figure 4.14, the flow data shows even more inconsistencies compared to A20. There is a high peak in flow at 2500 m, although there are no discontinuities at this point. Besides, flows seem to decrease after on-ramps and increase after off-ramps, which is not in-line with expectations. However, the increase in flow around the lane merge is as expected, as the flow after the lane merge is 1.5 times the flow before and the peak in flow happens shortly after the lane merge. However, the steady decrease in flow after the lane merge and on-ramp cannot be explained by looking at the road. During congested conditions, two bottlenecks (one at the lane merge and one further downstream of the road section) can be identified. The same can be see in the heatmaps of flow (Figure 4.15) and speed (Figure 4.16) on

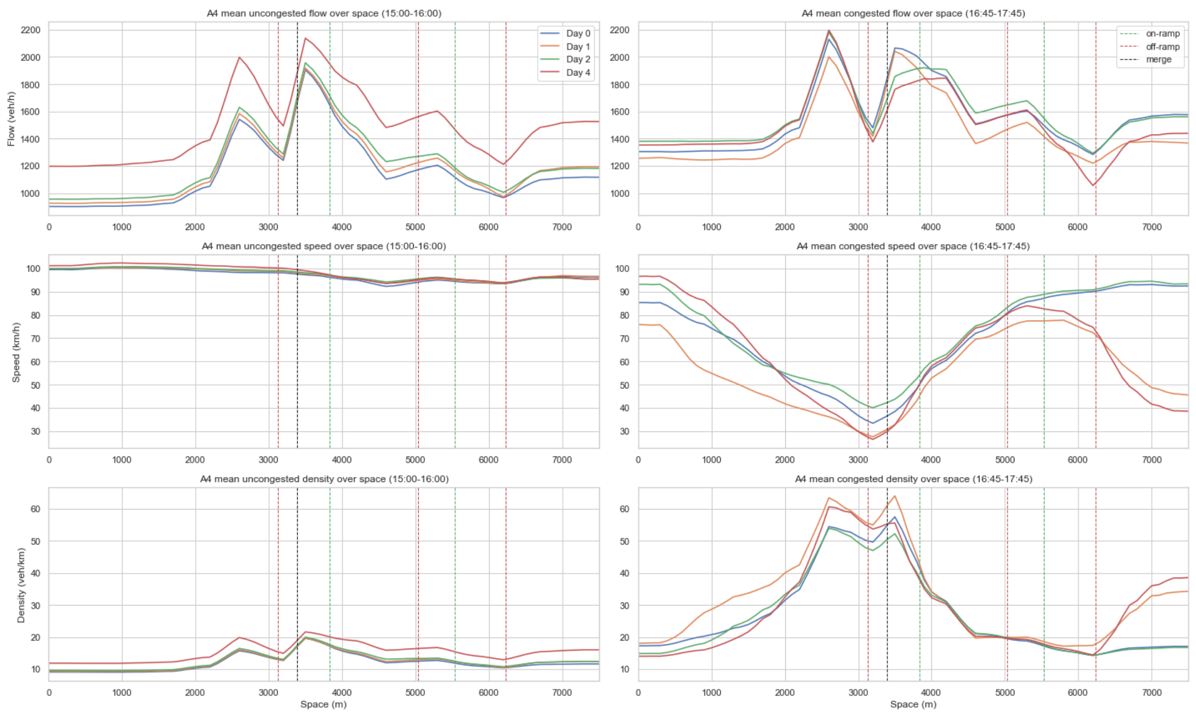


Figure 4.14: Flow, speed and density means over space for A4 in congested and uncongested conditions

the 20th of November. The flow patterns also show the two peaks identified in Figure 4.14. Three bottlenecks can be detected: one at the lane merge (3400 m), one at an off-ramp (which could be due to spillback) and one bottleneck downstream from the analyzed section.

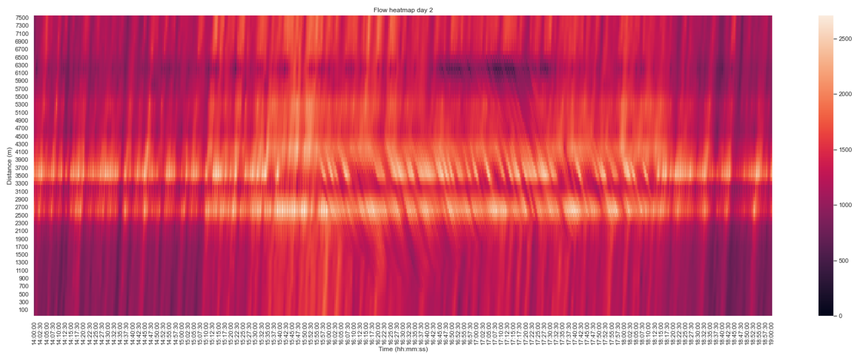


Figure 4.15: Heatmap showing flow over space and time for A4 on 20th of November

When looking in the fundamental diagrams in Figure 4.17, one can see that the fundamental diagram has a strange, square-like shape with both high flows and densities, which especially shows in the three-lane section. It is quite strange that the highest flows are registered both no the three-lane and two-lane section, as the two-lane section would likely reduce the capacity on the three-lane section. This is most likely caused by the two peaks in flow around the lane merge, which are hard

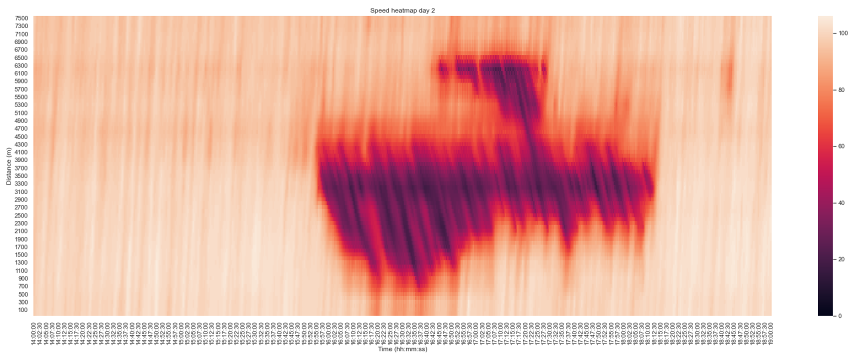


Figure 4.16: Heatmap showing speed over space and time for A4 on 20th of November

to explain. When looking at the correlations, the correlations of flow over space and flow over time

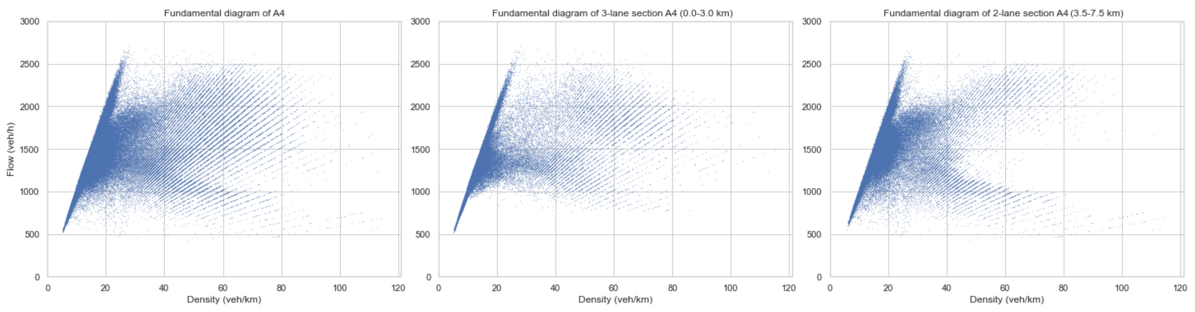


Figure 4.17: Fundamental diagram (u,q) for A4 plus two sections

seem to be the most promising which was also the case at A20. When analyzing correlations of flow over space, as can be seen in Figure 4.18 the following areas can be detected:

- *2900 m:* a flow decrease at approx. 2700 m, approximately coincides with the off-ramp at exit 14.
- *3300 m:* a flow increase at approx. 3100 m, approximately coincides with the lane merge.
- *3800 m:* a flow decrease at approx. 3500 m, is 300 m in front of the on-ramp of exit 14 (although this would lead to a flow decrease)
- *4900 m:* a flow increase at approx. 4700 m, which is near the off-ramp of exit 13 (although this should lead to a flow decrease)
- *5500 m:* a flow decrease at approx. 5300 m, which is shortly before the on-ramp of exit 13, could be explained by the turbulence of the weaving section of the off- and on-ramp.
- *6400 m:* a flow increase at approx. 6200 m, which is shortly after the off-ramp of exit 12, which could be due to a decrease in turbulence since this is the end of a weaving section.

When analyzing the correlation of flow over time, as depicted in Figure 4.19, one can see a similar wave pattern compared to A20. The Fourier analysis, from which the results are given in Figure 4.20, results in a period of 2.25 min, which would result in a period of 2 min, which is slightly lower than A20.

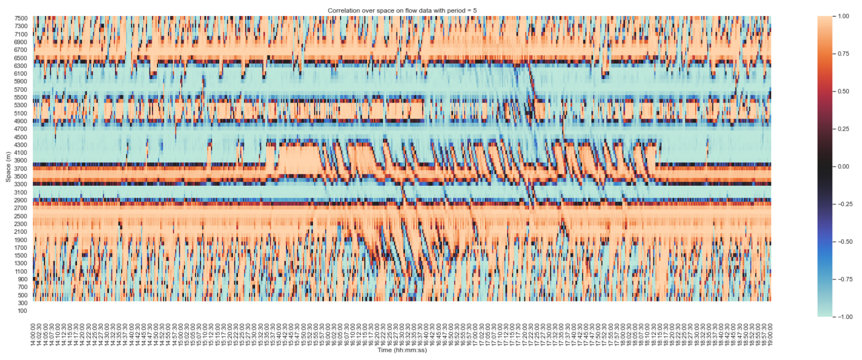


Figure 4.18: Heatmap showing Pearson correlation coefficients of flow over space (window size = 5) for A4 on 20th of November

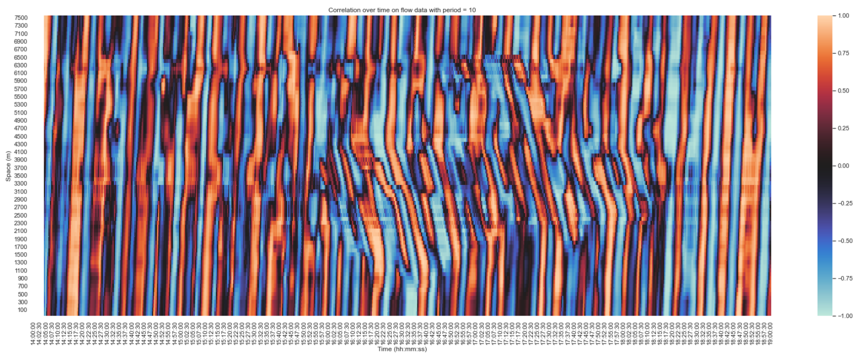


Figure 4.19: Heatmap showing Pearson correlation coefficients of flow over time (window size = 10) for A4 on 20th of November

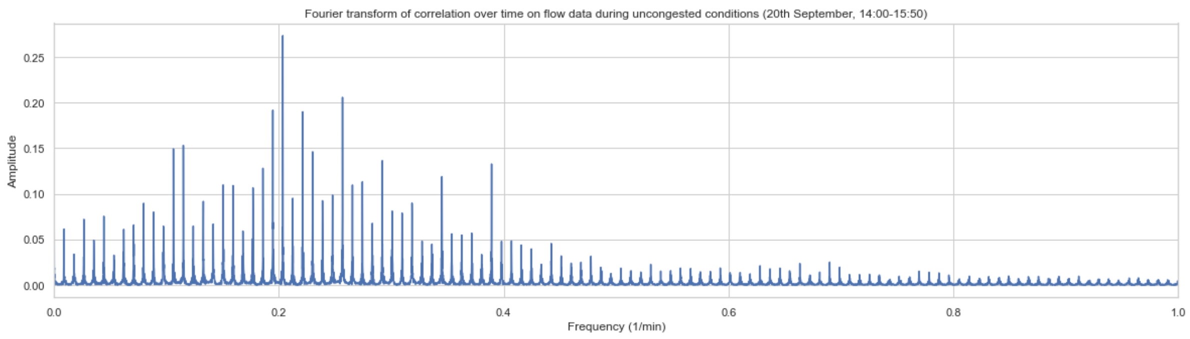


Figure 4.20: Results of Fourier transform of correlation over time on flow data during uncongested conditions (20th September, 14:00-15:50)

4.3. Conclusions

After reporting all the findings, it is important to relate those to the goal of this data analysis and extract conclusions which can help to formulate requirements regarding the neural network. First, one can see that the average values, differences in flow and standard deviations of speed give a good idea of the location of discontinuities, bottleneck locations and congestion spillback. When looking at the heatmaps, one can see clear regions of (approximately) equal flow on the flow heatmap, whereas speed is a good indicator of the extent of congestion. This might hint that flow data provides good information to subdivide the highway spatially, as the borders of these zones of (approximately) equal flow often coincide with discontinuities. When looking at the fundamental diagram, it becomes clear that the capacity of the three-lane road section is limited by the lane merge at the two-lane section. The breakdown in traffic on this three-lane section could not be expected by looking at this section alone. These spatial relationships are therefore important to correctly predict traffic. Since both sections have such different characteristics (maximum capacity, shockwave speed), these might be important input data for the neural network to correctly make distinctions between several types of roads.

After analyzing the data, some interesting correlations were shown in the data. When analyzing data over space, the zones which were seen in the heatmaps could be detected. The borders in the flow data were found to be quite stable and the correlation coefficients within the zones remained relatively similar. Since it is likely that the nodes within the neural network have some spatial meaning, it would be good to divide the nodes according to the results of this correlation analysis. It would be interesting to spatially divide the road according to the results of this correlation analysis. This and other results proves flow is a relevant input variable to make distinction between possible traffic situations. When looking at the change in correlations over time, a minimum timestep of 2.0 min was determined if all information should be saved. Since traffic demand can be quite fluctuating in reality due to a non-uniform inflow of cars, this fluctuating flow pattern over time can be expected. However, if the depth and complexity of the neural network is limited, it would be good to filter out some noise in the data, which can be done by using a moving average or resampling the data over a certain window in time to filter out fluctuations.

However, the data analysis also shows that it is quite cumbersome to extract conclusions on data alone. Data quality can be quite fluctuating, as both the A20 but especially the A4 dataset had some flow data which are hard to explain based on traffic flow theory, most likely caused by erroneous detectors. Besides, the used ASM can also result in unexpected results and artefacts in the data, as it introduces a bias in the data due to its smoothing based on the expected data propagation (with free-flow or shockwave speed). Besides, as the flow data needs a quite large moving average period to smooth fluctuations, one can conclude that data over time is quite noisy when compared to data over space although smoothing has been performed using ASM. As flow data is based on 30-second detector loop countings, only one car count difference in this half a minute can lead to a difference in flow of 120 veh/h. Although much of this noise has been filtered out by the ASM filter, or can be explained by random car arrivals, even the best filter cannot fix bad data quality. It is questionable whether the neural network should be able to capture this wave pattern or this might result in bad prediction quality.

5 Methodology

In the previous chapters, both theoretical and the practical sides of short term traffic flow speeds were discussed. In chapter 2, background information is given on microscopic traffic flow theory on highways whereas in chapter 3, neural networks were introduced as well as their application for traffic forecasting. In chapter 4, actual flow and speed data were analysed to get insight into actual congestion patterns. Based on the findings of these chapters, a research methodology was created. Besides the methodology itself, this chapter also contains hypotheses which were used to assess the accuracy of traffic forecasting neural networks.

This chapter is structured as follows, which is also depicted in Figure 5.1. First, the model design choices and hypotheses on the model workings are discussed, giving some insight in what the model should predict and how it ought to function. Secondly, the data preparation of the datasets obtained by the experimental setup (chapter 6) will be discussed, to make them suitable as neural network inputs. Thirdly, the neural network designs will be introduced, including the methods used to incorporate flow and speed data, the model structure and hyperparameters used. Afterwards, both the model training procedure will be introduced, as well as the sensitivity analysis performed on the best performing model structures. Lastly, the metrics used for the model assessment will be discussed briefly, also introducing the baseline model.

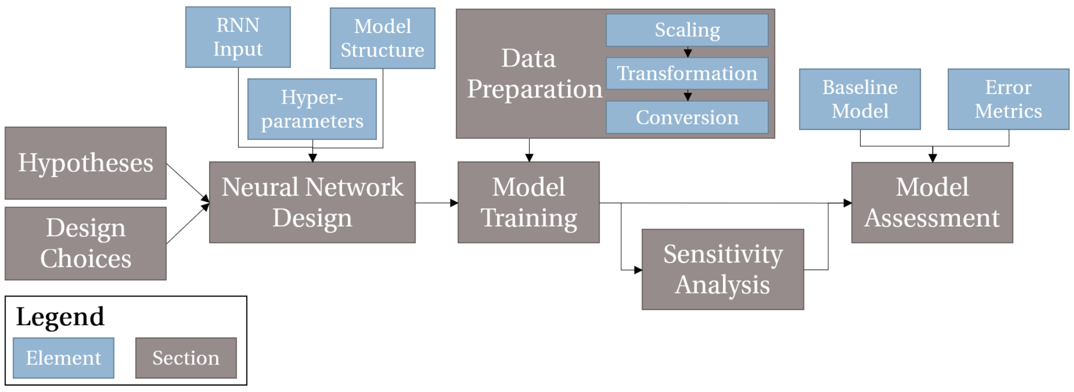


Figure 5.1: Structure of the methodology chapter.

5.1. Design Choices and Hypotheses

In the previous chapters, an overview was given on both traffic flow theory as well as how a neural network functions. To translate these into a neural network design, both **design choices** regarding the neural network and **hypotheses** are formulated. The design choices mostly reflect choices made during the design process, whereas the model hypotheses focus on what the model is expected to be able to do. Therefore, both will be treated separately.

5.1.1. Design Choices

Before comparing model structures, it is important to formulate some general model design choices, which will be used as a starting point for the neural network design. Although the hypotheses will be used to assess how the neural network performs in different scenarios, the design choices prescribe on a more fundamental level what the model should do and focuses on the input and output data, as well as the model type used.

Design choice 1: *the neural network should be able to handle multivariate spatiotemporal data.*

Based on chapter 2 and chapter 4, we can conclude both flow and speed are useful in traffic forecasting. Since speed was found to be most relevant when assessing congestion patterns, the output should consist of at least the speeds. For the input variables, both speed and flow can be relevant, therefore, either only speed or flow and speed were used. Besides, in chapter 3, the most advanced traffic prediction models used spatiotemporal data within the prediction process, meaning that they use data both over space and over time as input. Therefore, all models should be able to handle multivariate (flow and speed) spatiotemporal data..

Design choice 2: *the neural network should consist of GRU units within an RNN structure.*

From chapter 3 could be concluded that RNNs are very suitable for time series analysis and that the GRU unit lead to an even higher model accuracy, while having less parameters compared to an LSTM. Therefore, the GRU cell will be the main building block of the model. A sequence-to-sequence a one-shot model have been introduced in section 3.3, which both have advantages and disadvantages. Therefore, both variants have been implemented. The optimal hyperparameter combination will be determined during training.

Design choice 3: *the neural network should be able to predict at least traffic speeds over the same spatial dimension as present in the input data.*

With respect to the spatial dimensions, we could conclude from section 3.4 that a common approach is that the spatial dimensions of the input and output are equal, which means that predictions are made for all road segments of which input is retrieved. Therefore, this will also be the case for this neural network.

Design choice 4: *the temporal dimensions will be as follows: time interval = 2 min, input sequence length = 15, prediction horizon = 10.*

Although the spatial output dimension will be equal to the input, the temporal output dimension should be considered carefully. The temporal dimension depends on both the input sequence length, the prediction horizon and the time interval, as determined in section 3.4. For this research, the time interval was chosen at 2 minutes, the input sequence length at 15 steps (30 minutes) and the prediction horizon at 10 steps (20 minutes). These values were also used by Li *et al.* [9] and fit the requirements for traffic predictions to be useful (e.g. [15, 42]). Since both the input sequence length and prediction horizon are important factors for the model performance, they will be evaluated using a sensitivity analysis.

5.1.2. Hypotheses

Since the model design choices are set, hypotheses can be formulated concerning what is expected from the neural network. Accepting or rejecting these hypotheses results in more insight in the functioning and limitations of the model. The hypotheses aim at providing insights into three categories. The first category of hypotheses, regarding *traffic flow theory*, are concerned with what traffic patterns the model can actually learn. By reflecting the resulting predictions on what traffic patterns can theoretically be predicted, insights regarding the expected model accuracy can be tested. The second category of hypotheses, concerning the *neural network architecture*, are used to test some assumptions on how the neural network architecture influences the resulting predictions and prediction accuracy. Lastly, the third category focuses on the input and output *data*, since correctly choosing these might be the most important element for the model to correctly predict traffic.

Hypotheses Regarding Traffic Flow Theory

Hypothesis 1: *the model should be able to make traffic predictions while implicitly using input-output relations based on the shockwave theory which can be used to calculate the shockwave speed between different traffic phases, as explained in chapter 2.*

Based on traffic flow theory, it was concluded that predicting the point in time at which traffic breaks down is difficult. However, using shock wave theory, it is possible to predict how both the upstream and downstream congestion head will propagate over the road, given traffic flow does not change abruptly. Therefore, the model should at least be able to correctly predict how the congestion will propagate over the network once the input data contains observations where traffic broke down near the bottleneck. However, predicting this breakdown itself will be challenging.

Hypothesis 2: *it is expected that the model will have a higher accuracy when the congestion patterns consist of synchronised patterns with a single jam head, as for example the moving synchronised pattern, compared to general patterns containing many stop-and-go waves.*

As has been explained in chapter 2, predicting traffic congestion consists of two components, namely predicting congestion emergence and propagation, and there is far more theory on predicting the latter compared to the former. Although bottlenecks are relatively easy to locate, the time at which traffic breaks down depends on many (unobservable) variables. As stop-and-go waves are very narrow space-wise, often occur due to local instabilities and propagate relatively fast (-16 to -20 km/h), it is expected that it is relatively hard to predict their occurrence. However, synchronised patterns, as the LSP, take place over a longer period over time and have a lower shockwave speed, making it easier to correctly predict the congestion propagation. Therefore, datasets containing these synchronised patterns will likely lead to better prediction results.

Hypothesis 3: *it is assumed that there will be large differences in accuracy over space, depending both on the congestion patterns as well as the spatial road characteristics.*

When predicting congestion patterns, one can make two assumptions about the error rate over space. As in the second hypothesis was stated that it is hard to predict the time at which traffic breaks down, the error is likely to be higher near the bottleneck locations compared to locations further away. Furthermore, the prediction is likely to be worse on the edges of the considered highway, since there is less spatial information available on the upstream or downstream traffic conditions to base its prediction on.

Hypotheses Regarding the Neural Network Architecture

Hypothesis 4: *the sequence-to-sequence GRU RNN is expected to perform better than the one-shot model for traffic speed prediction.*

In section 3.3, the sequence-to-sequence model was introduced, which has autoregressive properties. It is expected that this autoregressive prediction performs better than a one-shot prediction, since it only has to predict one timestep ahead and by unrollling this multiple times, can predict an arbitrary number of steps ahead. It is assumed that the neural network has to implicitly estimate a shockwave speed between the free-flow and congestion phase as it has to predict how the upstream and downstream congestion heads propagate through the network. If the neural network has to do this in one shot, the model has to instantaneously predict this shockwave speed 10 steps ahead although the input data were aggregated much more finely, which would lead to a more complex function which would be fitted to correctly fit this input-output relation. However when a sequence-to-sequence model is used, the model is trained to correctly predict only one step ahead and can unroll this function numerous times, using previous predictions as new input. By breaking down the prediction process in smaller steps, each step using the same function, it is likely the accuracy of the predictions will increase.

Hypothesis 5: *the loss function will influence the prediction accuracy, as sharp fluctuations in speed between the traffic phases differ from a regular regression problem.*

As could be seen in chapter 2 speeds on highways can fluctuate quite promptly when a traffic phase change happens. Polson and Sokolov [50] already noticed this and mentioned that this has to be considered when designing a neural network structure predicting traffic speeds. Incorrectly predicting a different traffic phase can quickly lead to high errors, as the difference between free-flow and congestion speed is high. On the other hand, traffic flow theory suggests some traffic breakdowns can be difficult to predict as the causes of this breakdown are unobservable. One can think here of sudden fluctuations in traffic demand or an accident caused by a driver. As no correct input information is present, it can be almost impossible to reduce these errors.

On the other hand, different loss functions lead to different strategies in reducing the error: whereas the MSE focuses on decreasing errors with a high magnitude, the MAE focuses on decreasing the overall error. As the loss function is the sole metric one minimises when fitting a neural network, experts think it has a large impact on the model accuracy. Therefore, it is important to assess how different loss functions cope with these specific challenges when training a neural network.

Hypotheses Regarding Input & Output Data

Hypothesis 6: *having both flow and speed data as model input will improve the accuracy of the speed predictions compared to only using speed.*

Based on the results of chapter 4, it was concluded that both flow and speed hold correlations and information on the traffic state. Besides, in chapter 2, flow was found to be an important parameter both for calculating the shockwave speed as well as for determining and classifying the emerging congestion pattern. Since it would theoretically make sense to include both flow and speed data as model inputs, it will be likely that the model accuracy will increase too.

Besides making sense theoretically, including both flow and speed data could lead to models which are potentially more robust and generalisable. Since both flow and speed are included, the model will be able to distinct a wider variety of traffic situations compared to using only a single one. This will lead to a higher accuracy more robustness since the model can distinct different input-output combinations better while doing so by including parameters supported by traffic flow theory.

Hypothesis 7: *by combining the flow and speed data per road segment and per time interval, the model accuracy will increase.*

Including both flow and speed data, as proposed in hypothesis 6, results in a multivariate GRU RNN structure. From section 3.4, it became clear that designing a multivariate RNN structure requires design choices to be made on two levels: on the one hand, the high-level neural network design containing the general structure of the RNN layers, on the other hand the input and output layer design with respect to both flow and speed parameters. By processing flow and speed data inputs in a smart way, using traffic flow theory, it will be easier for the model to train on the input data and accuracy will therefore increase.

5.2. Data Preparation

In order to train the neural network, flow and speed data was used from several roads simulated with the microscopic traffic model AIMSUN, which is elaborated in chapter 6. However, this general dataset must be converted to tensors of both input data and the corresponding output labels, in order to train the model.

Generating the dataset from this table can be divided in a couple of consecutive steps which will be discussed in this chapter, namely:

1. transposing the data to a table per day
2. converting these tables to tensors which meet the model design choices to be used as input data

First, the data had to be transposed into a tabular format which would make it easy to both generate the tensors for model training as well as to assess the data for an analysis of the model outputs. In order to do this, a four-dimensional table structure was used, with as dimensions $[d, x, t, p]$ where d = the simulation replication (or day), x = the detector location, t = the time point and p = the detector traffic variables considered (being either speed or flow/speed). Saving the data in this structure has several advantages. By aggregating the data per simulation replication, it is easier to visualise the data of one day and to prevent an input data sequence accidentally containing data belonging to two different days. Secondly, by storing the detector parameters in a separate dimension, this would allow for more flexibility when assessing different input layer structures used for including multivariate data.

Secondly, the data should be converted to tensors which can be used as input for the TensorFlow model. The first step that has to be done is data scaling as unscaled data can lead to bad model results due to two reasons. First, if values can have a very large range, such as speed data (ranging from 0 to up to 2800 veh/h/lane), this can lead to exploding gradients or a failing learning process. Secondly, if there is a large imbalance in the values of input variables, the parameters with larger values can be over represented in the model training process, making parameters with low values less significant. This can be a problem since traffic flows can have much higher values than traffic speeds.

The data was scaled using data normalisation, which rescales the data to a range between $[0,1]$ without changing the distribution of the data. This has been done using the following equation:

$$y = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5.1}$$

To make the data normalisation independent of the specific road characteristics, $x_{min} = 0$ and $x_{max} = 2800$ veh/h/lane or $x_{max} = 120$ km/h, as this is equal to the maximum speed limit (and we are not interested in predicting higher speeds) and the maximum flows generally observed on highways.

After scaling, the data should be converted to a datatype which can be used by TensorFlow. There are two datatypes which can be used as input, namely a NumPy array or a (TensorFlow) Dataset. Using a Dataset has some advantages over a NumPy array, as one can use a generator to generate sliding window datasets of arbitrary length from the original dataset, which is easy if many prediction horizons are assessed and it reduces memory usage as it does not have to store all input sequences in memory. However, this increases the computational time of the model due to the computations necessary to continuously generate a new data entry. Therefore, it was chosen to generate each dataset individually and store them as NumPy arrays, since this decreases the computational time while keeping memory load manageable. The input and output sequences were generated from the dataset using a sliding window approach, generating overlapping sequences for each point in time, according to the design choices of the model structure. As mentioned before, each input-output data combination only contained data belonging to one simulation replication.

The data was split according to a training, test and validation split of 70%, 20% and 10%. The validation dataset was used for the early stopping mechanism and hyperparameter optimisation, whereas the test results were used to assess the performance of the different model structures. Since each dataset consists of a fixed number of traffic demand scenarios which were each simulated for several iterations, the dataset was **balanced** to make sure that the scenarios were distributed equally over the split. Besides, the dataset was also **shuffled** to make sure that the scenarios were mixed within each split in a random order. However, no shuffling took place within each day, to retain the temporal sequence structure of each day.

Finally, this resulted in the creation of six datasets: three input datasets with dimension n, x, s_{in} and three output label datasets with dimension n, y, s_{out} with n = the number of data points, x = the input layer dimensionality, y = the output dimensionality and s = the sequence length of the input s_{in} and output s_{out} data. Within the remaining of this chapter

5.3. Neural Network Design

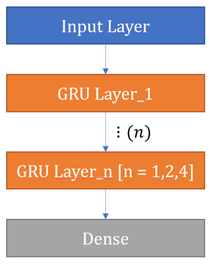
Within the neural network design, there were two different axes along which the model structures could be changed. Firstly, there were two different GRU model structures introduced in section 3.3, namely the one-shot and the sequence-to-sequence model. Secondly, there are several ways in which the two different input data types (only speed or both flow and speed) can be incorporated into a model structure. Therefore, both choices will be elaborated. First, the implementation details of both GRU model structures will be elaborated. Secondly, a framework for including both flow and speed data is given, based on the literature study done in section 3.4. Lastly, the combinations and datasets which were analysed will be discussed.

5.3.1. GRU Model Structure Implementation

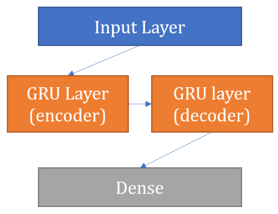
For the output data structure, two different model structures were discussed in section 3.3, namely a one-shot model and a sequence-to-sequence model. Whereas a one-shot model has only one value as output, a sequence-to-sequence model generates an output of arbitrary length, repurposing predictions in an autoregressive way. Both models have been implemented using TensorFlow and Keras, this section explains some of the implementation choices made.

The **one-shot** model structure was implemented in a quite straight-forward way. An input layer, containing the sequences having the predefined input sequence length of all traffic variables, was directly connected to a GRU layer. Depending on the hyperparameters, this GRU layer was connected to other GRU layers where the last GRU layer returned only the latest sequence. Finally, the last GRU layer output was returned to a dense layer, which then returned the final prediction 10-steps ahead. The output consisted of a tensor containing the speed values of each road segment 10 steps ahead. The layer structure can be seen in Figure 5.2a.

The **sequence-to-sequence** model structure, already elaborated in section 3.3 consists of an encoder and a decoder. The input data is transferred to the encoder GRU layer, which then shares its hidden state with the decoder, which then generates an output sequence using this hidden state and previously predicted values. This is depicted in Figure 5.2b with a horizontal connection between the encoder and decoder. A dense layer is then used to interpret each sequence value of the decoder and returns the prediction. Since the predictions of the decoder GRU layer are used for the prediction of the next timestep, the dimensionality of the model input and output has to be the same, as well as ideally the parameters that have to be predicted.



(a) One-shot model structure



(b) Sequence-to-sequence model structure

Figure 5.2: GRU RNN model structures

5.3.2. Multivariate Neural Network Structures

In hypothesis 6 was mentioned that when using multivariate spatiotemporal data, there are several possibilities in which the different traffic variables can be used within the model structure. In this chapter, alternatives for both will be introduced and assessed. First, the general structure of the RNN layers will be discussed, after which the input and output layer structure will be elaborated.

General NN Structure

As mentioned in subsection 3.4.4, two general neural network designs can be thought out, namely (*variant (a)*) one with only one RNN (Figure 3.11a) or (*variant (b)*) one with one RNN per parameter (Figure 3.11b). For the latter, it is important that the results of both RNNs are catenated, so that correlations between the flow and speed sequence can be found. Although both models are feasible, a disadvantage of variant (b) is that correlations between the variables can only be found after the RNN, which means that the correlations found between variables will be merely spatial instead of temporal. Since traffic is a spatiotemporal process, the expectation is that this model will fail to detect some of the spatiotemporal correlations between flow and speed within the input sequences. Therefore, only variant (a) has been implemented.

NN Processing Structure

As there will be only one RNN structure which will use both flow and speed parameters as input, it is important how both parameter types can be combined. For the processing layers of the NN structure, it is important that the RNN can capture correlations (i) between the variables, (ii) over space and (iii) over time. Since having temporal sequences as input for the RNN helps to find temporal correlations, (iii) is covered by the RNN structure and (i) and (ii) should be focused on. The idea of this section is to find a way to smartly combine parameters of two dimensions in the input layer, so the following GRU RNN layers can capture correlations between variables over both the spatial and the parameter dimension efficiently. One can roughly think of two ways to combine both dimensions. Either the dataset is 'flattened', leading to an RNN GRU input consisting of ns units, where n = the number of input parameters ($n = 2$ in case of flow and speed) and s = the number of sections, which covers the spatial dimension. A different option is to combine the flow and speed data per road segment into one value, reducing the RNN GRU input size to s units. This has also been done in the state-space neural network used by Van Lint *et al.* [106] for travel time prediction. Besides, from a traffic theory point of view, one could argue that a flow-speed combination can be reduced to a unique traffic phase, which could contain all necessary information to predict traffic speed patterns.

As can be seen in Figure 5.3, showing an example with flow (q) and speed (v) data for m segments, three of these input parameter bundling techniques have been proposed, where variant (a) and (b) combine flow and speed data into one dimension and variant (c) flattens the input data. *Variant (a)* (Figure 5.3a) is combining the flow and speed using a perceptron, which is by far the simplest method, which can be done with a linear activation function. Although low model complexity is a plus, it is not possible to model nonlinear relationships between variables. Since the relationship between flow and speed is nonlinear, this can pose a problem.

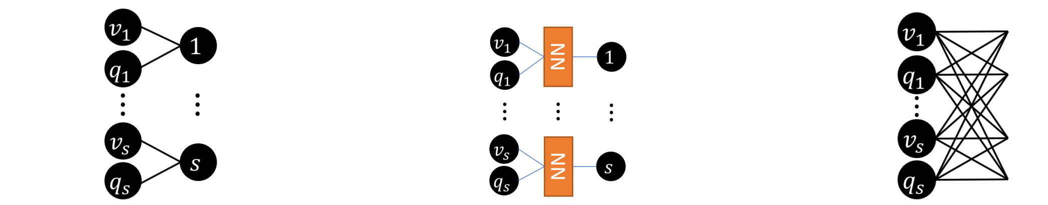
Variant (b) (Figure 5.3b) combines the data using a feed-forward neural network (FFNN), which can be made nonlinear by selecting an appropriate activation function such as ReLU, which is quite popular for nonlinear FFNN [8]. Both the amount of layers and the amount of hidden neurons can be adapted based on the complexity needed. Although this has not been researched, a parameter sharing strategy might reduce model complexity.

Variant (c) (Figure 5.3c) is rather naive, as it fully connects all the input variable to the (first) GRU layer. An advantage of this approach is that the neural network itself can optimize the weights according to the prediction accuracy. However, the accuracy of variant 1 and 2 might be higher based on the a-priori knowledge with which the model structure was made.

A smartly chosen and more complex modelling structure can improve the model accuracy in several ways. First, the number of trainable parameters can be reduced by approximately 23% in the example given in Table 5.1, and this example will only grow larger if there will be more road

Table 5.1: Trainable weights per input layer type having flow and speed data as inputs

Model type	Road segments	Hidden layers	Units/layer	Weights	Reduction
a (perceptron)	100	2	200	261,200	23.3%
b (non-linear)	100	2	200	262,600	23.0%
c (fully connected)	100	2	200	340,900	0%



(a) State per segment (perceptron) **(b)** State per segment (nonlinear FFNN) **(c)** Fully connected

Figure 5.3: NN processing structure variants. Each node represents an input variable of the dataset. For (a) and (b), the rightmost node will be connected with the first GRU RNN layer.

segments or traffic variables per road segment. Secondly, by introducing a-priori knowledge in the model neural network structure in the right way, it is likely the model accuracy will either benefit from this or it will not lead to a decrease in accuracy, while still having less parameters to be trained.

However, a custom input layer design can also have several disadvantages over a fully connected structure. First, creating a smaller amount of input neurons for the GRU cell can be especially advantageous if there is only little training data, since the model does not have to learn these model weights. By connecting individual nodes in a smart way, therefore introducing preliminary 'knowledge' in a model, not all possible model weights have to be fit since some will be set at zero. However, if there is abundant training data available, it is possible to use a fully connected layer as there is enough data to fit the larger amount of model weights. This will lead to a more flexible model structure. Secondly, it is debatable whether flow and speed can be combined into one state. One can imagine that the resulting function will be complex and non-linear, as can be deduced from the shape of a (simple) triangular flow-speed fundamental diagram (see Figure 2.1).

5.4. Model Training

After formulating the model design choices and hypotheses, preparing the data and designing the neural network structures, this section will elaborate on the way each model has been trained on the datasets. First, the training procedure will be explained, including the optimisation algorithm used during the learning process. Then, the hyperparameter tuning process will be assessed, including the parameters to be tuned.

5.4.1. Training Procedure

In this chapter, the most important parameter choices for the training procedure will be presented, which are summarised in First, the optimisation algorithm has to be chosen. In subsection 3.1.1, the concept of training has been explained as well as the gradient descent (GD) and stochastic gradient descent (SGD) methods. Despite stochastic gradient descent performing quite well, a few better performing optimisation algorithms have been developed, which are based on these methods but often perform better than these. One of them is Adam [107], which is seen as the standard optimisation algorithm for Deep Learning, which has been proven to work well on various deep learning problems and is also the default choice in TensorFlow. It combines the advantages of two other developed algorithms (AdaGrad and RMSProp) over a classic SGD optimisation, namely

- having a per-parameter learning rate, which improves performance on sparse learning prob-

Table 5.2: Parameter values chosen for model training procedure

Parameter	Value
Optimisation algorithm	Adam
Learning rate	1e-4
Batch size	32
Number of epochs	250
Early stopping	
Patience	6
Min_delta	1e-9

Table 5.3: Hyperparameters used while training model variants

Hyperparameter	One-shot	Sequence-to-sequence
Layers	[1,2,4]	1
Units/layer	[1,2,4]*segments	[1,2,4]*segments
Output activation	[linear]	[sigmoid]
Loss	[MSE, MAE]	[MSE, MAE]

lems and

- adapting this learning rate based on the average of the previous gradient values, making its performance good on noisy training data.

Besides these two advantages, it is also computationally efficient and requires little hyperparameter tuning to get good results. Within Adam, several hyperparameters can be tuned, but since the standard hyperparameter settings also proposed by Kingma and Ba [107], which was proven to give good results on a wide set of learning problems. The learning rate has been changed to $1e-4$ (from $1e-3$, which has proven to converge within the number of epochs. The batch size was set to 32, as suggested by Masters and Luschi [108], Bengio [109].

For other learning related parameters, the following values were chosen. The maximum **number of epochs** was chosen at 250. Furthermore, an **early stopping** mechanism was implemented, to avoid overfitting (explained in subsection 3.1.2) as it stops training the model when the generalisation error increases. Therefore, the early stopping mechanism would work based on validation loss, as this represents the generalisation error better. The following other parameters were chosen:

- patience = 6, needing at least 6 iterations with a higher validation loss than the minimum value for the training to stop to ensure training to continue when a temporary increase in loss is observed,
- min_delta = $1e-9$, meaning that $L_{val}^{curr} - L_{val}^{min} < 1e-9$ for the early stopping mechanism to be activated

Together with the early stopping mechanism, this enabled good training of the model within the maximum of 250 epochs.

5.4.2. Hyperparameter Tuning

In order to find the best performing neural network combination, several hyperparameters can be combined. After defining the different hyperparameters, one can train the model with all the different combinations using grid search. All hyperparameters can be found in Table 5.3

Two different **loss functions** were used, namely both an MSE and MAE loss, which both have been explained generally in subsection 3.1.1. For the number of layers, using 1, 2 and 4 layers have been tested, whereas the number of nodes per layer depends on the number of segments (and therefore the input dimensionality). In subsection 3.3.1, different combinations of layers and units per layer were explained, stating it mostly trial and error based. Deeper models are able to capture more complex, nonlinear relations between input and output variables [6]. Therefore, it is expected that in more complex traffic situations or longer roads with more complex spatiotemporal correlations, models with a higher number of layers or more units per layer are expected to lead to lower errors. However, deeper and more complex models are also more difficult to train, which potentially leads to a higher error [30]. Therefore, it is likely that per situation an optimum in model depth can be found. To prevent overfitting, the maximum number of model parameters was chosen to be lower than the number of entries in the dataset.

For the output activation function of the dense layer, two activation functions were tested, namely a linear activation function (which is equal to having no activation function) and a sigmoid activation function, as presented in section 3.1. For regression problems, most of the time a linear output activation function is chosen, whereas for a classification problem, sigmoidal activation functions are preferred (such as a sigmoid or tanh function). However, sigmoidal activation functions are suggested to lead to more stable learning, as the derivative of these functions decreases for extreme values. For both the one-shot and the sequence-to-sequence model, originally, both output activations were tested. As was concluded that the output activation functions mentioned in Table 5.3 lead to the best results, which can be seen in Appendix A, only these were used in order to decrease the number of simulations necessary.

For each GRU cell, the standard Tensorflow/Keras hyperparameters were selected, having a tanh activation function and a sigmoid recurrent activation function, since this enabled using the CuDNN kernel and faster training via a GPU.

5.5. Sensitivity Analysis Temporal Horizon

After finding the best model configuration using grid search, a sensitivity analysis was used to test the effect of changing parameters on the best performing models. Specifically, this was done to assess the effect of the input parameter length and the prediction horizon on the accuracy of the model. Since these were chosen based on the model configuration of Li *et al.* [9] and were excluded from the grid search, in an effort to limit the number of model runs, the effect of these parameters on the accuracy was not taken into account.

It is expected that the **input sequence** length has limited effect on the accuracy. It is not expected that inputting values observed longer than 30 minutes ago will contain relevant information for the prediction task. Besides, as GRUs are optimised on longer sequences which are often present when doing natural language processing, it will be able to extract the most relevant sequence values even from longer sequences. Therefore, it is likely that the accuracy will gradually increase with a higher input sequence length, but eventually will stabilise. Therefore, it is expected that higher input sequence lengths will have limited effect on the model accuracy.

All chosen parameters can be found in Table 5.4, on which a grid search has been performed. For both datasets, a smaller and larger input sequence value has been chosen. The largest value was chosen relatively high to see whether the GRU RNN could extract the most relevant information from longer sequences. For the second dataset, only smaller prediction horizon values were tested against the current value of 10 steps, since the model accuracy was relatively poor and it was expected this was due to the combination of a large prediction horizon and complex congestion patterns. Since prediction accuracy was high for the third dataset, both a smaller and larger prediction horizon has been tested.

It is expected that the prediction horizon will impact the prediction accuracy far more, as in

Table 5.4: Input sequence and prediction horizon values chosen for the sensitivity analysis.

Dataset	Input sequence	Prediction horizon
Dataset 2	[10, 15, 25]	[2, 5, 10]
Dataset 3	[10, 15, 25]	[5, 10, 15]

general, higher prediction horizons will lead to more uncertainty in the predictions and therefore a higher error. However, it is expected that this accuracy will also depend on the congestion patterns present in the data. The more complex these patterns will be, the faster model accuracy will decrease when prediction horizons get higher. Therefore, it is important to assess the sensitivity of the prediction horizon on the model accuracy per simulation dataset, since this provides more insight in this sensitivity per traffic pattern type.

5.6. Model Assessment

To assess the results of each model, several methods have been used to assess the results on several levels of detail. Most of the time, the accuracy of neural networks is assessed based on general error metrics which aggregates over all outputs generated by the model. Although this can give a general overview of the accuracy of the model, it is also helpful to look more into detail and aggregate on different levels. In case of a spatiotemporal prediction problem, it would be natural to also assess errors over space and time. Therefore, several methods that have been used will be presented.

First, the baseline model will be introduced, which was used to compare the model results. Secondly, some generic error metrics used to analyse neural networks are presented, as well as an interpretation of them in a traffic context. Thirdly, a method is given which qualitatively assesses the results of a single speed prediction for one simulation iteration. Lastly, new error metrics are proposed which calculate errors over space and/or time aggregated over all simulation iterations of a dataset.

5.6.1. Baseline Model

In order to evaluate the results of the prediction not only with each other, but with respect to not having a prediction model, a 'baseline model' was introduced. For the model to have any relevant predictive power, it should be more accurate than the baseline model. For a baseline model, two model types are popular. A historical average (HA) is used often (e.g. [58, 84, 110]), which simply models the prediction to be the average of all values of all days at that timepoint. Secondly, the random walk (RW) is popular [51, 54], where one assumes flow or speed values increase or decrease randomly over time, which can lead to quite poor estimations. A third option would be to predict no change, so the predicted value would be equal to the last known value, or $x_{t+m} = x_t$ with t = the last timestep and m = the shift in time of the prediction. Although the author has not found any papers using this method as a baseline model, it is used by Google Maps in its routing algorithm and is used by Rijkswaterstaat to give drivers insights in the travel times on routes between the same origin-destination pair. When choosing a short prediction horizon or when traffic conditions are quite stable, one can imagine this method works quite well. Therefore, 'no change' is chosen as baseline model it is expected to perform better than HA, considering the relatively short prediction horizon taken, and will most definitely perform better than RW.

5.6.2. General Error Metrics

First, we will look at the general error metrics used to assess the model and give their interpretation in a traffic-oriented setting. These general error metrics are most common when assessing the accuracy of neural networks, and result in one single number which aggregates the error of all outputted values. The following index notation is being used: $y^k_{s,t}$ where y = the ground truth value, s = the spatial segment index (with S being all segment values), t = the temporal index (with T being all time steps) and k = the simulation replication index, or 'day' (with K being all replications). For predicted values, $\hat{y}$ is used.

As the sequence-to-sequence model outputs all input data and returns the whole predicted sequence, the error metrics are also calculated differently between the two model types. To accurately compare the model results with the one-shot model, all errors are only calculated on the last *speed* values of each sequence.

The mean absolute error (MAE), which has been introduced in subsection 3.1.1, simply calculates the absolute difference between prediction and ground truth, and averages this over all predictions made. Using the index notation mentioned before, it is calculated using the following equation, where n = the total amount of values:

$$MAE = \frac{1}{n} \sum_{s\in S, t\in T, k\in K} \left| y_{s,t}^{k} - \hat{y}_{s,t}^{k} \right| \tag{5.2}$$

When interpreting the MAE, the magnitude of the error does not affect the weighting of it. Therefore, a small error occurring in a large amount of predictions can have the same effect on the error term as a large error for a small amount of predictions. This can make this error less relevant, as one is more interested in minimising the (larger) errors occurring at the borders between traffic phases and less in small speed fluctuations within a traffic phase.

The root mean squared error (RMSE) is similar to the MSE loss function introduced in subsection 3.1.1, but the square root is taken, resulting in the following equation:

$$RMSE = \sqrt{\frac{1}{n} \sum_{s\in S, t\in T, k\in K} \left(y_{s,t}^{k} - \hat{y}_{s,t}^{k} \right)^2} \tag{5.3}$$

By squaring the result, the unit of of the RMSE is the same as the output, making its value easier to interpret compared to the MSE. As large differences are penalised higher compared to small differences, it is especially suitable if one is more interested in larger prediction errors over smaller errors. Within traffic speed prediction, the RMSE can be interpreted as follows, when compared to the MAE. When two models have comparable MAE values but a high RMSE value, the error is focused on a *small* amount of *large* mispredictions, for example when the traffic phase is mispredicted. The model with a low RMSE will have a relatively *large* amount of *smaller* errors, for example by incorrectly predicting the speeds within a phase.

The mean absolute percentage error (MAPE) calculates the difference between prediction and ground truth as a ratio of the ground truth, resulting in the following equation:

$$MAPE = \frac{1}{n} \sum_{s\in S, t\in T, k\in K} \left| \frac{y_{s,t}^{k} - \hat{y}_{s,t}^{k}}{y_{s,t}^{k}} \right| \tag{5.4}$$

By dividing the error by the ground truth value, the MAPE will especially penalise errors when the congested state is not detected, as ground truth speed values during congestion are lower than when in free-flow. Let's assume v_{free} = 120 km/h and v_{cong} = 30 km/h. Mispredicting a state will lead to an absolute difference of $\left| v_{free} - v_{cong} \right|$ = 90 km/h. However, when the ground truth value is v_{cong}, $MAPE$ = 300% whereas when the ground truth is v_{free}, $MAPE$ = 33%. Therefore, the MAPE is only useful for detecting mispredictions of the congested phase. This makes it far less relevant than the MAE and RMSE and its resulting values should be interpreted with caution.

5.6.3. Qualitative Heatmap Assessment

As a first step, the predictions were analysed by showing the prediction errors for a specific simulation iteration by creating a speed heatmap of the predicted and ground truth speeds and the residuals of the two. An exemplary heatmap including these residuals can be seen in Figure 5.4. The residuals, defined as $\hat{y} - y$, give a clear overview on which spatiotemporal locations are systematically overestimated $(\hat{y} > y)$ or underestimated $(\hat{y} < y)$.

By assessing the complete heatmap of the results over space and time, one can see both spatial as well as temporal mismatches with the ground truth. This can give more qualitative insights in the prediction accuracy. As most congestion patterns in chapter 2 are also analysed using speed heatmaps, analysing the prediction results in the same format will definitely help to draw parallels with traffic flow theory as well. Besides qualitatively comparing the prediction heatmaps with the ground truth heatmaps, the **residuals** are also plotted. As 0 km corresponds to the most downstream segment, the trajectory of a vehicle on this road would move from the lower left (starting at 1600 m) to the upper right (at 0 m).

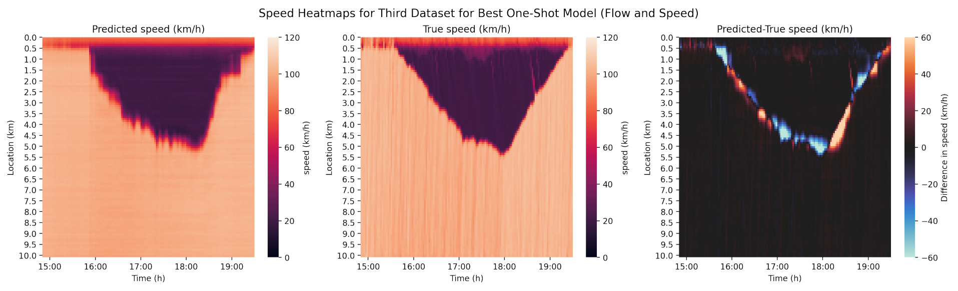


Figure 5.4: Exemplary speed heatmaps and residuals for a simulation iteration of the third dataset.

5.6.4. Spatiotemporal Error Metrics

Although these general error metrics can be useful to generalise and compare the performance of several traffic predicting neural networks, they provide little insight in the causes of the error. When assessing the error of traffic forecasts, one is especially interested in the accuracy when traffic has a higher uncertainty. This can happen at specific locations in time where traffic demand fluctuates(i.e. during peak hours) and/or space where road capacity is lower(i.e. at bottleneck locations). After all, if the higher model accuracy is the result of a better free-flow speed estimation, but prediction accuracy near bottlenecks decreases, the added value of this increase in accuracy is debatable. Therefore, it would be interesting to calculate these errors over space and time.

In this section, several error metrics are proposed which can aid the interpretation of the error. First, the error over space is assessed. Secondly, the error over time will be interpreted. Thirdly, methods are proposed to assess the error over both space and time.

Spatial Errors

By calculating each error metric for each individual detector location separately, one can assess the performance of the neural network on a spatial level. In this way, hypothesis 3 can be answered in a better way and one can assess which road stretches have congestion patterns which have higher predictability (as the errors will be lower) compared to other road stretches. This makes it possible to relate the error to the road layout and potential bottleneck locations. Calculating the error per spatial segment results in the following equations:

$$RMSE_s = \sqrt{\frac{1}{n} \sum_{t \in T, k \in K} \left(y^k_{s,t} - \hat{y}^k_{s,t}\right)^2} \tag{5.5}$$

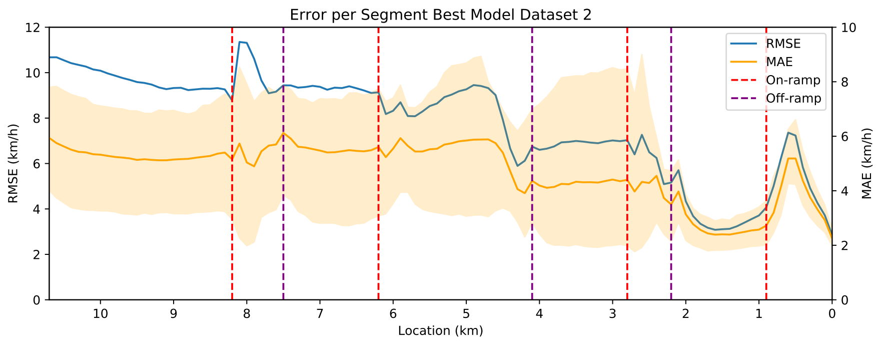
$$MAE_s = \frac{1}{n} \sum_{t \in T, k \in K} \left|y^k_{s,t} - \hat{y}^k_{s,t}\right| \tag{5.6}$$

Only the MAE and RMSE have been included, which both have a different scale to compensate for the difference in magnitude. The MAPE was discarded as it contained the least relevant information of all error metrics and resulted in a decrease of readability of the graph.

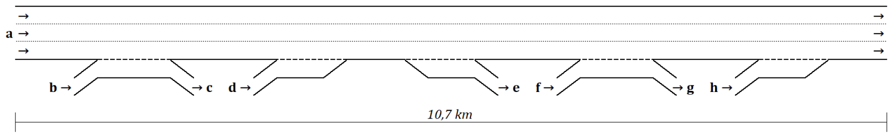
To visualise this error metric, a 2D line plot was chosen, as the line plot emphasises the continuity of the spatial data and enables the visualisation of large detector sets in an intuitive way. The final result can be found in Figure 5.5. The x-axis, containing the segment location, was defined so road traffic drives from the left to the right as it does in the road layout in Figure 5.5b. As this is equal to the reading direction, it could result in a more intuitive interpretation. The most important aspect in the choice of the x-axis is that when interpreting the graph, the results are easy to relate to the road layout. The horizontal dotted lines indicating the exact locations of on- and off-ramps further help relating the results of the plot to the road layout.

For the MAE, a confidence band is included depicting the interquartile range (IQR), showing the range of results between the 25th and 75th percentile, of the MAE which was calculated over different *simulation iterations*. The IQR provides information on the variability of the prediction errors for different simulation scenarios. As the line corresponds to the overall MAE, this shows the *mean* MAE value over all simulation iterations (instead of the median). This confidence interval has only been calculated for the MAE as only for this error metric, the mean of all MAEs calculated per simulation iteration would be equal to the general spatial MAE. This is not the case for the RMSE due to the square root in its equation.

Although both the MAE and RMSE results look quite similar, in Figure 5.5a, some differences can still be observed. Main differences can be observed in the order of magnitude, although the RMSE values deviate from the MAE values (for example at 8 km). In chapter 7 and chapter 8, this difference will be elaborated further.



(a) Error over space.



(b) Road layout.

Figure 5.5: Exemplary plot of MAE/MAPE error metrics assessed over space. The coloured band over the MAE represents the IQR of the MAE values per replication.

Temporal Errors

An error over time can also be depicted in a similar way as has been done for the spatial errors. This enables fluctuating errors to be related to changes in traffic demand and resulting changes in traffic inflow. Calculating the resulting errors per time results in the following equations to be used:

$$RMSE_t = \sqrt{\frac{1}{n} \sum_{s \in S, k \in K} \left(y^k_{s,t} - \hat{y}^k_{s,t}\right)^2} \tag{5.7}$$

$$MAE_t = \frac{1}{n} \sum_{s \in S, k \in K} \left|y^k_{s,t} - \hat{y}^k_{s,t}\right| \tag{5.8}$$

An exemplary plot can be found in Figure 5.6, which is quite similar to interpret compared to the spatial error in Figure 5.5. The main difference is that on the x-axis, time is depicted.

Spatiotemporal Errors

The last metric proposed is similar to the qualitative speed heatmap assessment, but then calculates these errors over all simulation iterations. This also results in a heatmap, but the value of each cell corresponds to the error of a segment at a time which is calculated over all simulation iterations.

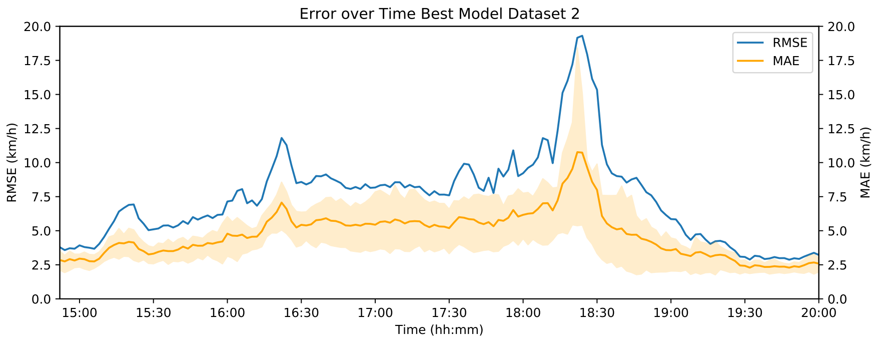


Figure 5.6: Exemplary plot of MAE/MAPE error metrics assessed over time. The coloured band over the MAE represents the IQR of the MAE values per replication.

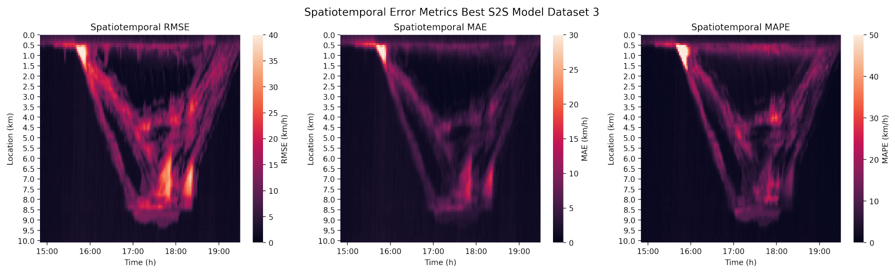
This is done using the following equations:

$$RMSE_{s,t} = \sqrt{\frac{1}{n}\sum_{k\in K}\left(y^k_{s,t} - \hat{y}^k_{s,t}\right)^2} \tag{5.9}$$

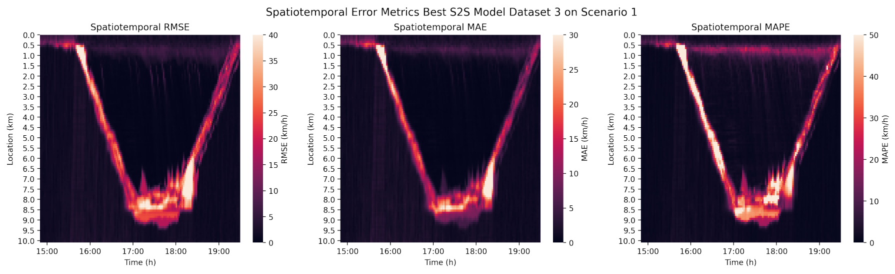
$$MAE_{s,t} = \frac{1}{n}\sum_{k\in K}\left|y^k_{s,t} - \hat{y}^k_{s,t}\right| \tag{5.10}$$

$$MAPE_{s,t} = \frac{1}{n}\sum_{k\in K}\left|\frac{y^k_{s,t} - \hat{y}^k_{s,t}}{y^k_{s,t}}\right| \tag{5.11}$$

An example of the resulting heatmaps can be found in Figure 5.7. Calculating the errors for all iterations also results in a quite noisy heatmap, as can be seen in Figure 5.7a. However, when assessing the errors for only one scenario, where traffic demand is approximately equal, one can see clear similarities when the figure is compared to the residuals in Figure 5.4. However, since the resulting errors are based on multiple iterations, conclusions drawn from this proposed spatiotemporal error metric are better generalisable as they are applicable to multiple simulation iterations.



(a) Spatiotemporal error for all iterations of dataset 3.



(b) Spatiotemporal error for all iterations belonging to the first scenario of dataset 3.

Figure 5.7: Exemplary heatmap of the spatiotemporal error metrics for dataset 2.

6 Experimental Setup

In order to train and evaluate each neural network, traffic data has to be retrieved which in this case has been done using a microscopic traffic flow model. By creating different scenarios and simulating different roads, a wide variety of congestion patterns can be generated. This allows for the model accuracy to be evaluated on a wide variety of traffic situations. The following chapter will elaborate on how this traffic flow and speed data was created, assessing the different conditions under which traffic has been simulated.

This chapter is structured as follows. First, the microscopic simulation software will be introduced, as well as the configuration and parameters which have been used. Secondly, the different datasets, each consisting of several scenarios, will be elaborated. Lastly, the steps used to prepare the simulation data to use them as neural network inputs will be explained.

It should be noted that for the heatmaps in this chapter, the distance of 0 km represents the most downstream segment of the road and distances are ascending in the upstream direction.

6.1. Microscopic Traffic Model Parameters

As has been mentioned in the methodology, a microscopic simulation model was used to generate datasets for training the neural network. Compared to using actual data, this enables for more control over the traffic patterns on which the neural network is trained, since the traffic scenarios can be adapted easily to meet the research requirements. In the context of this thesis, this has two advantages. First, since the model accuracy can be assessed on scenarios having specific traffic patterns, it is easier to relate the prediction accuracy to the type of congestion patterns modelled. Besides, when designing a location transferable neural network, one can exactly design the traffic congestion patterns present in the data. As one can isolate different properties and has better knowledge on the different characteristics of the dataset, it will in the end be easier to trace differences in results back to different properties.

All datasets were generated in AIMSUN NEXT 20, which is a popular microscopic traffic model software. For all datasets, only the evening peak hour and some adjacent hours have been simulated, which would correspond to times between 14:00 and 20:00, instead of simulating full days. Since no regular congestion occurs during off-peak periods and speeds are likely to be equal to free-flow speeds, speed predictions during off-peak hours are not of much added value. Including them would result in a much higher overall accuracy, although it is unlikely a neural network will be used to predict traffic during off-peak hours. Besides, congestion during off-peak hours mostly occurs due to incidents instead of high traffic flow, making it very hard to predict their occurrence and the implications of a certain accident. Besides, it will result in less simulation data needed since large periods with free flow, containing little relevant 'information' for the model, are avoided. Although the time which was assigned to the simulation data is arbitrary, evening peak hour has been chosen, as this was also present in the preliminary data analysis. To reduce the complexity of the dataset generation, each dataset consists of only a couple of scenarios. For each scenario a different traffic

Table 6.1: Important microscopic traffic model parameter values.

Parameter	Value
Warm-up period	15 min
Detector spacing	100 m
Time interval data	2 min
Road type	'Motorway'
Lane width	3.5 m
Aggressiveness	30%
Cooperation	60%

demand was determined, resulting in other congestion patterns. To create a bigger dataset, all scenarios have been simulated for 100 iterations, each scenario having a different random seed. With each chosen random seed in AIMSUN, slight differences in network loading and vehicle characteristics can occur. Therefore, within each scenario, each iteration contains some differences from the other iterations, resulting in a quite diverse dataset.

The design parameters used while building the model can be seen in Table 6.1. The standard virtual loop detectors of AIMSUN were used, which save flow, (TIME/SPACE) mean speed and density at each location, which were aggregated on a 2 minute level. The road type 'motorway' has a maximum speed of 120 km/h, although the mean desired speeds of cars (110 km/h) and trucks (85 km/h) was slightly lower. From the driving parameters, both the aggressiveness and cooperation were changed to create more realistic traffic patterns. The aggressiveness is increased from 0 to 30%, to create more sudden lane changes leading to more traffic breakdown. By reducing the cooperation from 80 to 60%, the resulting car merging process was less smooth, which also resulted in more congestion.

6.2. Dataset Characteristics

As mentioned before, three different highways have been simulated, which lead to three different dataset. Each *dataset* consisted of a couple of *scenarios*, which each have a unique traffic demand and therefore different resulting flow and speed patterns. Of each scenario, 100 *iterations* have been simulated. Each iteration corresponds to what would be a day when using simulation data. The resulting sizes of each dataset can be found in Table 6.2.

In this section, the datasets will be presented in a chronological order. Since the development of each next dataset has been done using insights of the previous dataset(s), the thought process and considerations done while developing these datasets will be explained first. This introduction will be followed-up by a in-depth introduction of each dataset individually.

The first dataset consists of a short three-lane highway (1.5 km) with at the end a lane merge bottleneck, as this is a relatively simple bottleneck which results in the formation of a moving synchronised pattern (MSP) upstream of the bottleneck location. As the road is relatively short and the congestion patterns rather simple, this would make for a good first assessment of the possibilities of a neural network for traffic forecasting. However, from the first results could be concluded that the low accuracy of the model was also caused by the dataset. The short road length resulted in

Table 6.2: Sizes of datasets resulting from simulations.

Dataset	Scenarios	Iterations per scenario	Total days	Road segments	Time per iteration
Dataset 1	2	100	200	16	5.5 hours
Dataset 2	3	100	300	101	6 hours
Dataset 3	4	100	400	108	5.5 hours

little information for the model to base its predictions on. Besides, the simple congestion patterns resulted in a relatively good performance of the baseline model.

The second dataset should therefore consist of more complex congestion patterns. To do this, a much longer three-lane highway (10.7 km) was created, containing several on- and off-ramps. Traffic demand was tuned so that a mix of GPs with stop-and-go waves and MSPs occurred. As a result, the model accuracy improved, resulting in a lower error compared to the baseline model. However, after assessing the results qualitatively, it was observed that the model had difficulties tracking the correct shockwave speeds of the stop-and-go waves in the GP and the congestion tail of the MSP.

This resulted in the creation of the third dataset which, like the first dataset, consisted of a 3-lane highway with a lane merge bottleneck. However, the road length increased to 10 km and the different demand patterns resulted in the formation of one MSP per simulation iteration, each scenario having a different shockwave speed. Therefore, this dataset is very suitable in checking whether a neural network is able to predict the correct shockwave speed while only having to predict traffic breakdown once per day.

6.2.1. First Dataset

As a first dataset, a simple lane merge bottleneck has been simulated. It consists of a 2000 m straight 3-lane section followed by a 300 m 2-lane section, which is shown in Figure 6.1. Detectors are placed every 100 m from 1500 m upstream of the lane merge until the lane merge. Two traffic demand patterns have been created, which simulate traffic patterns on the A20 around the evening peak hours between 14:00 and 19:30. Traffic inflow is set per half hour, and is shown in Figure 6.2 One of them creates heavy congestion whereas the other creates lighter, reoccurring congestion waves. Since Hoogendoorn *et al.* [28] argue that the traffic composition (trucks/cars) clearly has an influence on the breakdown characteristics, the demand pattern contains only cars. From the two demand patterns, 100 'draws' are taken which contain slight differences in model results, which results in 200 'days' of data in total. The first scenario results in very heavy congestion, with high shockwave

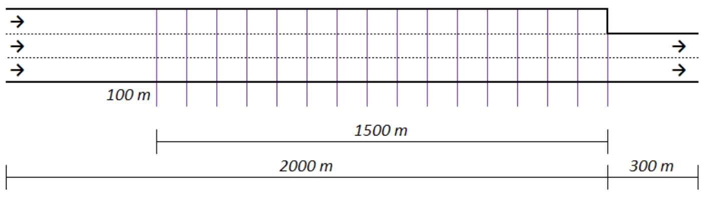


Figure 6.1: Schematic view of simulated road with detectors (in purple).

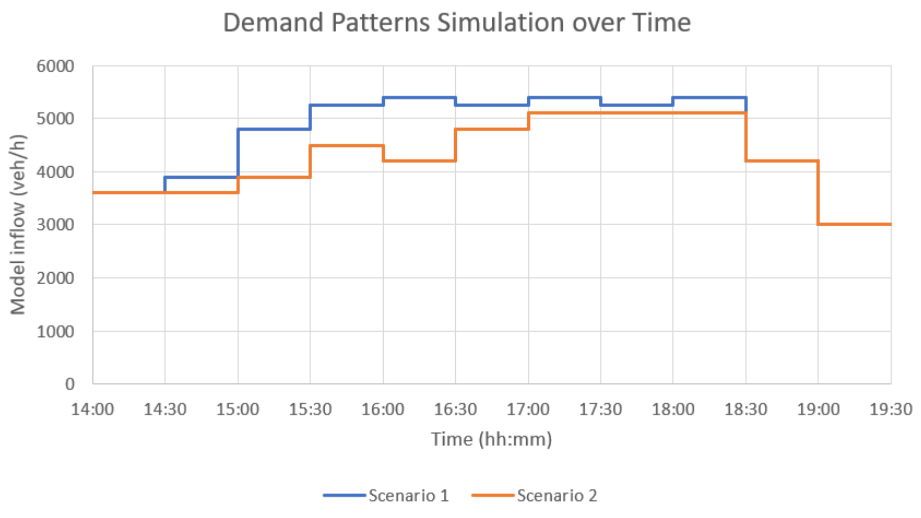
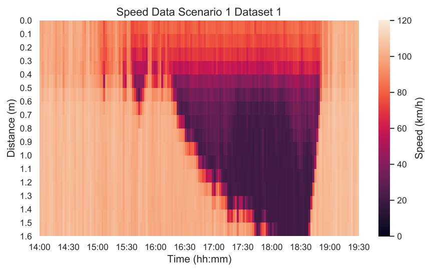


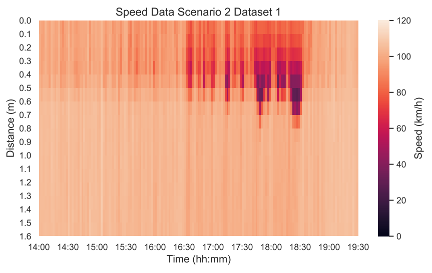
Figure 6.2: Demand patterns of two simulation scenarios.

Table 6.3: Traffic in- and outflow for first dataset (in veh/h).

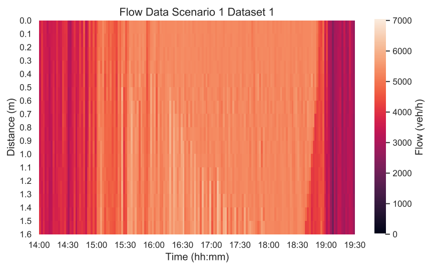
	14:00	14:30	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00
Scenario 1	1200	1300	1600	1750	1800	1750	1800	1750	1800	1400	1000
Scenario 2	1200	1200	1300	1500	1400	1600	1700	1700	1700	1400	1000



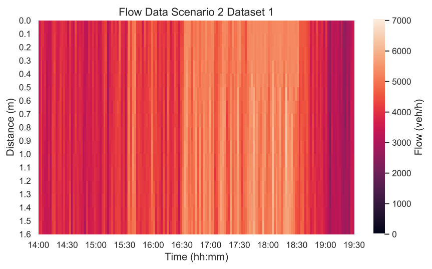
(a) First scenario, speed



(b) Second scenario, speed



(c) First scenario, flow



(d) Second scenario, flow

Figure 6.3: Speed and flow data heatmaps for first dataset.

speeds with which the upstream congestion head moves upstream and downstream. For the second scenario, traffic demand was slightly lower and the resulting shockwave speeds were more diverse.

6.2.2. Second Dataset

Since the first dataset had some shortcomings, some other requirements were formulated for the second dataset. The first requirement was that the road layout and congestion patterns resulting from the simulation should be more complex. By doing so, the baseline will be a less favourable prediction method and more complex prediction methods will most likely perform better. As a second requirement, the road length should increase and the patterns themselves should propagate over a larger distance. Therefore, there will be more information present in the data to base a prediction on, which possibly will lead to much better results.

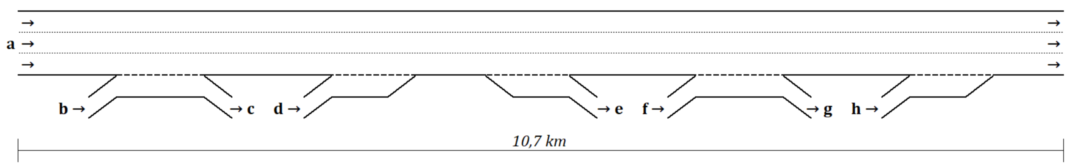
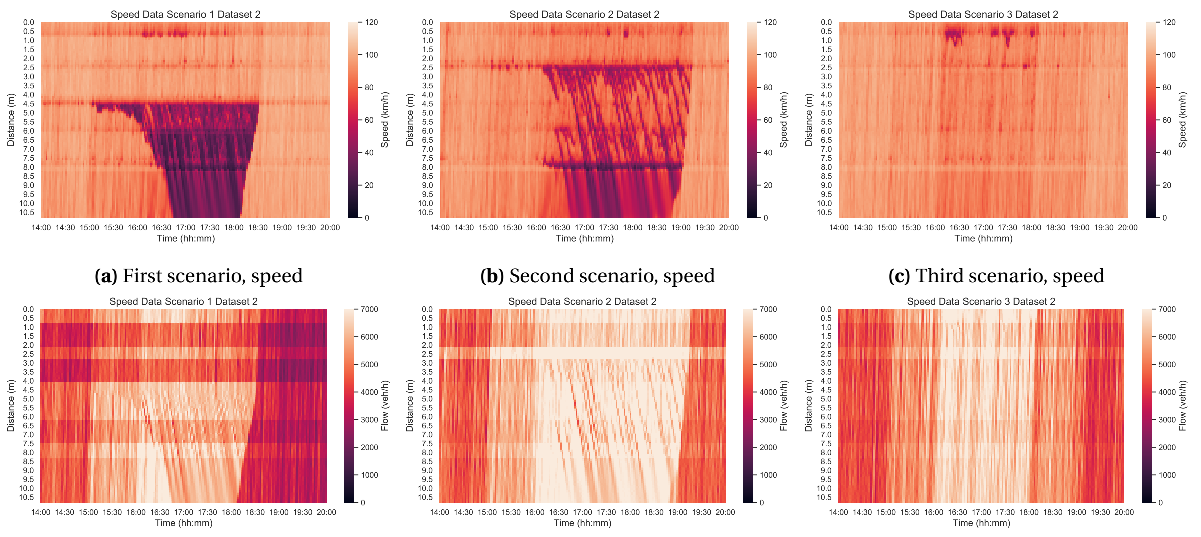


Figure 6.4: Schematic view of road of second dataset.

To meet both requirements, a longer, more complex road with a length of 10,7 km long was created, containing of several on- and off ramps, as can be seen in Figure 6.4. For this road, three traffic demand scenarios have been generated. To simplify the process of loading the model, each scenario consists of a base traffic demand, as can be seen in Table 6.4, which is then per point in time multiplied with a fixed factor, as depicted in Figure 6.6. Although this multiplication factor over time is the same for each scenario, the difference in base traffic demands results in a big difference between the roads. We now zoom in on each scenario to provide a brief description of the scenario and the resulting speed heatmap. Both trucks and cars are present in the vehicle inflow, which leads



(a) First scenario, speed **(b)** Second scenario, speed **(c)** Third scenario, speed

(d) First scenario, flow **(e)** Second scenario, flow **(f)** Third scenario, flow

Figure 6.5: Speed and flow data heatmaps for second dataset.

Table 6.4: Traffic in- and outflow for road second dataset (in veh/h).

Section	a	b	c	d	e	f	g	h
Scenario 1	2600	200	-500	300	-700	500	-300	600
Scenario 2	2600	150	-700	100	-100	600	-150	200
Scenario 3	2400	50	-200	100	-50	200	-100	200

to additional turbulence and increases the complexity of the traffic jam. The resulting speed data can be seen in Figure 6.5

Scenario 1 is characterised by a large moving synchronised pattern (MSP) which starts 4.3 km upstream from the end of the road, as can be seen in Figure 6.5a. As it occurs just upstream from off-ramp **e** (see Figure 6.4), one can conclude an off-ramp induced traffic jam occurs since there is a large amount of vehicles exiting the road at this off-ramp. The resulting lane changing behaviour upstream of this off-ramp causes a reduction in capacity and therefore the occurrence of a traffic jam.

Scenario 2 is characterised by a critical traffic flow from **b** to **g** and high merging and diverging flows on the on- and off-ramps in between, resulting in a complex speed pattern as can be seen in Figure 6.5b. A general pattern occurs as the pattern consists of synchronised flow and stop-and-go waves between weaving section **f/g** and to weaving section **b/c**. A stationary congestion head occurs at the weaving section between **f** and **g**, since the inflow at on-ramp **f** is higher compared to the first scenario. Together with the lane-changing from the off-ramp, stop-and-go waves emerge too from this location, which either solve at off-ramp **e** or propagate downstream to weaving section **b/c**. Between weaving section **b/c** to the upstream end of the road, a larger amount of stop-and-go waves and more consistent synchronised flow conditions occurs.

Scenario 3 is characterised by a lower inflow on the main highway section **a**, and also has far lower in- and outflows at the intermediate on- and off-ramps. Therefore, free-flow conditions occur during most time on the road, as can be seen in Figure 6.5c. However, one can see that between 16:00 and 18:00, the higher traffic flows result in slightly lower traffic speeds, which possibly also is due to the inhomogeneity in speeds between the two vehicle types. Furthermore, some smaller localised synchronised pattern (LSP)s are triggered near on-ramp **h**.

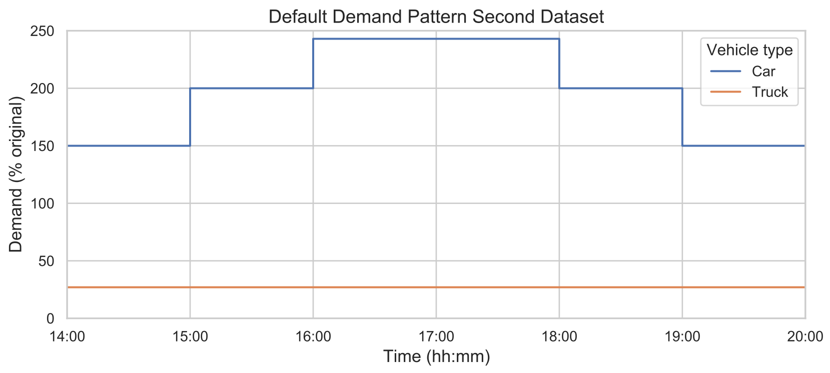


Figure 6.6: Demand patterns of two simulation scenarios.

Table 6.5: Traffic In- and Outflow for Third Dataset (in veh/h)

	14:00	14:30	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00
Scenario 1	4000	4500	5000	6300	6300	6300	5200	5200	4100	4100	4100
Scenario 2	4000	4500	5000	5800	5800	6100	6100	4500	4500	4100	4100
Scenario 3	4000	4500	5000	5000	5000	5100	5200	5200	5000	4000	3500
Scenario 4	4000	4500	5000	5800	5800	5800	5200	5200	4600	4600	4600

6.2.3. Third Dataset

Therefore, for the third dataset, a different approach was taken. The approach was the same as with the first dataset, by taking a road with a lane merge bottleneck and simulating a stationary jam wave. However, two things were changed, namely:

1. the road length was increased from 2 km to 10 km, and
2. per scenario, the propagation speed of the upstream jam head was changed.

This resulted in the road design shown in Figure 6.7. By increasing the road length, more data was available for the neural network to train on, making it easier to predict traffic speeds given the prediction horizon of 20 minutes. Besides having more data for the neural network, the longer road length also resulted in a longer time for the upstream jam head until it reaches the border of the road. This will likely increase the accuracy of the predictions, as the propagation time of the upstream jam head is much lower than the prediction horizon. By changing the shockwave speed of the upstream jam head between the scenarios, one could validate whether the model could make a distinction between different demand patterns leading to the prediction of different speeds with which the jam head propagates.

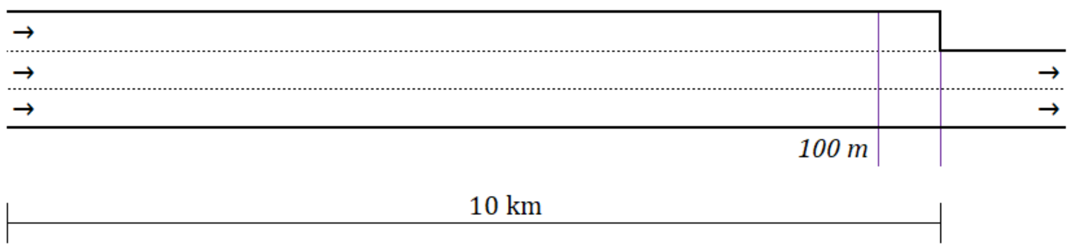
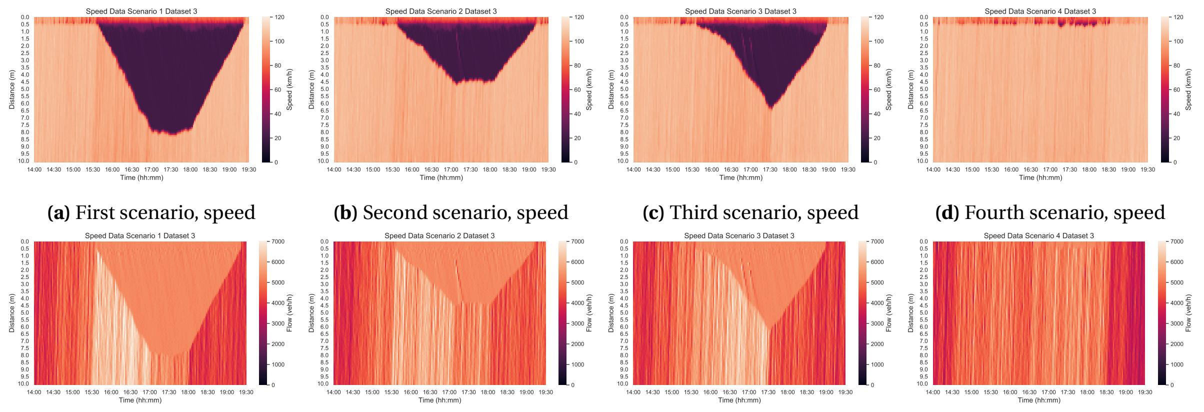


Figure 6.7: Schematic View of Road of Third Dataset

Based on this, four different scenarios were formulated, which all consist of a moving synchronised pattern (MSP) with different shockwave speeds. Each scenario has the same start, between 14:00 and 15:30, where traffic demand was below road capacity leading to free-flow traffic speeds. After this period, the different traffic demand patterns lead to differences in traffic patterns.



(a) First scenario, speed **(b)** Second scenario, speed **(c)** Third scenario, speed **(d)** Fourth scenario, speed

(e) First scenario, flow **(f)** Second scenario, flow **(g)** Third scenario, flow **(h)** Fourth scenario, flow

Figure 6.8: Speed and flow data heatmaps for third dataset

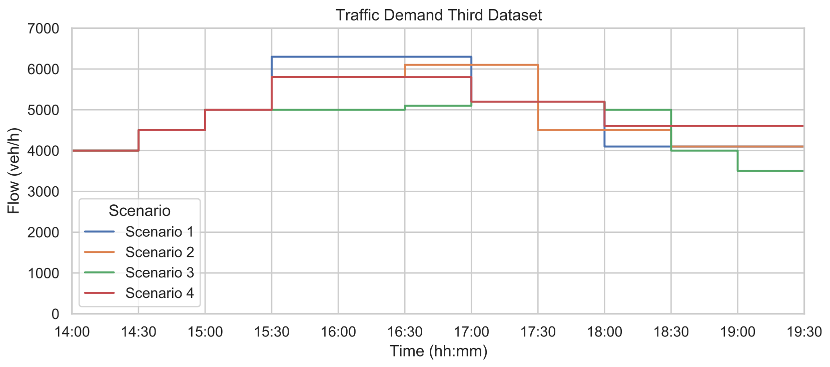


Figure 6.9: Traffic demand of third dataset

Scenario 1 has a high traffic demand between 15:30 and 17:00 ($v_{wave} = -5$ km/h), as well as a low traffic demand between 18:00 and 19:30 ($v_{wave} = 5$ km/h). Between 17:00 and 18:00, a traffic demand was chosen leading to a shockwave speed of approximately 0 km/h, leading to a stabilised jam head location over time. This leads to a high shockwave speed of the upstream jam head both when congestion increases and solves again, so traffic congestion almost moves upstream to the edge of the road, as can be seen in Figure 6.8a.

Scenario 2 is quite similar to the first scenario, although traffic flows are lower between 15:30 and 17:00 ($v_{wave} = -2.5$ km/h) but higher between 18:00 and 19:30 ($v_{wave} = 2.5$ km/h). Therefore, both shockwave speeds are lower and congestion propagates upstream less, leading to a different congestion pattern which can be seen in Figure 6.8b.

Scenario 3 has a different approach as the shockwave speeds of both developing as well as solving congestion vary over time, to see whether these differences can be captured. Therefore, the traffic demand pattern between 15:30 and 19:30 was divided into four sections, where in the first two hours traffic demand gradually increased and the last two hours it gradually decreased. This lead to differences in shockwave speeds over time which are mentioned in Table 6.6.

Scenario 4 is again a scenario without congestion, representing for example the weekends. Although traffic demand is highest between 17:00 and 18:00, speeds remain to be equal to the free-flow speed, as can be seen in Figure 6.8d.

Table 6.6: Shockwave speeds third scenario of third dataset.

	15:30-16:30	16:30-17:30	17:30-18:30	18:30-19:30
v_{wave}	-2 km/h	-4 km/h	3 km/h	5 km/h

6.3. Data Preparation for Neural Network

Table 6.7: Excerpt of SQLite database AIMSUN

did	oid	eid	sid	ent	countveh	countveh_d	flow	flow_d	speed	speed_d	density	density_d	occupancy	occupancy_d	headway	headway_d
611	477	d10	0	1	777.0	-1.0	4662.0	-1.0	74.53	-1.0	61.469	-1.0	36.93	-1.0	-1.0	-1.0
611	478	d09	0	1	783.0	-1.0	4698.0	-1.0	70.47	-1.0	61.84	-1.0	37.15	-1.0	-1.0	-1.0
611	479	d08	0	1	791.0	-1.0	4746.0	-1.0	67.99	-1.0	63.99	-1.0	38.45	-1.0	-1.0	-1.0
611	480	d07	0	1	799.0	-1.0	4794.0	-1.0	56.96	-1.0	73.81	-1.0	44.35	-1.0	-1.0	-1.0

Before the neural networks can be trained on a dataset, the simulation data from AIMSUN had to be transformed so it has the correct structure for training the neural network. The simulation data from AIMSUN was stored in an SQLite database, containing a table in which all detector data was stored, aggregated on a 2-minute level. Each entry consists of one detector observation at one timepoint, as can be seen in a data excerpt in Table 6.7. The original SQLite databases were up to 1.2 GB in size and contained a lot of data columns which were irrelevant for the neural network inputs Therefore, the first step was to delete all irrelevant columns and storing this new dataset, reducing the file size to approximately 25%. Only data related to the simulation run, timepoint, detector ID was stored, as well as flow and speed data aggregated for all vehicle types.

7 Results

After the retrieval and preparation of the data from the experimental setup, the prespecified neural networks have been trained and their results have been analysed. In this chapter, the results and findings of the experiments proposed in the methodology will be presented. In order to select the best performing model, first, several models have been trained according to the hyperparameters specified in chapter 5. The validation error of all models was calculated using the validation dataset in order to select the best hyperparameter combinations for each dataset. These final models were then evaluated using a test dataset to compare and select the best performing model.

The chapter is structured as follows. First, the results of the hyperparameter tuning will be presented for all three datasets, including an analysis of the effects of the hyperparameter combinations on the model accuracy. Secondly, the test errors of the best performing models will be introduced, comparing the prediction results in a qualitative way and presenting the spatial error. Lastly, the results of the sensitivity analysis will be discussed, to get insight in the effect of the input sequence length and prediction horizon on the model accuracy.

7.1. Hyperparameter Tuning

In chapter 5, different hyperparameters have been proposed which were then tested using grid search. A separate validation dataset was used to assess the error metrics of each hyperparameter combination in order to select the best performing model within these hyperparameters. The best hyperparameter combination was then selected for each dataset, input data type and model type. This section shows the validation errors of each of these models and also comments on the differences in accuracy found when changing the hyperparameters.

First, the results of the first dataset will be presented as well as the results of the custom input layers proposed. Secondly, the results of the second dataset will be introduced. Lastly, the results of the third dataset will be discussed.

7.1.1. Validation Errors First Dataset

The first dataset, which has been introduced in subsection 6.2.1, consists of a 2 km long road with a lane merge bottleneck on the downstream end, causing several moving synchronised patterns to occur. The prediction results in general were quite poor compared to the baseline, as can be seen assessing the error metrics of the best performing models within the hyperparameter grid search in Table 7.1. Only the RMSE of the models is lower than the baseline and all best performing models are relatively shallow, having only one layer.

In order to increase prediction accuracy, many different methods have been tested, such as using different loss functions, not scaling the input data, dropout, cross-validation, changing the output activation functions and drastically increasing the number of units per layer, of which the results can be found in Appendix A. Since these options did not lead to an error reduction and the model accuracy was still worse than the baseline, this dataset has not been tested on the

Table 7.1: Results and configuration of best performing models for different model types. The lowest values per error metric are shown in bold.

Model Type	MAE (km/h)	RMSE (km/h)	MAPE (%)	layers	units
speed only, MSE loss	8.94	15.42	18.27	1	32
speed only, MAE loss	**7.61**	16.12	**14.74**	1	32
flow and speed, a (perceptron)	9.32	15.99	19.20	1	32
flow and speed, b (non-linear)	9.29	16.01	19.37	1	32
flow and speed, c (fully connected)	8.90	**15.30**	18.24	1	64
Baseline ($v_{t+20} = v_t$)	*7.26*	*16.33*	*12.82*	-	-

sequence-to-sequence architecture and no sensitivity analysis will be executed. However, since the dataset was used to test several input layer structures to combine spatiotemporal speed and flow data as formulated in subsection 5.3.2, and was proven to be important for the choices of some hyperparameters, this section will focus on the model results with respect to these topics.

Although the dataset had some shortcomings, some results were proven to be relevant with respect to the other datasets and will be shared in this section. First, the model results will be assessed when using speed data as input only, as well as the effect of several loss functions on the resulting errors. Secondly, since this dataset was used to test several input layer structures to combine spatiotemporal speed and flow data as formulated in subsection 5.3.2, these results will be presented too.

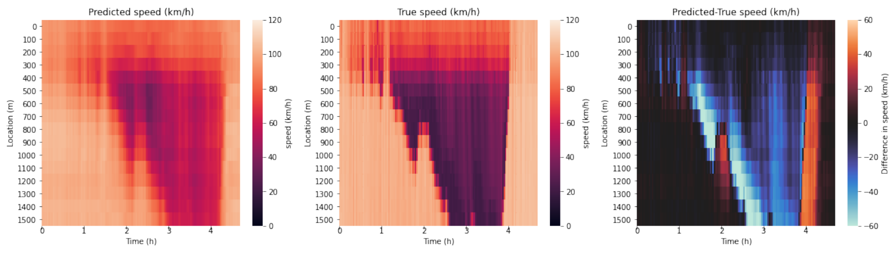
Results Speed Data Only

From Table 7.1 could be concluded that the error metrics of both speed-only models were quite high with respect to the baseline. In order to see what the cause is of these high errors, the resulting speed heatmaps were analysed as shown in Figure 7.1. It seems like the speed predictions look like a copy of the ground truth values but then shifted by 20 minutes, a period equal to the prediction horizon. This indicates the model does not predict traffic based on the input variables, but actually reuses the most recent input data value as prediction, although it does so with a lower accuracy compared to the naive baseline model.

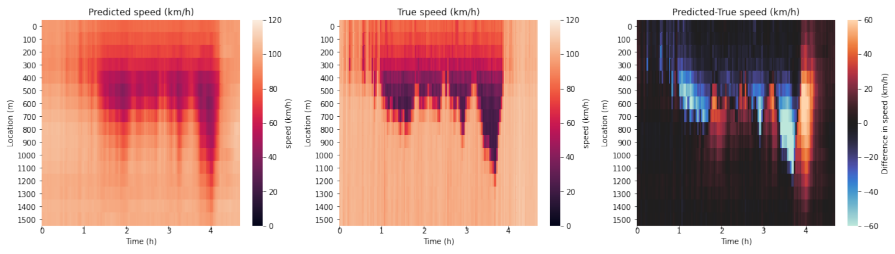
In order to increase the model accuracy, different loss functions have been tested for the speed prediction model, namely both an a MAE and a MSE loss function. When comparing the results in Table 7.1, a significant difference in error metrics can be observed between the two used loss functions. Using a MAE loss function, the MAE and MAPE are relatively low, but using an MSE loss function, the RMSE is lower. To see how the model predictions led to this result, let us qualitatively compare the resulting speed heatmaps of models trained on both loss functions on one exemplary congestion pattern.

When looking at the results when using a MSE loss function (Figure 7.1), a continuous overestimation of the speed during the congested phase can be observed, as the predicted speed is much higher than the true, observed speed. However, when assessing the predicted speed heatmap using the MAE loss function of the same congestion pattern (Figure 7.2), no overestimation of the speed takes place. As a result, the residuals are much higher when the solution of the congestion tail is mispredicted.

This difference in prediction results for the two different loss functions is confirmed by Figure 7.3, in which the predicted speed is plotted against the ground truth speed. When using the MSE loss, one can see that predicted speeds do not fall below 40 km/h, although ground truth speeds reach below 20 km/h. Besides overestimating low speeds, high speeds seem to be underestimated too. For MAE loss, there also is a slight underestimation of high speeds, but the



(**a**) Heavy Congestion Pattern



(**b**) Medium Congestion Pattern

Figure 7.1: Speed heatmaps of prediction result for the model with the highest accuracy when trained on dataset 1, using only speed data and a MSE loss function.

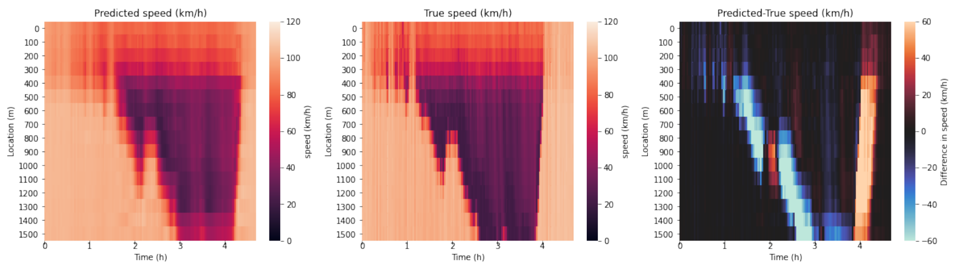


Figure 7.2: Speed heatmaps of prediction result for the model with the highest accuracy when trained on dataset 1, using only speed data and a MAE loss function.

overestimation of speeds when in congestion is largely absent. The large concentration of observations at $[v_{pred}, v_{gt}] = [25, 105]$ or $[105, 25]$ km/h for both loss functions shows that the model does not correctly predict the phase changes between free-flow and congestion.

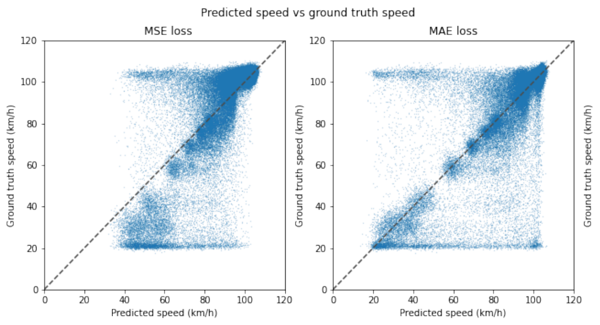


Figure 7.3: Predicted speed versus ground truth speed

Combining Flow and Speed Data

In subsection 5.3.2, several input layer structures have been proposed combining the flow and speed data belonging to a road segment at a certain time. It was assumed these custom structures had several advantages over a fully connected approach, as the number of trainable parameters would be smaller and parallels could be drawn with the concept of traffic phases within traffic flow theory.

Table 7.2: Results of grid search for models trained on the second dataset. For each data and loss function combination, the lowest values of each error metric have been made bold, whereas the model with the overall lowest error has been highlighted in green.

			MAPE (%)			RMSE (km/h)			MAE (km/h)		
Data	Loss	Layers	108*	216*	432*	108*	216*	432*	108*	216*	432*
Speed	MAE	1	7.80	7.76	7.84	8.41	8.33	8.38	5.04	4.98	5.01
		2	**7.55**	7.71	7.80	**8.20**	8.25	8.30	**4.92**	5.00	5.00
		4	7.84	7.83	7.81	8.32	8.34	8.41	5.10	5.01	5.07
	MSE	1	9.52	9.50	9.55	8.84	**8.78**	8.84	5.93	**5.83**	5.86
		2	9.61	9.64	9.59	8.81	8.90	8.88	5.89	6.02	5.97
		4	**9.46**	9.75	9.72	8.74	9.11	8.95	5.87	6.19	6.11
Speed + flow	MAE	1	7.55	7.53	7.66	8.14	8.09	8.42	4.89	4.87	4.95
		2	7.53	**7.43**	7.51	8.21	**8.07**	8.08	4.93	4.83	**4.82**
		4	7.67	7.60	7.48	8.22	8.24	8.27	4.94	4.88	4.86
	MSE	1	9.14	**9.14**	9.28	8.54	**8.52**	8.65	5.72	5.69	5.75
		2	9.40	9.38	9.58	8.63	8.62	8.76	5.73	**5.67**	5.82
		4	9.42	9.38	9.58	8.61	8.63	8.75	5.70	5.71	5.89
Baseline	-	-		*14.05*			*15.71*			*8.98*	

* = units per hidden layer

Table 7.3: Results of grid search sequence-to-sequence for models trained on the second dataset. For each data and loss function combination, the lowest values of each error metric have been made bold, whereas the model with the overall lowest error has been highlighted in green.

		MAPE (%)			RMSE (km/h)			MAE (km/h)		
Data	Loss	108*	216*	432*	108*	216*	432*	108*	216*	432*
Speed	MAE	7.86	7.28	**7.05**	8.51	**8.06**	8.20	4.88	**4.74**	4.76
	MSE	8.58	**7.80**	8.33	8.37	**8.00**	8.28	5.26	**5.09**	5.14
Speed + flow	MAE	**8.47**	10.98	10.73	**8.91**	11.97	11.44	**6.06**	8.20	7.86
	MSE	10.35	8.92	**8.51**	10.55	9.10	**8.54**	7.39	6.31	**6.01**
Baseline	-		*14.05*			*15.72*			*8.98*	

* = units per hidden layer

Variant (a) used a perceptron to connect flow and speed, variant (b) a small feed-forward neural network (FFNN) combining flow and speed data per segment and variant (c) connected all input parameters directly to the first GRU layer in a fully connected way. The FFNN in variant (b) consists of one hidden layer having four units, each having a ReLU as output activation function. The results of the best performing model structures, presented in Table 7.1, show that variant (c) using the fully connected input layer performed significantly better than the other two. Besides, the more complex non-linear approach of variant (b) performed worse than the perceptron approach of (a). Since the results of variant (c) were significantly better, the conclusion can be drawn that designing a custom input layer will not improve the accuracy and that in this case, accuracy decreases when complexity increases. Therefore, on the following datasets, only variant (c) will be used when combining flow and speed.

7.1.2. Validation Errors Second Dataset

The second dataset, as explained in subsection 6.2.2, consisted of three scenarios: a free-flow scenario, a scenario with moving synchronised patterns and a scenario with general patterns containing many stop-and-go waves. The results of the one-shot model are presented in Table 7.2 and for the sequence-to-sequence model in Table 7.3, where the lowest error metrics per dataset and loss function are in bold and the overall best model is marked green. In general, the sequence-to-sequence model has a lower error than the one-shot GRU RNN model, although the difference between both models is quite small. When looking at the two models, no clear conclusions can be drawn with respect to the best performing loss function, input variables and number of layers and neurons. To go into more detail, a short assessment will be given per model type.

When assessing the results of the hyperparameter grid search for the *one-shot model* in Table 7.2, a couple of things can be noticed. The lowest errors are achieved when using speed and flow data as input, having a model with two GRU RNN layers consisting of 216 units which was trained using a MAE loss function. This best performing model, as well as all other models, performed significantly better than the baseline. Zooming in on each of the hyperparameters individually gives the following conclusions. First, using a MAE loss function leads to a significantly lower error compared to a MSE loss function for all three error metrics, including the RMSE. For all layer, unit and node combinations, models using both flow and speed data as inputs perform better than models using only speed data, although the difference is small (MAPE: -0.12 %pnt, RMSE: - 0.13 km/h, MAE: -0.10 km/h). The number of hidden layers and neurons per layer have no clear effect on the model accuracy as all error metrics are quite similar for different layer-unit combinations. However, a clear optimum in the model structure can be found, as both the deepest as well as the shallowest models led to an increase in error.

When looking at the *sequence-to-sequence model* results in Table 7.3, the best performing model uses only speed data, has 216 units and was trained using a MAE loss function (as both RMSE and MAE are lowest for this model type). It is quite striking that using speed and flow as input data increases the model error significantly, opposite to the one-shot model. Besides, it seems the models using flow and speed data have less stable results, as there are larger fluctuations in error observed with changing the number of units per layer compared to the one-shot model. The effect of the loss function on the model results differs from the one-shot model in two ways. First, no loss function was found to give the overall lowest errors, as the lowest MAE and MAPE is obtained using an MAE loss function, whereas a MSE leads to a lower RMSE value. Secondly, the differences in error metrics between the two loss functions are not as big as for the one-shot model.

7.1.3. Validation Errors Third Dataset

The third dataset, as presented in subsection 6.2.3, consists of four different scenarios: one free-flow scenario and three scenarios consisting of a localised synchronised pattern with different shockwave speeds. When comparing the results of the one-shot GRU RNN (Table 7.4) with the sequence-to-sequence GRU RNN (Table 7.5), the sequence-to-sequence model has a significantly better accuracy compared to the one-shot model. For both model types, the speed and flow dataset in combination with a MAE loss function results in the highest accuracy. As some differences could be observed between the two model types, their results will be assessed separately.

For the *one-shot model*, the lowest errors were observed when the speed and flow dataset and a MAE loss function were used for training, on a model with 4 layers which each consist of 202 units. This is quite similar to the second dataset, although the optimal number of layers is higher (4 instead of 2 layers).

One can see that the loss function had a significant influence on the model quality. Predictions are much worse using a MSE loss function (Figure 7.4b) as there is no sharp border between the two traffic phases, but it seems the speed values are gradually shifting around the shockwave separating the two speeds. When using a MAE loss function (Figure 7.4a) the prediction errors are limited to mispredictions of the position of this shockwave. The accuracy of the predictions is quite good as the residuals are small and the predicted speed heatmap accurately resembles the true speed

Table 7.4: Results of grid search for models trained on the third dataset. For each data and loss function combination, the lowest values of each error metric have been made bold, whereas the model with the overall lowest error has been highlighted in green.

			MAPE (%)			RMSE (km/h)			MAE (km/h)		
Data	Loss	Layers	101*	202*	404*	101*	202*	404*	101*	202*	404*
Speed	MAE	1	9.66	9.40	9.39	10.68	10.75	10.72	4.18	3.99	3.95
		2	9.31	8.52	8.85	10.47	10.27	10.47	3.99	3.70	3.83
		4	**8.20**	8.56	8.91	**10.20**	10.23	10.39	3.73	**3.70**	3.79
	MSE	1	18.01	18.06	17.83	12.94	12.87	12.81	7.92	7.65	7.68
		2	17.83	17.56	17.34	12.73	12.71	12.65	7.55	7.45	7.47
		4	**17.10**	17.58	17.30	12.64	**12.60**	12.65	7.55	**7.42**	7.58
Speed + flow	MAE	1	8.22	7.49	7.55	9.68	9.20	9.34	3.64	3.57	3.55
		2	8.26	7.43	7.22	9.66	9.24	9.23	3.82	3.49	**3.48**
		4	7.53	**7.08**	7.26	9.28	**9.14**	9.24	3.63	3.50	3.49
	MSE	1	17.40	17.23	17.30	12.48	12.42	12.49	7.43	7.33	7.46
		2	17.19	17.11	**16.83**	12.32	12.32	12.36	7.21	**7.21**	7.34
		4	17.01	16.91	16.84	**12.30**	12.38	12.47	7.21	7.32	7.49
Baseline	-	-		*25.73*			*25.27*			*10.28*	

* = units per hidden layer

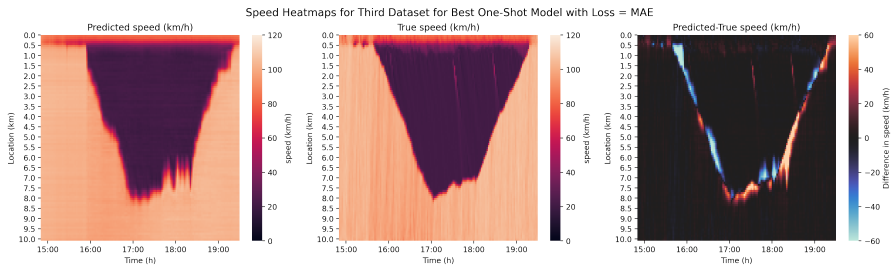
Table 7.5: Results of grid search sequence-to-sequence model third dataset. For each data and loss function combination, the lowest values of each error metric have been made bold, whereas the model with the overall lowest error has been highlighted in green.

		MAPE (%)			RMSE (km/h)			MAE (km/h)		
Data	Loss	101*	202*	404*	101*	202*	404*	101*	202*	404*
Speed	MAE	**6.51**	7.39	7.54	10.77	**10.56**	10.88	3.64	**3.54**	3.68
	MSE	**7.83**	8.27	8.72	**9.71**	10.04	9.97	**3.88**	4.09	4.17
Speed + flow	MAE	**5.21**	5.87	5.33	8.62	**8.21**	8.71	**2.84**	3.09	2.88
	MSE	6.22	**5.31**	6.18	**8.24**	8.98	8.42	3.20	**2.97**	3.33
Baseline	-		*25.73*			*25.27*			*10.28*	

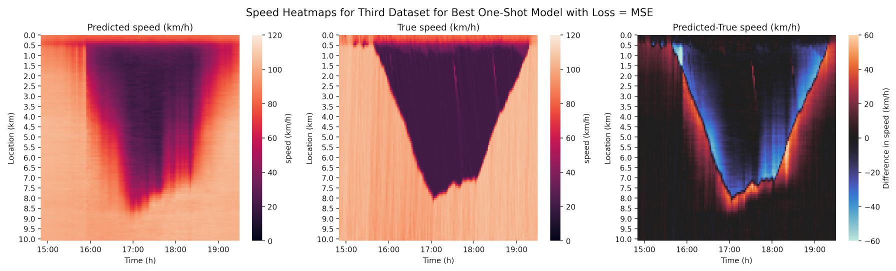
* = units per hidden layer

heatmap. These differences cause the error metrics to be much higher using the MSE loss function compared to the MAE loss function.

For the sequence-to-sequence model, the best results were achieved using both flow and speed input data, a MAE loss function and 202 hidden units. For all models, including speed and flow data resulted in a significant decrease in error metrics. The effect of the number of hidden units on the prediction error is less significant, although using 404 hidden nodes clearly leads to worse results compared to the other combinations. In general, using a MAE loss function leads to lower values for all error metrics, although the difference is not as big compared to the one-shot model. No major differences can be found between the predictions and residuals between the two loss functions, as their predictions and residuals shown in Figure 7.5 look very much alike. However, especially when the upstream congestion tail location remains stable over time, one can see that a MAE loss function results in a sharper difference in speed between the two phases. The sequence-to-sequence model training seems to be much more robust, as in this case neither model did not get stuck in a local minimum during training.

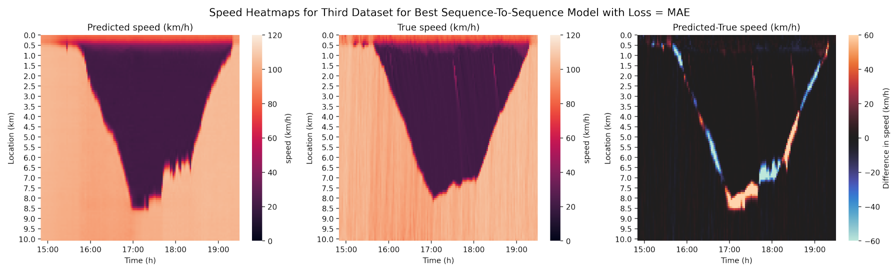


(a) Heatmaps using MAE loss function

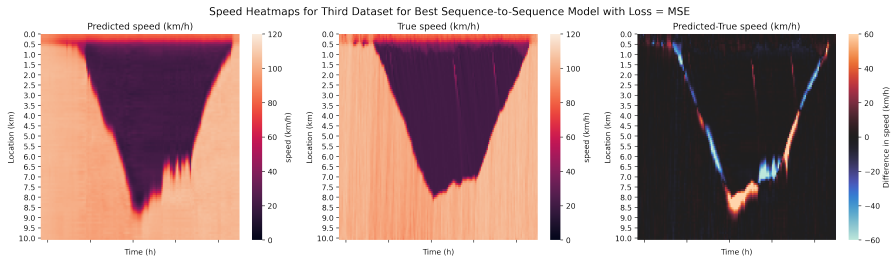


(b) Heatmaps using MSE loss function

Figure 7.4: Speed heatmaps of prediction result for best one-shot model when having a MAE or MSE loss function



(a) Heatmaps using MAE loss function



(b) Heatmaps using MSE loss function

Figure 7.5: Speed heatmaps of prediction result for best sequence-to-sequence model when having a MAE or MSE loss function.

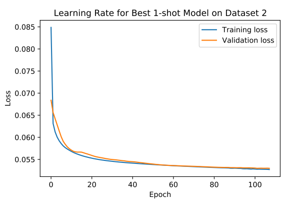
7.2. Model Test Results

After assessing the effect of hyperparameter tuning using validation data, the loss function, number of hidden layers and number of units per hidden layer could be fixed for each combination of input data and model type per dataset. By calculating the test errors of these best performing models, the best performing model structure in general could be given. By also analysing the resulting speed heatmaps qualitatively, the implications of these differences in error between the models can be interpreted.

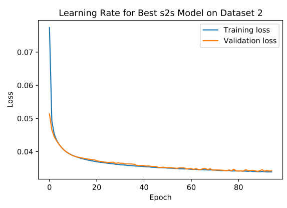
This section is structured as follows. First, the test errors will be discussed for both datasets. Secondly, the prediction results of the most accurate GRU RNNs will be analysed qualitatively. Lastly,

Table 7.6: Test errors of the models having lowest training error per dataset, data types and model type, both for the second and third dataset. The lowest error metric values have been made bold.

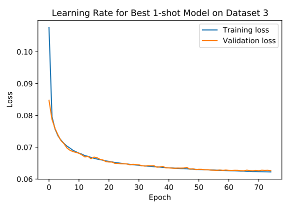
		Dataset 2			Dataset 3		
Model	Data	MAPE (%)	RMSE (km/h)	MAE (km/h)	MAPE (%)	RMSE (km/h)	MAE (km/h)
S2S	Speed	**7.16**	**8.09**	**4.73**	6.41	10.84	3.64
	Speed+flow	7.67	8.13	5.06	**5.12**	**8.68**	**2.85**
1-shot	Speed	7.58	8.35	4.96	7.89	10.01	3.67
	Speed+flow	7.38	8.12	4.83	6.91	9.10	3.47
Baseline	-	*14.06*	*15.71*	*8.98*	*25.73*	*25.27*	*10.28*



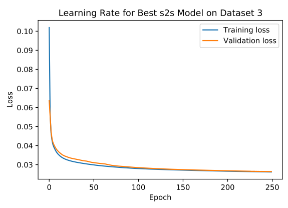
(a) Dataset 2, one-shot, speed+flow



(b) Dataset 2, s2s, flow



(c) Dataset 3, one-shot, speed+flow



(d) Dataset 3, s2s, speed+flow

Figure 7.6: Learning rates for best models of test set.

the error terms will be shown per segment, to see how the errors differ over space. Individual speed heatmaps of each model considered in this chapter can be assessed in Appendix B.

7.2.1. Test Error Results

When assessing the test errors for the four different model type and input data combinations on the second and third dataset, as shown in Table 7.6, the test errors do not differ much from the validation errors. For both datasets, the sequence-to-sequence model led to the best results, although the best performing dataset differed. Although adding flow data resulted in a significant drop in errors for dataset 3, it led to an increase in MAE and MAPE for dataset 2.

The learning rates, present in Figure 7.6, show that the learning process was stable, and all loss function values converged well within the 250 learning epochs. However, since both the validation and training loss converge to the same value, we see that both datasets are quite similar and the model might overfit or have limited generalisation power, as explained in chapter 3. In case the maximum number of epochs is smaller, the early stopping mechanism intervened and the training process was halted. Only for the sequence-to-sequence model of the third model, the early stopping did not intervene, so it was trained over the full 250 epochs.

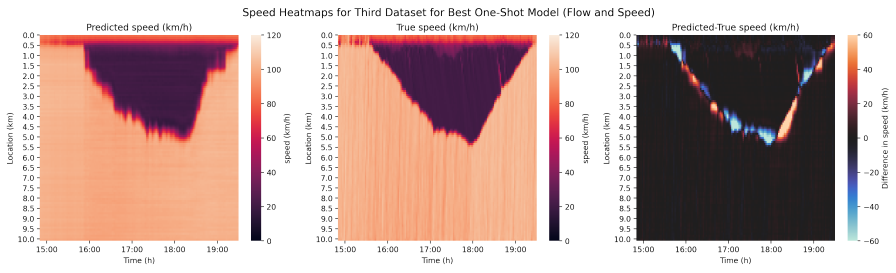
7.2.2. Qualitative Comparison of Model Configurations

For the best one-shot and sequence-to-sequence models of both datasets, a speed heatmap of each scenario has been included in Appendix B. This section will zoom in on the most interesting findings found from these heatmaps. When assessing the heatmaps of dataset 2, one can see that the predictions look quite accurate. However, both the one-shot and the sequence-to-sequence model are unable to correctly capture the propagation of the upstream jam head for the first scenario (containing an MSP) and the exact positions of the stop-and-go waves for the second scenario (containing a GP). However, for the third dataset, both models could quite accurately capture the shockwave speeds for all different models.

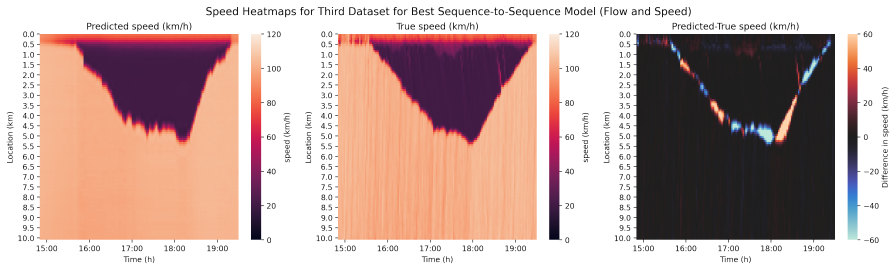
In this paragraph, the effect of both the model type as well as the input data on the prediction quality will be assessed in order to explain the differences found in the error metrics.

Model Type

For both datasets, the best performing model was a sequence-to-sequence model structure, especially for the third dataset. When qualitatively assessing the results of congestion patterns on a one-shot and sequence-to-sequence model trained on flow and speed data of the third dataset, a couple of differences can be noted, which are also visible in Figure 7.7. Firstly, the sequence-to-sequence model seems to predict the shockwaves between traffic phases in a much better way. Secondly, it seems as if the sequence-to-sequence model can better capture the traffic breakdown at the bottleneck location.



(a) Speed heatmaps for one-shot model



(b) Speed heatmaps for sequence-to-sequence model

Figure 7.7: Speed heatmaps of prediction result on third dataset for sequence-to-sequence and one-shot model having the highest accuracy.

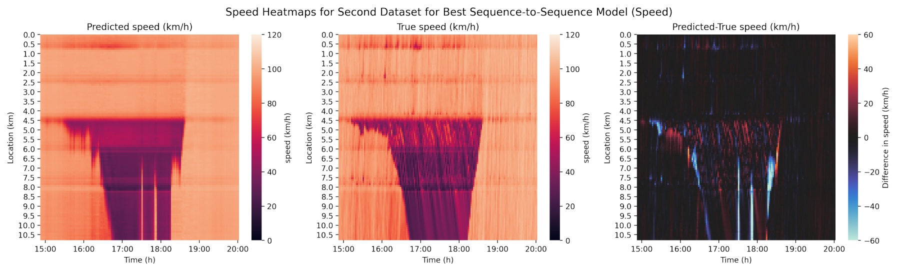
Let us first zoom in on the differences in shockwave prediction. In the one-shot model, the predicted speed heatmap has rough, 'low-resolution' borders between the free-flow and congested phase, as the shockwave does not propagate smoothly and there is a larger zone of 'averaged' speeds between the free-flow and congested phase. The shockwaves resulting from the sequence-to-sequence model seem to be better predicted and also smoother compared to the one-shot mode. Although there are some mispredictions of the shockwave speed when looking at the residuals, the overall prediction result looks much better. A similar observation can be made when comparing speed heatmaps from dataset 2 of the best one-shot model (Figure B.1b) and the best sequence-to-sequence model (Figure B.2c) in Appendix B, as the sequence-to-sequence model more accurately predicts the breakdown of the first stop-and-go wave compared to the one-shot model.

Besides, it seems as if the sequence-to-sequence model can capture the traffic breakdown process in a better way, as the error around t=15:40 and x = 0.5 km is much lower for the sequence-to-sequence model compared to the one-shot model. Since temporal data is not encoded in the RNN input data, traffic breakdown can be predicted by looking at correlations due to an increase in flow or the decrease in speed present before traffic breakdown (as can be seen in Figure 6.5). Apparently, the sequence-to-sequence model is better at capturing these correlations than the one-shot model.

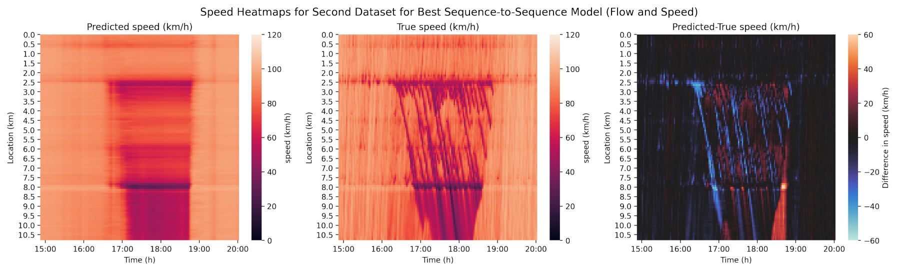
Input Data

When looking at the effect of input data, only the sequence-to-sequence models will be assessed as these were the best performing models. Since dataset 3 led to different conclusions with respect to the input data to be included compared to dataset 2, both will be assessed separately.

For the second dataset, including flow in the input data led to a small increase in error, albeit not very high. When looking qualitatively at the difference in the resulting heatmaps, both predictions seem to be of rather low quality. In Figure 7.8, one can see that individual stop-and-go waves cannot be captured by both models.



(a) Speed heatmaps for model trained on speed data only.



(b) Speed heatmaps for model trained on speed data only.

Figure 7.8: Speed heatmaps for a model trained on the second dataset, for a scenario containing general patterns with stop-and-go waves. The models having the highest accuracy when trained on either speed data or both flow and speed data are shown..

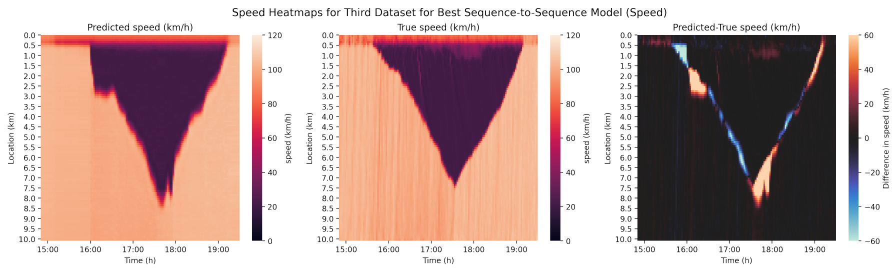
For the third dataset, including flow data led to a significant decrease of all error metrics compared to only flow data. After analysis of the resulting predicted speed heatmaps, it seems that including flow data leads to a better prediction of the time at which the congestion pattern emerges or starts solving, as can be seen by the example in Figure 7.9. Looking at the model using flow and speed data, this can be observed at two locations. When looking at the emergence of the jam wave around 15:45, this model predicts the time at which the congestion emerges fairly accurately and also correctly predicts the speed with which the shockwaves propagates upstream. When looking at the point at which the jam wave starts solving, around 17:40, it accurately predicts the point in time after which traffic flow decreases and the congestion tail propagates upstream, although the position of the upstream jam head is not predicted correctly. From the prediction results of the model using only speed data can be observed that the model initially mispredicts the emergence of the congestion and also suffers from an ’overshoot’, where it initially predicts the shockwave will move further upstream than it actually will. The GRU RNN also has trouble predicting the time at which the congestion starts to solve, as one can see it predicts the congestion tail moves upstream further than it actually does.

7.2.3. Results Custom Error Metrics

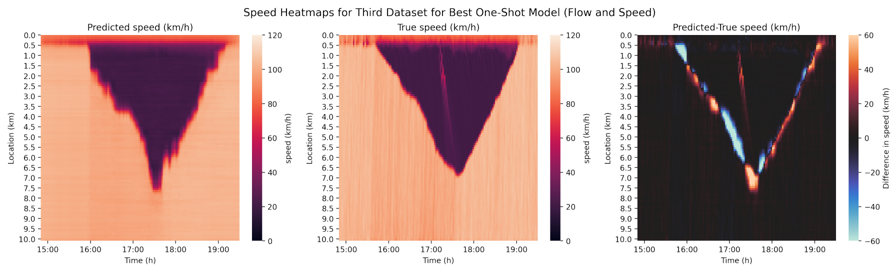
In subsection 5.6.4, it was suggested to also assess the error metrics on a spatial and/or temporal level. By calculating error metrics for each segment, one can see whether there are large spatial or temporal differences in the performance of the GRU RNN. In this section, the results of these error metrics will be presented for both the second and third dataset. For the model with the highest accuracy of each dataset, the spatial error metric will be presented first, the temporal error metric secondly and lastly, the spatiotemporal error metric results will be shown.

Dataset 2

For the highest model of the second dataset, which was the sequence-to-sequence model with only speed data, the results of the *spatial error metrics* can be seen in Figure 7.10. One can see that the error is the lowest at 0 km and between 1 and 2 km, as in all scenarios traffic is in free-flow around these positions. Between 2 and 4 km, the error increases more as the stop-and-go waves of the



(a) Speed heatmaps for model having speed input data.



(b) Speed heatmaps for model having flow and speed input data.

Figure 7.9: Speed heatmaps of prediction result on third dataset for sequence-to-sequence model having either flow or flow and speed input data.

second scenario emerge here. Between 4.5 and 10.7 km, the error increases again since in this zone, both the stop-and-go waves of the second scenario and the MSPs of the first scenario emerge. The location of the on- and off-ramps also shows a clear correlation between the spatial error and the on- and off-ramp locations. Errors increase upstream from off-ramps, due to lane changing turbulence, or downstream from on-ramps, due to an increase in traffic flow. The higher errors at these locations are in line with the expectations, since these situations are prone to cause traffic breakdowns.

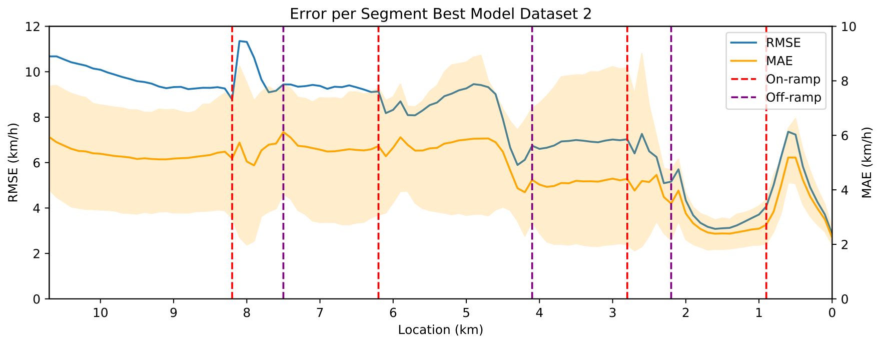
Analysing the interquartile range (IQR) of the MAE allows us to quickly retrieve insights related to differences in errors over the different scenarios. The confidence interval is largest between 2.5 and 4 km, as congestion only occurs in this area for one simulation scenario. It is lowest between 0 and 2 km as in this area, congestion is either fairly stable over the scenarios or non-existent. As errors generally increase when congestion occurs but decrease if traffic flows at free-flow, large fluctuations in MAE can be expected at locations where congestion occurs irregularly.

Assessing the *temporal error metrics* in Figure 7.11, a couple of conclusions can be drawn more clearly compared to looking at the qualitative heatmaps. One can see instantaneously that both the error as well as the IQR increases during peak hours and then decreases in the shoulder regions. At 16:20, a small peak can be observed due to congestion caused by an increase in demand. A much larger peak can be observed at 18:20, which coincides with the start of congestion solution. Qualitative assessment of the prediction heatmaps already showed that large errors can be observed around this time, which becomes even clearer when looking at this graph.

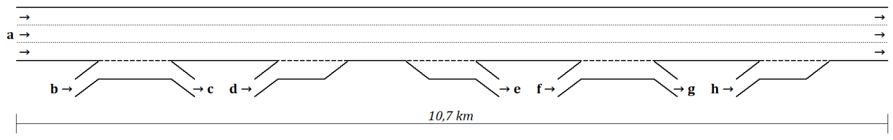
The *spatiotemporal error metrics* in Figure 7.12 summarise the conclusions that could be drawn from the spatial and temporal error metrics and show quite some overlap with the qualitative speed heatmaps made. It confirms that the highest errors are observed during the solution of the MSP patterns, where little stop-and-go waves are present. One can also see that the network has difficulties with capturing the finer stop-and-go wave patterns. As the heatmap is quite noisy due to the various scenarios which are present in the data, a spatiotemporal error heatmap per scenario can be found in Appendix D.

Dataset 3

For the third dataset, the *spatial error metrics* show that difference in error over space varies more than for the second dataset. The error is highest at 0.6 km, as this is the location where the down-



(**a**) Error over space of best performing model on dataset 2.



(**b**) Road layout dataset 2.

Figure 7.10: MAE/MAPE error metrics assessed over space for the highest accuracy model of dataset 2. The coloured band over the MAE represents the interquartile range (IQR) of the MAE values per replication.

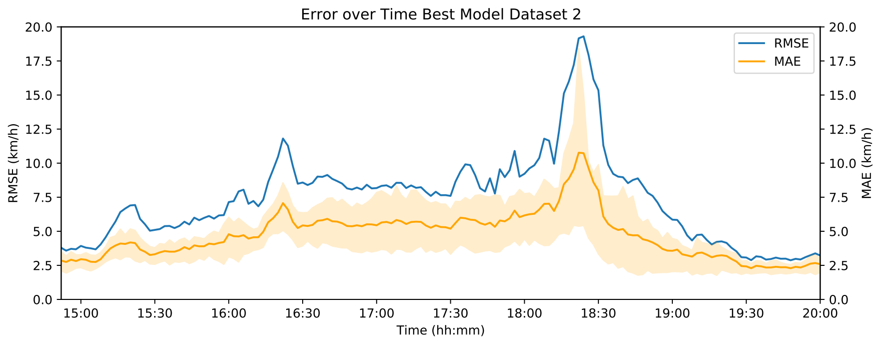


Figure 7.11: MAE/MAPE error metrics assessed over time for the highest accuracy model of dataset 2. The coloured band over the MAE represents the IQR of the MAE values per replication.

stream congestion head is situated and the MSPs emerge. After this initial peak, error stabilise but smaller peaks are present at 4 km and between 7 and 8 km. This coincides with the most downstream locations of the congestion heads in the scenarios. The error decreases rapidly between 9 and 10 km since traffic is always in free-flow at this section. One can also see that the RMSE increases more compared to the MAE when congestion occurs.

The *temporal error metrics* in Figure 7.14 show that, similar to the second dataset, errors increase during peak hours and then decrease in the shoulder regions. At 15:50, a peak can be observed since an increase in traffic flow initiates congestion for all scenarios. At 17:50 and 18:20, an error peak is visible which coincides with the start of congestion solution for different scenarios. One can see that around these two peaks, the 95% confidence interval is largest, which again is caused by variability in the scenarios.

The *spatiotemporal error metrics* in Figure 7.15 are much noisier compared to those of the second dataset, probably caused by the larger variability in scenarios. Therefore, an spatiotemporal error metric per scenario can be found in Appendix D However, they enable to draw some conclusions regarding the error which cannot be seen when looking at only space or time. First, one can see that the higher error during peak hour is caused by mispredicting the shockwave speed between free-flow and congestion, as the error is larger around the contours of the congested phases. Secondly, one can see that a larger error is not concentrated on a location or time separately, but rather a combination of space and time. The largest errors are observed at 0.6 km around 15:50, which corresponds to the traffic breakdown at the downstream congestion head, and at 4 km and 6-8 km

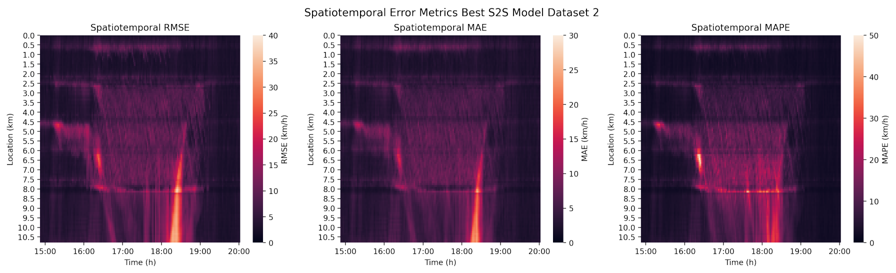
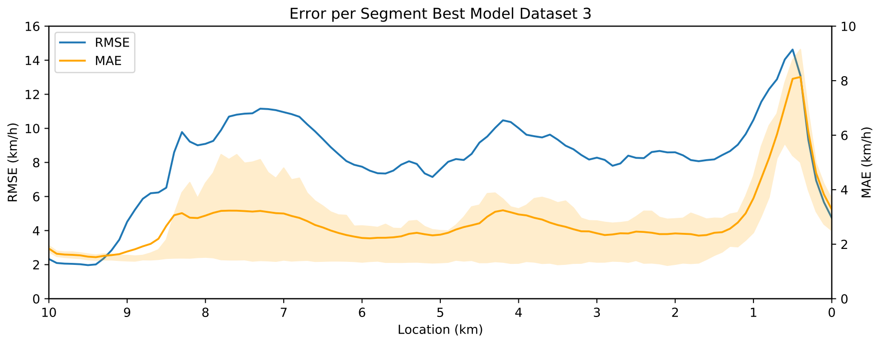
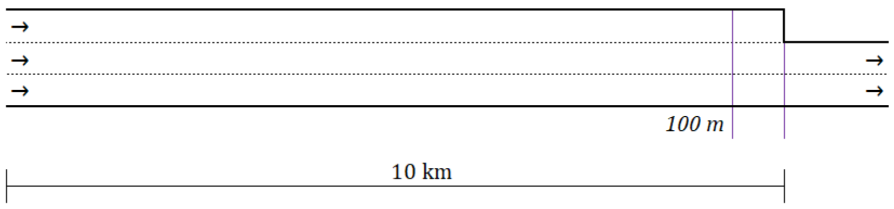


Figure 7.12: Error metrics calculated over space and time for the highest accuracy model of dataset 2.



(**a**) Error over space of best performing model on dataset 3.



(**b**) Road layout dataset 3.

Figure 7.13: Error over space of best performing model and road layout of dataset 3.

around 17:50 and 18:20, which corresponds to the traffic solution from the upstream congestion tail. Compared to the spatial and temporal error metrics, the heatmaps show that large errors occur at a combination of space and time, either at the start of peak hour near the congestion head or the end of peak hour near the congestion tail.

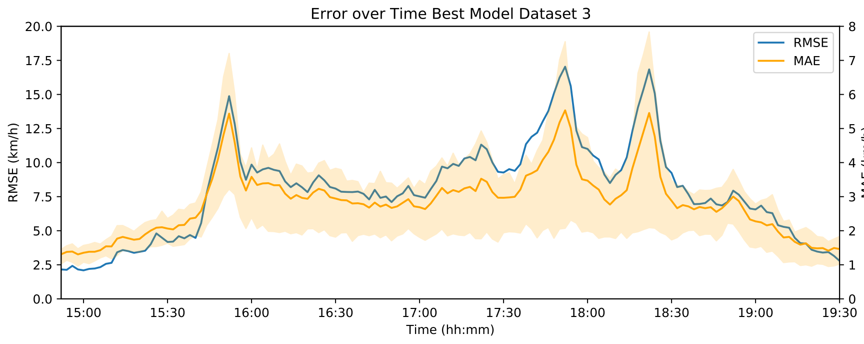


Figure 7.14: MAE/MAPE error metrics assessed over time for the highest accuracy model of dataset 3. The coloured band over the MAE represents the 95% confidence interval of the MAE values per replication.

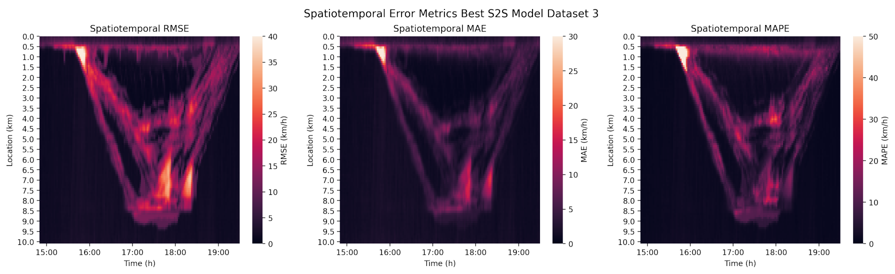


Figure 7.15: Error metrics calculated over space and time for the highest accuracy model of dataset 3.

7.3. Sensitivity Analysis

In section 5.5, a sensitivity analysis was proposed to validate whether the chosen prediction horizon and input sequence length led to the best results, since it was assumed the prediction horizon had a big impact on the prediction errors. In this section, the results of this sensitivity analysis will be discussed. In Appendix C, the speed heatmaps can be found of the best models for each dataset and prediction horizon.

7.3.1. Dataset 2

Looking at the sensitivity analysis in Table 7.7, one can see that the lower the prediction horizon is, the lower the observed error is, as the decrease in RMSE and MAE is almost linearly. This is as expected in the hypotheses and not very surprising. However, it is also striking to see that the input sequence length has a big effect on the prediction accuracy, especially the input sequence having a length of 25 steps. As the prediction horizon increases, the accuracy of this longest input sequence increases as well, leading to significantly lower errors with a prediction horizon of 10 steps. Although it cannot be concluded that the input sequence length should increase with an increasing prediction horizon.

When assessing the results qualitatively, the biggest difference could be observed on the scenario where many stop-and-go waves emerged from the bottleneck. One can see that at lower pre-

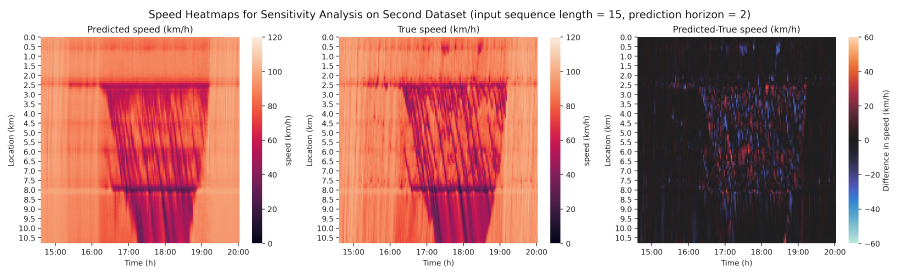
Table 7.7: Test errors of sensitivity analysis best model on dataset 2. Per prediction horizon, the lowest error metric values have been made bold.

	MAPE (%)			RMSE (km/h)			MAE (km/h)		
Prediction Horizon	10*	15*	25*	10*	15*	25*	10*	15*	25*
2 steps	4.54	**4.52**	5.00	4.95	**4.92**	5.26	3.18	**3.17**	3.43
5 steps	**5.83**	5.90	6.04	**6.35**	6.38	6.46	**3.96**	3.99	4.03
10 steps	7.30	**7.19**	7.41	8.27	8.50	**7.88**	4.78	4.85	**4.70**

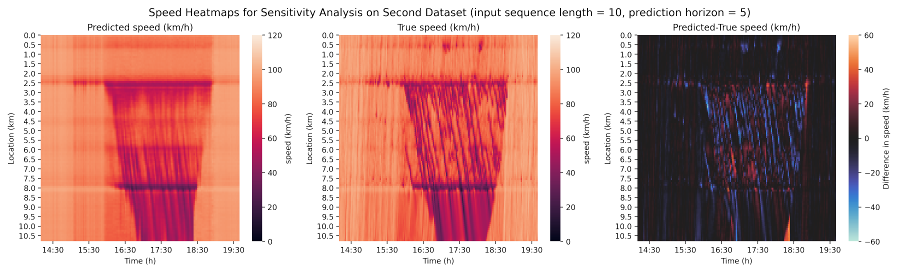
* = input sequence length, in steps ($\Delta t = 2$ min)

diction horizons, the model is able to predict the propagation of the stop-and-go in a better way. When the prediction horizon is two timesteps (4 minutes), one can see in Figure 7.16a that the stop-and-go waves are clearly visible in the prediction and the speed with which they propagate backwards is (approximately) equal to the speed observed in the ground truth. However, looking at the residuals, a fine pattern of underpredicted speeds is visible which indicates that some mispredictions still happen.

When the prediction horizon is five timesteps (10 minutes), one can see in Figure 7.16b that the prediction of these fine stop-and-go waves deteriorates quite quickly, as the residuals show systematic underprediction of stop-and-go wave speeds. Although the model cannot capture the specific structure of these stop-and-go waves, the prediction might still be accurate enough for travel times or getting a general overview of traffic conditions.



(a) Speed heatmap for input sequence length = 15 and prediction horizon = 2.



(b) Speed heatmap for input sequence length = 10 and prediction horizon = 5.

Figure 7.16: Sensitivity analysis results on one congestion pattern of best sequence-to-sequence model for dataset 2.

7.3.2. Dataset 3

The results for dataset 3, shown in Table 7.8, look different compared to dataset 2. One sees that a decrease in prediction horizon from 10 to 5 timesteps leads to a large decrease in error, although an increase in prediction horizon to 15 steps leads to a relatively smaller error increase. It is striking to see that for both a 10 and 15 step prediction horizon, an input sequence length of 25 timesteps (50 minutes) leads to a significantly lower error compared to the used input sequence length of 15 timesteps (30 minutes).

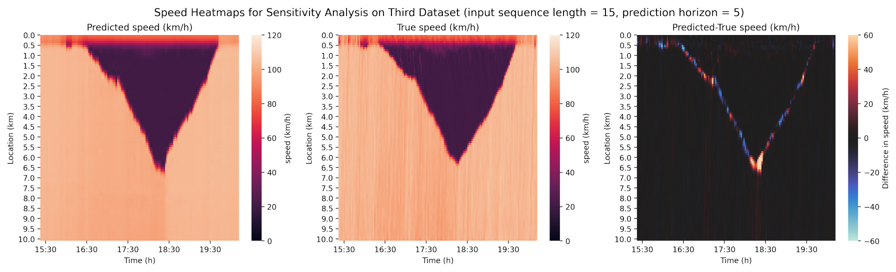
Table 7.8: Test errors of sensitivity analysis best model on dataset 3. Per prediction horizon, the lowest error metric values have been made bold.

	MAPE (%)			RMSE (km/h)			MAE (km/h)		
Prediction Horizon	10*	15*	25*	10*	15*	25*	10*	15*	25*
5 steps	**3.18**	3.18	3.26	4.90	**4.89**	4.89	1.95	**1.95**	1.97
10 steps	5.14	5.36	**4.95**	8.48	8.42	**7.94**	2.74	2.78	**2.65**
15 steps	7.48	7.74	**7.23**	11.46	11.24	**10.46**	3.60	3.57	**3.36**

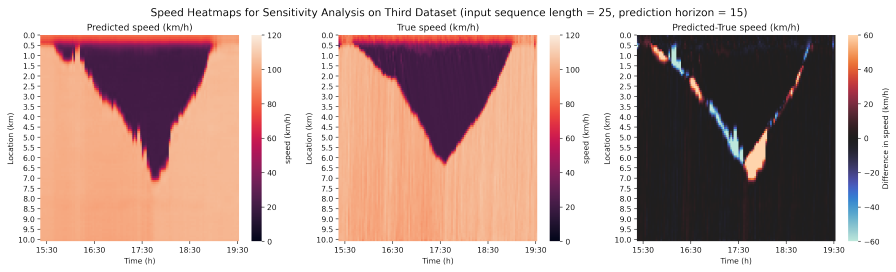
* = input sequence length

For the third dataset, a shorter prediction horizon of 5 steps (10 minutes), combined with an input sequence length of 15 steps (30 minutes), results a lower model error, which is as expected. Looking at the resulting speed heatmaps in Figure 7.17a, one can see the residuals are low and the shock-waves are well predicted. Since there is not much difference between the error metrics when the input sequence length changes, the GRU RNN is capable to extract the most important data from longer sequences.

When the prediction horizon is set at 15 steps (30 minutes), it is remarkable to see the quality of the predictions still is quite high. When looking at the predicted speeds in Figure 7.17b, the shock-wave is predicted quite accurately, and the shape of the shockwave is preserved quite well. However, artefacts can be observed in the data as well as higher residuals near the shockwaves. The model also has difficulties predicting the solution of the congestion, as the residuals are high near the most upstream position of the upstream congestion tail.



(a) Speed heatmap for input sequence length = 15 and prediction horizon = 5.



(b) Speed heatmap for input sequence length = 25 and prediction horizon = 15.

Figure 7.17: Sensitivity analysis of best sequence-to-sequence model for dataset 3.

8
Discussion

After presenting the main findings obtained by analysing the different GRU RNN combinations, this chapter will discuss the results in more detail. This will be done based on the several hypotheses which were formulated regarding the functioning of the models. Besides assessing the validity of the hypotheses, this chapter will also comment on three other findings on 1) the usefulness of the proposed error metrics, 2) the results of the sensitivity analysis and 3) the shortcomings of using simulation data. Furthermore, as using simulation data has significantly influenced the results, comments will be made on the shortcomings of using simulation data.

This chapter is structured as follows. First, the hypotheses will be restated according to their categories and they will be accepted or rejected, based on the results. Secondly, the results of the sensitivity analysis performed on the prediction horizon and input sequence length will be discussed. Thirdly, the proposed error metrics will be evaluated by commenting on the conclusions that can be drawn for them and discussing ways one could use them for other traffic forecasting problems. Lastly, comments will be made on both using simulation data in general as well as specific shortcomings in the datasets.

8.1. Hypothesis Testing

In this section, the hypotheses formulated in subsection 5.1.2 will be accepted and rejected, based on the findings stated in the results. They will be presented according to the same categorisation as has been used in chapter 5. First, hypotheses regarding traffic flow theory are evaluated, focussing on what elements of a congestion pattern the model should be able to predict. Secondly, hypotheses on the neural network architectures will be evaluated. Lastly, hypotheses regarding which input and output data is most useful will be evaluated.

Hypotheses Regarding Traffic Flow Theory

Hypothesis 1: *the model should be able to make traffic predictions while implicitly using input-output relations based on the shockwave theory which can be used to calculate the shockwave speed between different traffic phases, as explained in chapter 2.*

After quantitatively and qualitatively assessing the predicted traffic speed patterns of all datasets, we can conclude that it is possible to accurately predict shockwaves in congestion patterns which can also be deduced based on shockwave theory. However, for the second dataset, the best results were achieved using only speed as input data, although for shockwave theory two of the three fundamental traffic variables, i.e. flow, speed, and density, are needed. Therefore, this hypothesis cannot be accepted for all datasets. Based on the flaws present in the predictions for dataset 1 and 2, it was concluded some prerequisites have to be met regarding to the prediction horizon and the input data. If only the data of a small road stretch was used as input or the prediction horizon has become too high to predict fast propagating shockwaves, it is unlikely the model can correctly predict the propagation of these shockwaves. This statement will now be substantiated using the results from the datasets.

When looking at the prediction results of the several datasets, the accuracy of the predictions differed greatly, from which can be concluded that not only model parameters, but also the road segment is a very important factor on the accuracy of the model. This was found using the spatial error metric in subsection 7.2.3 and the resulting heatmaps which can be found in Appendix B. For the first dataset, the congestion heads could not be tracked accurately, which could be contributed to limitations of the dataset and will be assessed later in this chapter.

Looking at the results of the second dataset, which consisted of both moving synchronised patterns (MSPs) as well as general patterns (GPs) in which both synchronised flow as well as stop-and-go waves are present. For a prediction horizon of 10 steps (20 minutes), both the stop-and-go waves as well as the congestion heads of the MSPs were not clearly displayed in the resulting speed predictions. However, for smaller prediction horizons (of 4 or 10 minutes), the finer structure of the stop-and-go waves and the congestion head of the MSPs were made visible (see Figure 7.16).

The results were best for the third dataset, where patterns were of the type MSP having a stationary downstream jam heads and a moving upstream jam head. On this dataset, especially the sequence-to-sequence model is capable of predicting the propagation of shockwaves between the free-flow and congested phase in an accurate way. For all different shockwave speeds that were tested within the scenarios, the model could predict the propagation of the shockwaves quite well. Therefore, the model did not just generalise to learning or detecting one specific shockwave speed, but is able to accurately make predictions for situations having different shockwave speeds. As the input of the model having the highest accuracy consists of spatiotemporal flow and speed data, it is likely to do so by comparing speed and flow data of both the free-flow as well as the congested phase. However, when the upstream congestion tail remained stationary over time, a correct shockwave speed was hard to predict, resulting in high errors (as shown in Figure 8.1 and Figure 7.15). It is likely this is due to two causes. First, the exact position of the border between free-flow and congestion was unstable, causing the shockwave speed to fluctuate constantly. On the other hand, the model might have difficulties predicting the suddenly decreasing flow causing the shockwave speed to decrease, as little upstream flow segments was present in the dataset.

Hypothesis 2: *it is expected that the model will have a higher accuracy when the congestion patterns consist of synchronised patterns with a single jam head, as for example the moving synchronised pattern, compared to general patterns containing many stop-and-go waves.*

When comparing the error metrics between models trained on the second dataset and models trained on the third dataset, no large difference could be observed. However, assessing prediction results and errors over space and/or time reveals that the predicted congestion patterns of the second dataset, containing general patterns, were of much lower quality compared to the third dataset, containing moving synchronised patterns. Besides, for dataset 2, there are much larger fluctuations in errors compared to dataset, both over space and time as well as when assessing the interquartile range of the MAE. Therefore, the assumption that these stop-and-go waves were harder to predict than synchronised patterns (at least at longer prediction horizons) was true and

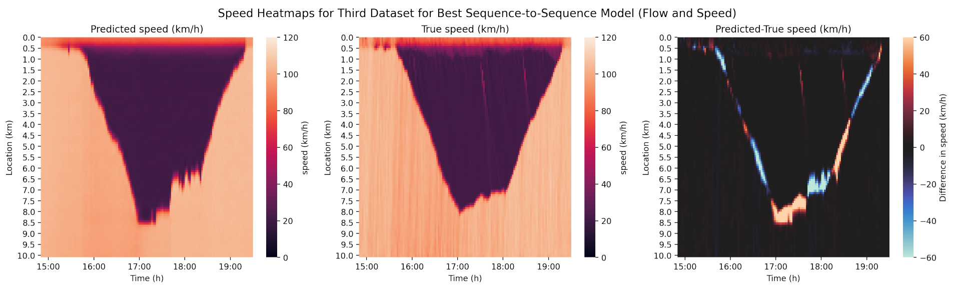


Figure 8.1: Speed heatmap for best sequence-to-sequence model trained on the third dataset, second scenario.

this hypothesis can be accepted. In chapter 2, it was concluded that the emergence of congestion, which is the moment congestion starts to form due to traffic breakdown, is harder to predict compared to the propagation of the congestion heads once traffic has broken down. As these stop-and-go waves emerge more often than a single moving synchronised pattern and propagate upstream quite quickly, it is harder to correctly predict their emergence and upstream propagation given the long prediction horizon.

Hypothesis 3: *it is assumed that there will be large differences in accuracy over space, depending both on the congestion patterns as well as the spatial road characteristics.*

In general, it was observed that the error depends on the complexity of the congestion patterns and whether the causes leading to changes in the shockwave speed where present in the input data, as errors were only lower on sections where no congestion was observed. As the errors did not always increase near the spatial edges of the traffic data, no clear relation between accuracy and edges of the roads was observed. Therefore, this hypothesis could be partially accepted.
When formulating this hypothesis, it was assumed a high prediction error at a point in space could be caused by two factors. First, errors could be higher at locations where congestion is present, since this will lead to higher errors compared to predicting free-flow. Secondly, errors could be higher at the upstream and downstream edges of a road, due to missing upstream or downstream flow and/or speed data the model can base its predictions on. After assessing the error metrics of the predictions over space for the second and third dataset, the errors indeed seem to differ greatly over space. The error is mostly related to the congestion patterns, as it is higher at locations where congestion occurs. However, no increase in error was observed at the upstream and downstream edges of the road, although this could be expected since upstream or downstream traffic data is missing at these sections. Therefore, the congestion patterns in the data influence the accuracy over space most.

After relating the error metrics over space to the road layout and the observed speed patterns (see Figure 7.10 and Figure 7.13), the following observations were made. The error metrics were highest near the lane merge, on- and off-ramp locations. Near the on- and off-ramps, the changing traffic flows and speeds were hard to predict as changes in traffic demands at these discontinuities are not observable. Near the lane merge, this higher error is caused by mispredictions in this breakdown process due to the stochasticity in the traffic breakdown process. For the third dataset, some peaks in error were also noticed at the maximum propagation of the upstream congestion heads in the different simulations, because of difficulties with predicting the new correct shockwave speed. This partially can also be caused by the aggressive changes in traffic demand within the simulations, as traffic demand decreases quite abruptly when the upstream congestion head should remain stationary or should start to propagate downstream for the solution.

Hypotheses Regarding Neural Network Architecture

Hypothesis 4: *the sequence-to-sequence GRU RNN is expected to perform better than the one-shot model for traffic speed prediction.*

From Table 7.6 could be concluded that the sequence-to-sequence model structure leads to better predictions than the one-shot model for both datasets, but the difference was especially big for the third dataset containing only GPs. In Figure 7.7, it was shown that the overall shape of the resulting predictions was of much better quality for the sequence-to-sequence model, as the resulting shockwaves were more continuous and smooth compared to the one-shot model. Therefore, this hypothesis can be accepted.

Several reasons might make a sequence-to-sequence model beneficial over a one-shot model. As the sequence-to-sequence model predicts a sequence of values instead of a single value in the future (see chapter 3), it can break down the prediction process in steps of 2 minutes, using a previous prediction when predicting the next sequence value [6, 8]. As its loss function is not based on one predicted value, but rather over the whole predicted sequence, this can lead to more accurate training [6, 8]. The propagation of errors over the prediction sequence, which can be a disadvantage of autoregressive models over one-shot models, seems to not be a problem here. Since the prediction process is quite continuous and the shockwave speed is almost constant over a longer time period, this particular prediction type could be very suitable for a sequence-to-sequence model. Especially on the third dataset, the sequence-to-sequence model outperformed the one-shot model, since the third dataset contained localised synchronised patterns with a clear shockwave speed compared to the many stop-and-go waves present in the second dataset. For the second dataset, the difference was less big and qualitatively, not much improvement could be observed from the sequence-to-sequence prediction heatmaps over the one-shot prediction heatmaps.

Hypothesis 5: *the loss function will influence the prediction accuracy, as sharp fluctuations in speed between the traffic phases differ from a regular regression problem.*

When analysing the results of the neural networks based on the used loss function, great differences in model error were observed for the first and third dataset when using a one-shot model. In case of the first dataset, using a MSE led to low RMSE values whereas using a MAE led to low MAE and MAPE values. However, for the third dataset, a MSE loss function resulted in consistently large errors for all model combinations.

For the first dataset, the prediction accuracy is quite low, both when using a MSE and a MAE loss function. However, one could observe that when using MSE loss, speeds were systematically under-predicted during congestion. This large difference results from their definition, as given in Equation 3.2 (MSE) and Equation 3.3 (MAE) in subsection 3.1.1. The MSE penalises large differences between prediction and ground truth higher. These are especially present at the phase transition positions where the upstream congestion tail moves upstream or downstream, since the model cannot accurately predict the emergence and propagation of this jam wave. By overestimating the speeds during congestion, the MSE loss values and RMSE values will be lower when the jam wave is solving, leading to a better model prediction according to the MSE loss function. Since the MAE loss function does not penalise large prediction errors more compared to smaller errors, as it is a linear loss function, this model will not suffer from the same artefacts.

For the third dataset, one can see that the predictions are much worse using a MSE loss function (Figure 7.4b), compared to the MAE loss function (Figure 7.4a). However, in contrast to the first dataset, the congestion patterns predicted using the MAE loss were quite accurate, leading to an even higher difference in error comparing both loss functions. When using a MSE loss function, there is no sharp border between the two traffic phases as speed values are gradually shifting around the shockwave separating the two speeds. Although there are differences between the datasets, the problem seems to be similar to the first dataset. First, since the MSE penalises large errors more than smaller ones, a wrong prediction of this shockwave will lead to large errors as the difference in speed between the phases is quite large. As the precise location of this shockwave is uncertain, having gradually increasing speeds near the shockwave prevent MSE values from becoming too high as speeds around the exact shockwave location are similar to this speed difference. Apparently, the sum of having a relatively smaller error on both the free-flow and congestion phase is smaller than mispredicting the shockwave. However, since the resulting errors of the trained model are large, the model seems to be getting stuck in a local minimum during the training process as a result from this approach.

Based on the findings on both datasets, it seems as if the model results are much more stable when using a MAE loss function compared to a MSE loss function. This might be because traffic speed prediction is different compared to regular regression problems (and even traffic flow prediction) since speeds can differ greatly between traffic phases and can change rather abruptly when a congestion pattern occurs. In case a certain congestion pattern can not be predicted well, neural network training is more likely to get stuck in a local minimum when using a MSE loss function. From the results it seems the MAE loss function is better able to capture sharp changes in speed between the two traffic phases than the MSE loss function and in general leads to more stable results. This suggests that using a MAE loss function is preferable over a MSE loss function.

Hypotheses Regarding Input & Output Data

Hypothesis 6: *having both flow and speed data as model input will improve the accuracy of the speed predictions compared to only using speed.*

As for the second dataset, results were more accurate when using speed data only, this hypothesis is rejected. Based on traffic flow theory, it makes sense to both use flow and speed data, this strongly depends on the chosen model and the traffic flow and speed data of the specific road. The combination of model limitations and the complexity of the input data can cause the results to vary greatly. It is therefore advised to make the decision to include flow data per situation, by testing both options when implementing a new neural network.

Based on the test results in Table 7.6, it is unclear whether adding flow data always leads to a higher accuracy compared to having only speed data. On the simple congestion patterns from the lane merge bottleneck of dataset 3, adding flow data clearly leads to better predictions. The resulting predicted speed heatmaps are capturing the congestion patterns well, as the shockwave speeds are well estimated. However, for the more complex general patterns in dataset 2, adding flow data actually leads to an increase in prediction error on the sequence-to-sequence model. Although for one-shot models, adding flow data always resulted in lower errors, this is not the case for sequence-to-sequence models on dataset 2.

It is likely that this can be explained by the differences in the generated output of both the sequence-to-sequence and the one-shot model. A downside of using sequence-to-sequence models is that due to its autoregressive nature, the model output always has to be equal to the model input, as the prediction of the previous time step is used as input for predicting the next time step. Therefore, the sequence-to-sequence model also has to correctly predict the traffic flows since during training, the loss function is evaluated on both flow and speed data. When comparing the ground truth flow and speed patterns of the scenarios of the second (Figure 6.5) and third dataset (Figure 6.8), one can observe that both the flow as well as the speed patterns of the second dataset are much more complex than the third dataset. Since the loss function of the model is optimised on both depends on the accuracy on flow and speed predictions, correctly predicting traffic flows might come at the expense of the accuracy of the speed predictions.

Hypothesis 7: *by combining the flow and speed data per road segment and per time interval, the model accuracy will increase.*

In subsection 5.3.2, two input layer structures have been proposed combining flow and speed data per section and time interval, which should lead to an equal or better accuracy compared to not combining the data. These input layer structures have been tested on the first dataset, but the results show that they both lead to an increase in model errors compared to a regular fully-connected approach. Since the error increase was quite significant (MAE: +0.4 km/h, RMSE: +0.7 km/h, MAPE + 1 %pnt), the hypothesis was rejected and has not been tested on other datasets.

Although combining the flow and speed data per segment results in less trainable parameters of the model (see Table 7.1), this might not be advantageous if ample training data is available since the model can ultimately fit the correct parameters itself. If enough data is available, one can argue that the added flexibility of the fully connected input layer might even result in a higher accuracy, since the neural network can then fit individual weights for each flow and speed value for each road segment instead of having to do so for each combined flow/speed variable. This is especially advantageous in case flows and speeds have different spatial correlations. Hence, combining flow and speed into one variable might be too restrictive and can therefore decrease the model accuracy.

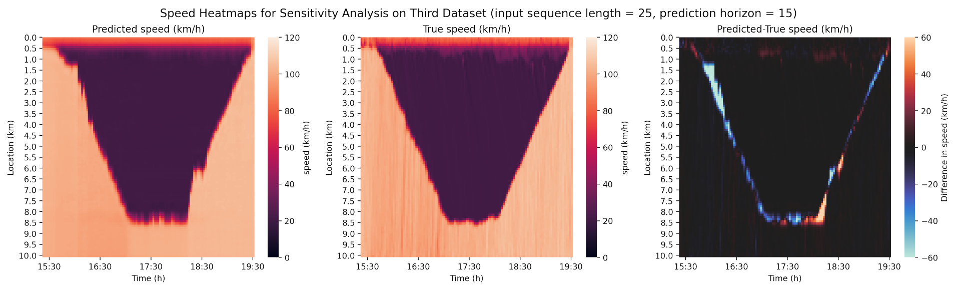
8.2. Sensitivity Analysis of Temporal Horizon

From the results of the sensitivity analysis, some surprising conclusions can be drawn. For the second dataset, a decrease in the prediction horizon led to a great decrease in the resulting prediction error. The resulting prediction heatmaps were also able to capture the complex stop-and-go patterns emerging from the bottleneck. This concludes that the model can capture these stop-and-go waves, given the input data and prediction horizon is well specified. For the third dataset, it was found that a higher prediction horizon of 15 steps resulted in an increase in errors but the resulting heatmaps show the results are still accurate.

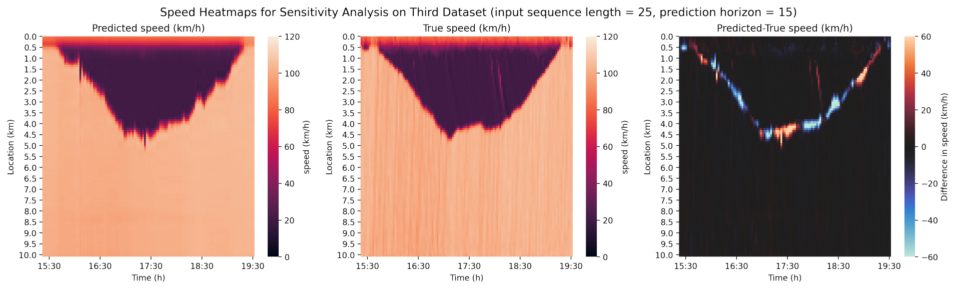
However, one could also see that for both datasets, an input sequence length of 25 steps (50 minutes) resulted in the lowest errors given the standard prediction horizon of 10 steps (20 minutes). This would mean that flow and speed data from up to 70 minutes before the actual prediction contribute to a higher prediction accuracy. However, from a traffic point of view, it is unlikely that there is a causality between flow and speed data over a time span of 70 minutes over a 10 kilometre long road section. As traffic will a maximum travel time of 20 minutes during heavy congestion, this will provide limited information on the future traffic flow.

As it is unlikely the added data will give more insight on the traffic conditions, some kind of overfitting could occur on the neural network when having too large prediction horizons, especially in case of the used simulation data. As there is no traffic flow theory related correlation over such long temporal horizons, the model might fit on trends present in the demand pattern of a road instead of it learning general traffic characteristics. Since there are only three to four unique demand patterns per dataset, many simulation replications in each dataset are quite similar. Although AIMSUN introduces stochasticity to the simulation variables for each replication within a scenario, this mostly result in small differences between the replications within a scenario. However, a longer input sequence length might increase the amount of unique sequences available in the dataset, which might incentivise fitting on previously observed traffic conditions instead of learning general traffic patterns. Having too high input sequence length might therefore decrease the model generalisability.

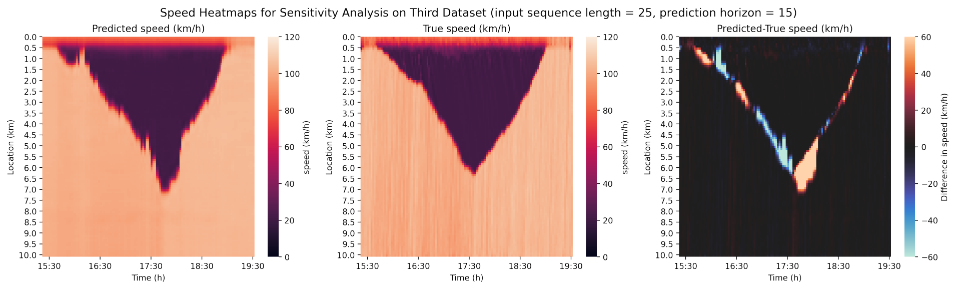
Signs of overfitting can actually be observed analysing the resulting heatmaps of the best model belonging to the third dataset, where the prediction horizon was 15 steps (30 minutes) and the input sequence length 25 steps (50 minutes). It can be seen that, almost by default, the predictions between 15:40 and 16:10 look exactly the same, as in all speed heatmaps a small congestion wave occurs at that time. Although the neural network has no input related to time, the longer input sequences might introduce some uniqueness to the inputs which can still cause overfitting to happen. Besides, it does not correctly predict the propagation of the shockwave when congestion starts to solve in Figure 8.2c, as it expects it to propagate further but correctly predicts the solution of congestion in Figure 8.2a and Figure 8.2b. From this might be concluded that the dataset has overfitted on Figure 8.2a and Figure 8.2b too, at the expense of a misprediction of a traffic breakdown in Figure 8.2c.



(a) Speed heatmaps first scenario.



(b) Speed heatmaps second scenario.



(c) Speed heatmaps third scenario.

Figure 8.2: Speed heatmaps of sensitivity analysis best sequence-to-sequence model trained on dataset 3 (input sequence length = 25 steps, prediction horizon = 15 steps).

8.3. Evaluation of the Error Metrics

In subsection 7.2.3, the results of all custom error metrics proposed in subsection 5.6.4 were presented. Presenting the errors in this manner lead to valuable insights which could be extracted in a more straight-forward way, compared to qualitatively assessing the resulting speed heatmaps. In this section, their usefulness will be discussed further, including guidelines on the interpretation if they would be used on other datasets.

In general, the RMSE and MAE were found to be the most relevant error metrics. As the MAPE weighs overestimations ($\hat{v} > v$) significantly more than underestimations ($\hat{v} < v$), it has to be interpreted with caution making it less suitable as an error metric. The results of both the RMSE and the MAE were quite similar, with the most important difference being the scale of the error metrics. The RMSE value will increase more than the MAE value when the magnitude of the residuals increases. The results section shows that when comparing different models, RMSE and MAE values can deviate quite strongly between the models, making it relevant to assess both. However, when assessing the accuracy of one single model over space and time, using only one will suffice. In this case, the MAE is preferred since the interquartile range (IQR) provides information on the distribution of the error over different days.

Assessing the error over space can aid with the interpretation of model predictions as it gives potential users insight in the locations where erroneous predictions are more likely to occur. By also marking the locations of on- and off-ramps, higher error values can be directly related to bottleneck locations. The same holds for the correlation over time, as it makes it easy to identify at what times potential fluctuations in traffic demand cannot be predicted by the model. After analysing these aggregated spatial or temporal graphs, the spatiotemporal heatmaps can be used to relate spatial and temporal peaks to each other. It can also be an intuitive way to interpret the model accuracy for traffic engineers, as its formatting is similar to speed heatmaps used to interpret congestion patterns. In case more information is necessary on the model accuracy in specific situations, comparing the speed heatmaps of the prediction and ground truth can also be a valuable tool.

The confidence intervals of the MAE show at which locations the error metrics can fluctuate strongly. When comparing the IQR over the different simulation iterations with the mean values of the MAE, one can see that the MAE values are strongly influenced by outliers. However, these outliers are probably important in predictions, as these consist of the deviant congestion patterns that disturb traffic most. The IQR will especially be large if there is a wide variety of congestion patterns present in the dataset and the model is not able to predict all different congestion patterns with the same accuracy. Therefore, this IQR can be seen as a measure of *stability*, giving insight in how well the model performs on all the different congestion patterns present in the dataset. Comparing the stability of the model accuracy by assessing the interquartile range of different models can therefore be very helpful to assess the model accuracy. It might be more beneficial to have a model where the MAE distribution over different days or congestion patterns is much narrower, although the overall MAE might be higher.

Although these error metrics work quite well with the simulation dataset, it is unclear what they will look like using empirical loop detector data, as this data source has much more variability in congestion patterns. Traffic flow and speed data of a road over different days are likely to have some similarity, as bottlenecks are often constant and morning and evening peak hours start around the same time. However, traffic demand in the simulation data is much more similar as traffic flows increase and decrease (mostly) at the same time for each dataset. Especially the spatiotemporal heatmaps can become quite noisy when a lot of different traffic congestion patterns are present in the dataset. Therefore, it is important to assess this shortcoming when these metrics are tested on other empirical data.

8.4. Simulation Data Quality

As the results and the conclusions on the previously discussed hypotheses highlight that the datasets had a big influence on the results of the neural networks, this chapter will briefly discuss the shortcomings of each dataset individually, as well as the general shortcomings of using simulation data and the implications on the retrieved results. First, the shortcomings of each dataset are discussed. Secondly, the shortcomings found in the simulation data will be assessed. Lastly, some artefacts present in the microscopic simulation software AIMSUN will be discussed.

8.4.1. First Dataset

The first dataset consisted of a 2 km long road with a lane merge bottleneck, resulting in the occurence of a moving synchronised pattern (MSP). The errors of the models trained on the first dataset are significantly higher compared to the second and third dataset. In other words, it is expected that the errors are partly caused by the simulation dataset used to train and assess this model. The shortcomings of the first dataset were as follows. First, the road length was quite small, as the total road length was only two kilometres of which 1.5 km was equipped with detectors. In combination with a high traffic demand, the congestion propagated to the most upstream road section in around 20-30 minutes. Together with a large prediction horizon, this resulted in too little 'information' present in the dataset to base the prediction on. This can be explained by the following scenario. When training the model, there should be both a clear relation between a change in the input data and a change in the output data so a clear input-output function can be formed. However, as soon as there was one input sequence value which had some lower speeds or higher flows compared to free-flow, the related output data already showed that in 20 minutes from the last sequence value, traffic speeds had to drop for all road segments in the data.

8.4.2. Second Dataset

The second dataset consisted of a 10.7 km long road stretch containing many on- and off-ramps, resulting in a mixture of MSPs and a general pattern (GP) containing stop-and-go waves. For this dataset, some shortcomings were also noted. First, the complex traffic pattern of the second scenario, containing many stop-and-go waves was proven to be hard to predict by the neural network. Since the traffic breakdown process leads to the random formation of stop-and-go waves, it is almost impossible to predict the occurrence of individual waves. As a single stop-and-go wave would dissolve in around 20 minutes, it is not feasible for the model to predict the propagation individual stop-and-go waves over the road segment. Therefore, the resulting speed heatmaps did not capture these stop-and-go waves, resulting in a low prediction quality. For the scenario which mostly consisted of MSPs, the jam head propagated over only 6 km of the simulated road the increase in traffic flow during congestion was large. This resulted in a high shockwave speed, as the upstream congestion head reached the upstream border of the road within half an hour. As this is approximately equal to the prediction horizon, the resulting predictions did not clearly predict the correct propagation of this jam head.

8.4.3. Third Dataset

The third dataset consists of a 10 km long road with only a lane merge at the end of the road. The resulting MSPs have some more variety as scenarios had different shockwave speeds. The main idea behind introducing the third dataset was to check whether the neural network predictions could correctly predict congestion patterns having different or fluctuating shockwaves. Although it succeeded on that part, the used demand patterns and the resulting congestion patterns were quite artificial. When using empirical data, changes in traffic flow due to a different traffic demand will happen more gradually compared to the simulation data. This will lead to more fluctuating shock wave speeds and less abrupt changes in the shockwave speed. Besides, there will be far more discontinuities such as on-ramps and off-ramps present on actual highways, making a 10 kilometre road which only has one lane merge an unrealistic highway scenario.

8.4.4. Limitations of Simulation Data

Besides commenting on the limitations per dataset, due to the road layout or the demand patterns used, using simulation data in general for training neural networks has some shortcomings. Although this could not directly be observed in the results, it is expected that using simulation data greatly impacted the results obtained in this thesis as the variation in the congestion patterns present in each simulated dataset is much lower compared to actual traffic data. Since the replications of each scenario within a dataset all contained the same traffic demand pattern and the number of scenarios per dataset was limited, only little difference was found in the congestion patterns per dataset.

When using actual traffic data, there are much more different traffic patterns present compared to the simulations as several factors were not captured by the datasets. First, traffic demand can differ greatly between days, as events, holidays or extreme weather conditions can lead to congestion patterns which differ greatly from the average weekday congestion pattern. Secondly, traffic accidents can cause large disruptions on a road and are highly unpredictable, as the occurrence of traffic accidents cannot be predicted based on flow and speed data. Thirdly, as traffic demand in general grows over the years and traffic flows increase, the historical congestion patterns on which a model is trained can be quite different compared to the traffic flows and speeds it will use as input when it is applied. Therefore, to correctly assess the results of a neural network, the dataset should not be shuffled over time. It should be tested on the most recent spatiotemporal traffic data and trained on data from before that period, to see whether the model results are generalisable over time.

Due to the aforementioned factors, the accuracy of the model used on empirical data will most likely be lower compared to the simulation datasets. Secondly, because there is limited variation in the congestion patterns in the simulation dataset, a high risk in overfitting exists for the neural network. The learning rates in Figure 7.6, presented in section 7.2, already show that the validation loss is quite similar tot he training loss. When comparing the validation errors to the test errors, one can also see the difference is quite minimal. As the training, validation and test errors are quite similar, one can conclude that the three datasets are quite similar too making it not possible to assess whether the model is generalisable to different datasets. This increases the probability of the model being overfitted on the dataset.

What could also be observed when comparing the flow and speed of the loop detector data of the lane merge at A20 (in Figure 4.5 and Figure 4.4) with data from the simulated lane merge bottleneck (in Figure 6.8) was that the resulting flow and speed patterns differ greatly. Within the simulation, a clear capacity drop could be observed and speeds within the congested phase were largely stable. For the empirical data, this capacity drop could not be observed and traffic flows and speeds could fluctuate greatly over time.

For the lane merge simulation in the first and third simulation dataset, traffic speeds were rather homogeneous within the congested region. However, when looking at the speed heatmaps in chapter 4, one could see a fine pattern of stop-and-go waves (or narrow moving jams) within a synchronised flow area. Although traffic situations were quite similar, as the lane merge was the predominant source of congestion in the empirical dataset, this seems like a large difference in the resulting congestion patterns.

When the flow and speed data between the simulation and empirical loop detector data are compared qualitatively, one could see that the data quality of the simulation data was much higher compared to the empirical loop detector data. In the simulation, the resolution of the flow and speed data is quite high, having clear changes in flow near on- and off-ramps and rather abrupt changes in speed near the edges of the congested road section. This is not the case for the actual loop detector data, as flow as well as speed data changes happen rather gradually and large flow differences are observed although no on- or off-ramp was present.

Although it is outside the scope of this thesis to thoroughly assess the shortcomings of microscopic simulations or the data quality of loop detector data, using this data likely poses some limitations on the resulting flow and speed data.

8.4.5. Artefacts in Simulation Data

Some artefacts could also be observed. As was explained earlier in this chapter, the cooperation and aggressiveness values were change to obtain more realistic results on a macroscopic level. However, in practice, this lead to frequent vehicle collisions which lead to situations as shown in Figure 8.3, where situations are encircled in red where two vehicles have a negative headway (i.e. are positioned on top of each other). Although it is unlikely these situations will have had much impact on the retrieved flow and speed data, it is good to notice that density data from this model is likely to be invalid, especially on the upstream border between free-flow and congestion.

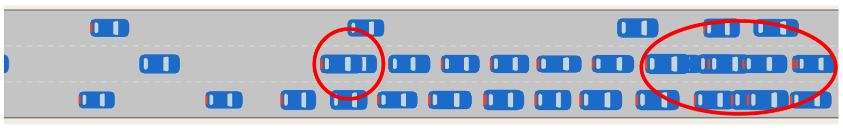


Figure 8.3: Example of simulation artefacts in AIMSUN.

9
Towards Spatially Flexible and Generalisable Traffic Forecasting Neural Networks

In the introduction, it was mentioned that there are some practical limitations on implementing neural networks for traffic forecasting. The neural networks used for traffic flow prediction have been trained on only one specific road, which also means that their hyperparameter settings and model structure have been optimised on one road only. Some researchers already tried to capture whether a machine learning traffic forecasting method can be used for traffic situations which are different from the ones it has been trained on. Van Lint and Van Hinsbergen [12], Oh *et al.* [15], Do *et al.* [44] mentioned 'location-specific' or 'site-specific' neural networks, Smith and Demetsky [111] are talking about 'model portability', as they 'develop' a model on one road and see if it performs comparably on different sites. Karlaftis and Vlahogianni [10] are talking about transferability of results when mentioning over-specified models having no generalisation power.

As one currently has to create a new neural network for each new road, based on a previously design neural network architecture, implementing a neural network poses couple of downsides, namely:

- it requires a lot of data of the new road to train the neural network on: although in general (and for the Netherlands) loop detector data is available and floating car data quality has increased tremendously, checking the data quality and preprocessing the data can pose a significant amount of work if this does not happen automatically; and
- extensive hyperparameter tuning might be required to reach the desired level of accuracy of the neural network for the specific road: depending on the computational effort of the neural network configuration used, this might require a lot of (computational) resources and time to train and assess all model variants.

A solution for the problems stated above might be to make traffic forecasting neural networks both more **generalisable** as well as **spatially flexible**. This chapter is used to explore both concepts and to identify which solution strategies can be used.

9.1. Definitions
This section will start of with some definitions which will be used throughout this chapter.

9.1.1. Model Generalisability
There are some concerns that training neural networks on data from one road only leads to overfitted models with little generalisation power [15]. As two roads can be very different, one cannot just use the same neural network which was trained on one road for traffic forecasting on

different roads. Due to differences in road layouts and different traffic demands, different types of congestion patterns are likely to occur. This can then lead to differences in model accuracy over different roads. Another side effect is that neural networks which are not generalisable are also likely to suffer from **concept drift** [11, 43], meaning that as time progresses, the data on which the model has been trained might not be representative for the current situation [112]. In the context of traffic, the traffic patterns on a road might change over time, due to (for example) differences in (i) traffic demand, (ii) driver behaviour, (iii) vehicle characteristics, (iv) road design or (v) policy and legislation [18]. This would mean that the neural network has to be retrained continuously for the accuracy to remain high. Therefore, the following term is introduced:

Congestion pattern generalisability: the extent to which a model will have a comparable accuracy on traffic patterns of either different categories or having different properties within a traffic pattern.

A model having a high generalisability may have the following advantages, as (i) it can indicate the model has learnt underlying traffic characteristics instead of location specific patterns in the traffic flow data and (ii) it can indicate the model is more robust to location specific changes in traffic flow.

9.1.2. Spatial Flexibility

A practical problem of using the same neural network for different roads is the incompatibility of the input and output layer sizes. Often, these depend on the characteristics of the data from the road it was designed on, which depends both on the number of segments the road was subdivided in and the length of a segment. When traffic data is retrieved from roads having different sizes, the dimensionality of this data can change due to a different number of segments and segment spacing. This can make it impossible to use the same neural network on different roads. Besides, from chapter 2 could be concluded that the resulting congestion pattern strongly depends on the road layout and positions of on- and off-ramps. Not accounting for both of them can result wrong predictions, as the model was not able to distinguish between the different (implicit) spatial dimensions of the datasets. Therefore, the following definition is proposed:

Spatial flexibility: the ability of a traffic forecasting neural network structure to accurately predict traffic variables on roads consisting of a different amount of segments and/or different segment lengths.

9.2. Traffic Predictability by Neural Networks

The quality of a prediction in general depends on two different factors. On the one hand, the model or method used for the prediction is important, as its strengths and weaknesses have implications on the quality of the prediction. On the other hand, this also depends on the specific prediction problem as some processes are easier to predict than others. Therefore, the term *predictability* is coined, which is defined as the ability of a system's state to be predicted. This section will further explore both concepts in the context of traffic prediction by neural networks.

In chapter 2, an overview was given of the aspects which are proven to be predictable, when looking at traffic flow theory. The conclusion was that predicting congestion can be broken down in predicting two aspects, namely the traffic breakdown process, or the emergence of congestion, and the propagation of a congestion pattern.

Predicting the exact time at which traffic breaks down is hard if not impossible to predict, as this is a stochastic process, influenced by many parameters. Although the breakdown process can be defined as a situation where supply, governed by the road capacity, is larger than traffic demand. As there is no fixed road capacity, it is unknown at which traffic flow a breakdown will be initiated. Brilon *et al.* [26] expects the road capacity depends on road geometry, control conditions, driver/vehicle populations and even travel purposes. van Lint [27] also suggests factors as ambient conditions (weather, road visibility), road works and traffic collisions. On the other hand, traffic demand is influenced by effects like traffic information, network effects (influenced by the traffic state of unobserved roads) and traffic demand in general [27].

A problem of some of these parameters is that they cannot be captured quantitatively, or are hard to predict. Variables which are hard to capture quantitatively are driver/vehicle populations traffic information, network effects (outside the considered highway), road works and road geometry. Although they can be observed, it is hard to to use them as neural network inputs. Variables which are not predictable are the actual traffic demand and the exact time and location a traffic collision will occur. Therefore, the model is expected to perform worse if these variables change significantly, as they cannot be accounted for in the input space.

When a traffic forecasting neural network has to be made generalisable and spatially flexible, many of these properties have to be explicitly passed in the input layer. All current traffic forecasting neural networks can implicitly learn these road-specific variables, since they are fitted on only one road. If a neural network should be applicable to several roads, this is not possible anymore and these parameters need to be explicitly passed to the neural network. This can pose a challenge when creating spatially flexibile and generalisable neural networks.

Let us present an example of the implications of this missing demand data, highlighting one of the most important unobservable parameters, namely traffic demand. In dataset 3, there were no traffic access and egress points besides the upstream and downstream edges of the road. As a higher traffic inflow on the upstream edge of the road would eventually reach the bottleneck causing congestion, the prediction process would be fairly easy. However, on actual roads, traffic flow near on- and off-ramps can vary quickly and cannot be predicted, since the future traffic demand of these on- and off-ramps is hard to observe. For example, a sudden increase in traffic demand on a on-ramp can cause a traffic breakdown which is unlikely to be predicted by the model. Since it is virtually impossible to predict short-term traffic demand, it is unlikely any model can accurately predict this situation.

The propagation of a congestion pattern was found to be predictable when at least two of the three macroscopic traffic variables are known. However, a prerequisite would be that either these traffic variables remain constant over space and time or are predictable. If, again, unobservable or unpredictable parameters change, the model cannot capture the effects of this. For example, if during congestion traffic demand changes or a traffic condition occurs, it is unlikely that these effects can be predicted. However, once they are observed and the traffic situation is rather stable, it will be able to predict the onset of these changes.

9.3. Scale-Free Approaches

In this section, two different approaches are proposed in order to make traffic forecasting neural networks scale-free. The first one, segmental neural networks, relies on training a neural network on a smaller road section, as these show more similarity compared to regular roads. By combining these smaller sections according to the layout and properties of that road, a larger neural network can be formed. This means that traffic can be predicted for different roads using the same building blocks. The second one is based on the term 'neuroscopic traffic model', coined by Hans van Lint. This approach relies on a core neural network which has learnt the general concept of traffic prediction, and can therefore be used on a range of roads with only minor changes.

9.3.1. Segmental Neural Networks

It could also be possible to not just have one neural network which predicts traffic speeds on one road, but to have a large variety of neural networks which can all predict traffic on a specific type of segment of a road. Although the road layout of two highways can differ quite a lot, individual segments of different roads can be quite similar. Two different three-lane segments, curves with a similar radius or on- and off-ramps with a similar split ratio (i.e. in-/outflow as function of the main traffic flow) will have similar capacities and will therefore have a similar 'effect' on the resulting flow

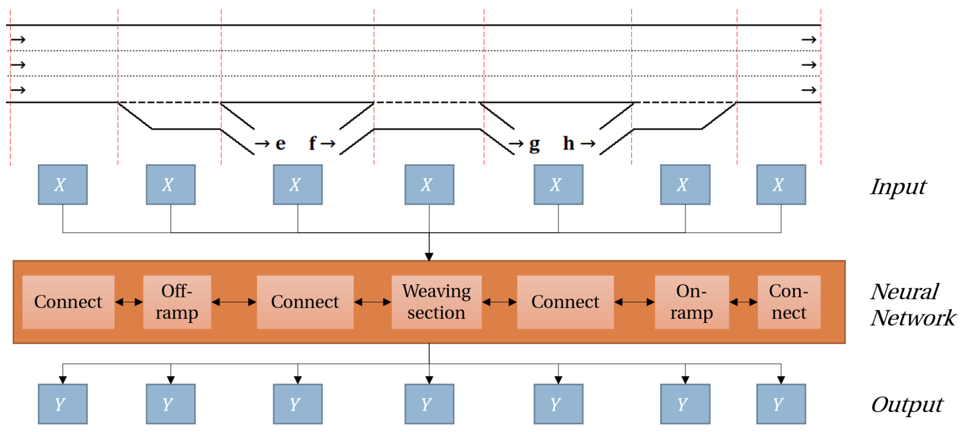


Figure 9.1: Schematic representation of a segmental neural network. The inputs belonging to each segment are depicted with X, whereas the outputs are shown by Y. Within the neural network, each road segment has a specific element corresponding to the characteristics of this section. These elements are connected so information can be shared.

and speed patterns. By creating a database of trained neural networks, each representing a specific segment, each road can be simulated by combining these 'segmental' neural networks with each other. In this case, limited extra variables have to be explicitly used as input for the neural network.

This would be similar to a (macroscopic) cell transmission model, where traffic on each link is modelled as well as the interactions with other links. In this model, one specifies traffic demands and supply based on the link characteristics (such as the flow,density relation from the fundamental diagram), leading to a model of the traffic flows on all links. Each time step, traffic flow and density in a cell is updated based on the supply of the upstream cell and demand of the downstream cell. Let us start with an example when upstream traffic conditions are relevant. If high traffic flows are observed or predicted on a segment *upstream* of a bottleneck, this can cause traffic flow to exceed capacity when these vehicles reach the bottleneck location and can therefore cause traffic breakdown. However *downstream* segments can also be of relevance as due to spillback, as the downstream congestion head can eventually reach more downstream segments.

When predicting traffic using a neural network, using only flow and speed data, supply and demand are not observable. Only the product of the two, which is traffic flow, is given as input. The neural network is expected to implicitly learn a representation of this process, by trying to fit its parameters so from a set of input values the correct output values can be found. The reason that these neural networks are not generalisable is that this supply and demand relation is unique for each road and as this relation is not captured in the input variables, the same neural network cannot be used on other roads.

Although this supply and demand relation is different for each road, it can be quite similar for individual segments within a road. The idea of these segmental neural networks is that we can have this large database of segmental neural networks, which are very well trained on a specific traffic situation, and combine them in a plug-and-play fashion. To forecast traffic an arbitrary road, one just has to combine segments of this database which are most similar to the segments of the simulated road to have accurate predictions. One does not have to train a new neural network from scratch in order to do this.

Challenges

In case we want this segmental neural network to predict flow and speed on a segment in a way similar to a regular (autoregressive recurrent) neural network, the model should be able to correlate the following variables or processes:

1. recently observed flow and speed of the assessed segment,
2. recently observed flow and speed of neighbouring segments,
3. the predicted flow and speed of the assessed segment.

4. predicted flow and speed of neighbouring segments,

5. weights and biases which correlate the observed/predicted flows and speeds of other segments to the assessed segment.

Although 1 to 4 can be relatively simple, the fifth process is tricky. One problem can be observed here. In a regular short-term traffic forecasting RNN, the segments located around the considered segment are always the same and the model weights determined during training take this into account. However, for this segmental RNN, this is not the case as segments can be coupled arbitrarily. As each segmental neural network is trained separately and should be used in a plug-and-play setting, it did not learn which weights would represent this particular combination of segments.

Essentially, this interaction between road segments in combination with the restrictions present in the observed data means that the future traffic conditions of a segment also depends on the specific combination of segments around this segment. However, since the idea is that these segment-specific properties are captured by the neural network of that specific network, it is not easy to share this 'information' with the other segments.

9.3.2. 'Neuroscopic' Core Traffic Model

Another possibility would be to reuse the same (core) neural network to predict traffic on several roads. This poses two challenges for which different methods can be used. First, it would be ideal if one can *reuse* a neural network which has learnt the general concept of traffic flow prediction and can be reused roads with different congestion patterns. Secondly, the model structure should allow a similar neural network to predict traffic on roads where the input and output data has a different *dimensionality*, for example when the number of road segments differs. Thirdly, it should be able to differentiate between roads, as different road layouts can lead to different congestion patterns.

In order to reuse a model on different roads *transfer learning* can play a role. Using transfer learning, instead of training a blank model to perform a particular task, one uses an already trained model on a similar task as a starting point. This can influence performance in three ways compared to a regular neural network: (i) the out-of-the-box performance of the model is higher, (ii) it trains faster, needing less training time to reach a higher performance or (iii) after extensive training, it performs better [6].

It is unlikely that it is possible to reuse exactly the same core neural network on all prediction tasks. If this was the case, it would be much easier to just take a current macroscopic model and use that for traffic forecasting. A quality of neural networks is that they can be fitted to the input data to find case-specific correlations. Within transfer learning, it is also possible to further train a pre-trained structure, for example by allowing the neural network weights to change within a certain bandwidth [6]. This might also lead to a better generalisable model, since it does not have to learn to distinct different highways based on the input parameters. By partially fitting this core neural network to one road, it can learn road-specific properties by using relations in the road-specific input-output data, deducing these relations from the data instead of specifying them in the input data.

In order for the model to handle different input dimensionalities, *representation learning*, which actually is a way to automatically extract features from large datasets containing raw data [39]. Many representation learning methods rely on dimensionality reduction, such as principal component analysis which finds the eigenvectors, along which the maximum variance in the data is found [30]. Essentially, deep neural networks also perform some form of representation learning, as the added complexity in the network can be used learn complex relations in large datasets. Within traffic flow forecasting, one could even see the flow and speed data inputs as raw data of which features need to be extracted.

There are several ways this (flexible) dimensionality reduction could be done, which will now be presented.

RNN Sequences as Spatial Dimension
When using a recurrent neural network, one could 'transpose' the spatiotemporal matrix: each input unit consists of a time interval and the sequence belonging to an input contains the data of all segments at that time. Similar to language processing, one can use padding or 'end of sequence' markers to account for these variable sequence lengths.

However, using this method might have some downsides. This can result in very long sequence lengths, as even the number of segments on the simulated highways was around 100 nodes. It makes sense to apply the same transformation to each timepoint per segment, if we assume traffic is stationary. Most of the times this is the case (see section 2.1. However, in other case, it has to apply the same transformation to each segment per timepoint, which would assume traffic is homogeneous. This definitely is not the case, as traffic conditions change over space. Therefore, this variant is disregarded.

Padding
Secondly, one could make use of padding [8] without having to transpose the input data. In this case, the number of input units would be set to the maximum number of road segments the neural network would be able to process. In case the number of segments of the input data is lower than the input dimension of the neural network, one can simply fill the input data of the non-used input units segments with zeros (or other values). Although the term padding is general is used within a sequence context, it can also be used in this way.

However, this rather simplistic solution might not work well in practice. First of all, this will only work on different highways which have a very high traffic pattern similarity, as the data of each road is directly given to this generalisable core neural network. As has been noted in the traffic pattern similarity definition, highways can differ significantly over space in ways which is not encoded in the flow and speed data. This can lead to very different traffic patterns. In order for this simple padding approach to work, many different highway-related parameters have to be included.

Autoencoders
A more sophisticated approach would be to use autoencoders [8], creating an encoder-decoder structure. Autoencoders are a more sophisticated dimensionality reduction method which can transform tensors to a different dimensionality. The encoder can do the following transformation on the input data:

$$\mathbb{R}^{i\times l} \rightarrow \mathbb{R}^{j\times l} \tag{9.1}$$

(9.2)

where i = the number of input units (the number of road segments), j = the number of output units (equal to the other neural network) and l = the temporal input sequence length. The decoder will then do the reverse operation to retrieve the desired output dimensions. These autoencoder structures essentially corresponds to an extra neural network layer with the purpose of changing the dimensionality. An exemplary graph structure of this neural network is presented in Figure 9.2. Since the number of segments per road structure can change, a different autoencoder has to be trained for each specific road as the dimensionality of the input can differ. Traffic pattern dissimilarity between roads might also require a different autoencoder to be trained on each road since a different transformation is required to transform the data to this traffic representation which the pre-trained core neural network can then use.

Challenges
The main challenge of these neuroscopic traffic models the model should receive input which is information which helps the model distinguish different highways. Previously, different parameters have been introduced which can change both the traffic supply and demand. As many of these parameters have to be explicitly passed to the neural network, this can result in a rather complex neural network structure. The extra parameters and increase in complexity, training this model will also be a challenge.

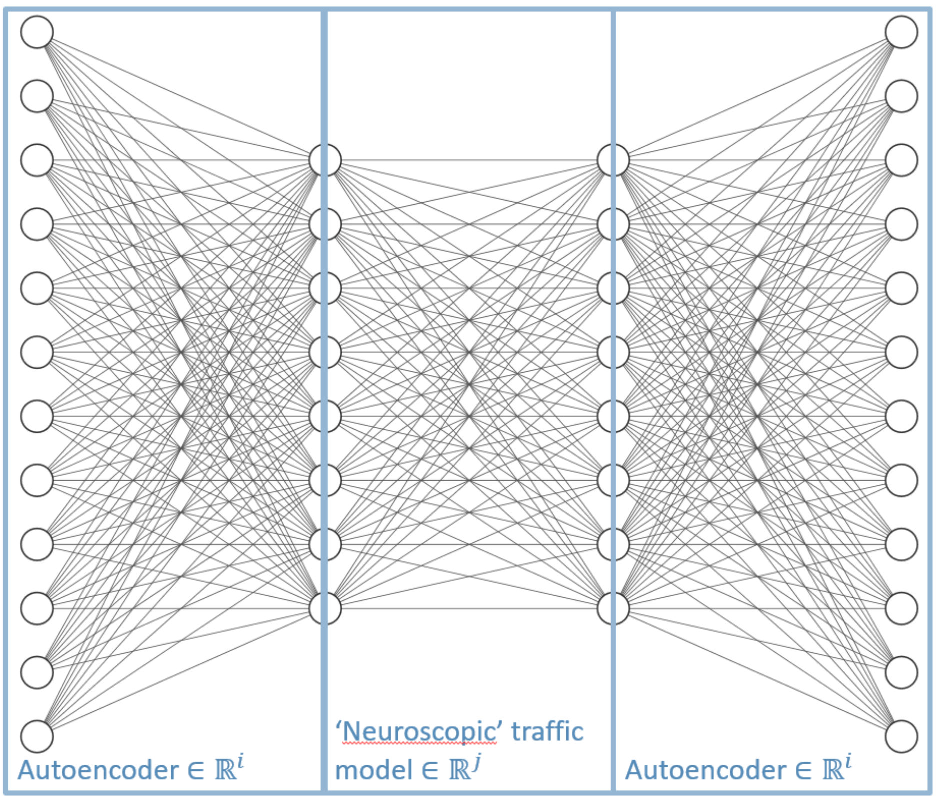

Autoencoder ∈ $\mathbb{R}^i$
'Neuroscopic' traffic model ∈ $\mathbb{R}^j$
Autoencoder ∈ $\mathbb{R}^i$

Figure 9.2: Autoencoder structure of 'neuroscopic' neural network. In this case, $i = 12$, $j = 8$ and the number of hidden layers is 2.

Besides, there still is a lot unclear on what this neuroscopic core network should actually learn and to what extent traffic over different highways actually is generalisable. Further research is therefore needed to assess the feasibility of this concept.

10 Conclusion & Recommendations

Neural networks are a promising method for short-term traffic flow prediction, as they can capture the complex nonlinear relations between traffic variables. However, most research focuses on developing complex neural network structures leading to lower errors compared to other architectures. This can result in overly complex models which are overfitted on a specific traffic network and hard to implement in practice [10]. Therefore, developing scale-free models which could be deployed on several different roads would aid the practical implementation of traffic predicting neural networks within an intelligent transportation system.

For the development of these scale-free models, several difficulties have to be overcome, one of them being a more thorough assessment of the accuracy of neural networks for traffic prediction. Bridging the gap between neural networks and traffic flow theory can provide insights in the potential and limitations of using neural networks for traffic flow prediction. In this thesis, the resulting predictions will be assessed in a spatiotemporal way, in order to be able to relate the resulting traffic prediction back to traffic flow theory. By doing this for several congestion patterns, conclusions can be drawn on whether the neural network can predict different types of traffic.

10.1. Conclusion

Based on this problem statement, the following research question could be formulated:

> ***What is the accuracy, assessed on different spatiotemporal levels, of neural network-based short-term traffic speed forecasting models on highways for different congestion patterns?***

In order to answer this research question in a structured way, this conclusion has been split up in several parts. First, the accuracy per congestion pattern is assessed from a traffic flow theory point of view. Secondly, the influence of several neural network architecture choices on model accuracy is explained. Thirdly, the added value of spatiotemporal error metrics is explained. Fourthly, the effect of using both flow and speed data is assessed. Lastly, the contributions to both the industry and science will be presented.

Accuracy per Congestion Pattern

Based on the literature research done in chapter 2, we could split up the traffic prediction process into two steps, namely (i) the **emergence** of congestion, describing at which point in space and time traffic will break down, and (ii) the **propagation** of congestion, which described how the congestion pattern would evolve after traffic broke down.

Little theory was available on predicting congestion emergence from a macroscopic perspective. As road capacity is not constant but rather stochastic and no clear variables could be found on which this probability depends, it is hard to predict the exact moment of traffic breakdown. However, plenty of theory was found on predicting the congestion propagation. Using shockwave theory, the propagation speed of congestion heads can be calculated using traffic flows and speeds. Since

this theoretical framework was based on some assumptions which are not always met, empirical congestion patterns were also discussed. These could roughly be divided into a synchronised pattern (SP), containing synchronised flow, and a general pattern (GP), in which stop-and-go waves occurred.
Although congestion emergence is hard to predict, predicting congestion propagation and finding the right 'shape' of the congestion pattern is possible. Both flow and speed were found to be relevant input parameters to predict this propagation, although predicting speeds as output is better for determining congestion than predicting flows. In general, synchronised patterns with a jam head or tail moving at a constant speed were found to be predictable, resulting in a high accuracy. However, predicting the exact emergence and propagation of stop-and-go waves was found to be impossible using a longer prediction horizon of 20 minutes, as the emergence of these congestion patterns can not be predicted. The accuracy of these predictions were therefore lower, although the model could beat the baseline by accurately predicting an average speed within this traffic phase.

Using Flow and Speed Data
Inputting flow and speed using a fully connected input layer resulted in the highest accuracy compared to efforts to combine flow and speed in a traffic phase. No clear conclusion could be drawn on whether adding flow data always leads to an increase in accuracy. On a dataset where a GPs were simulated, the best performing model used only speed data as input data. However, on a dataset containing mainly SPs, adding flow data resulted in a significant decrease of the prediction error. Both the model architecture as well as the complexity of the congestion patterns in the flow and speed data were found to be crucial factors. As for a sequence-to-sequence model the output parameters should be equal to the input parameters, both flow and speed have to be predicted. In case complex traffic flows are present in the dataset, adding flow data can lead to a decrease in accuracy.

Spatiotemporal Error Metrics
The spatial and temporal error metric were found to give relevant insight in the accuracy over space and time. The distribution of the MAE using the IQR gave insights in how the model error could deviate from day to day, probably due to a large variety of congestion patterns. In general, the spatial prediction error increased at locations where congestion occurs, such as bottleneck locations. The temporal errors peaked and during peak hours, especially at the start and end. The spatiotemporal error metric showed correlations of the error over space and time. Using this error could be concluded the model especially had trouble predicting the shockwave speeds.

Accuracy of Different Neural Network Architectures
This research focused on the use of a gated recurrent unit (GRU) recurrent neural network (RNN) for traffic speed prediction, since this network structure was found to perform well on input data containing temporal sequences where correlations between the input units remain stationary over time. A wide variety of RNN configurations has been proposed for short-term traffic predictions containing many different design choices, of which a taxonomy is given in Figure 3.10. The most promising neural network structures used speed and flow as model inputs and flow as model output, both spatiotemporal to capture correlations in the data over space and time. A taxonomy was proposed to distinct different design choices one can make when designing a neural network architecture. Within time series analysis, a GRU RNN neural network can be applied using a one-shot architecture, outputting a single value per output unit, or an autoregressive sequence-to-sequence model, outputting an output sequence where the previously outputted value is used when predicting the next sequence value. The sequence-to-sequence model lead to the highest prediction accuracy, probably due to its step-wise prediction process.

When looking at loss functions, a mean absolute error loss function lead to a higher accuracy and more stability in the results compared to a mean squared error loss function. Presumably, this is caused by the abrupt speed changes which can be present in the input data and high resulting errors when a traffic phase change is not correctly predicted.

Both the input sequence length and the prediction horizon were found to strongly impact the model accuracy. Although predicting GPs clearly benefitted from a shorter prediction horizon, longer prediction horizons still gave accurate results when predicting SPs. Larger input sequences lead to a higher accuracies, although the added value of a longer input sequence could not be substantiated using traffic flow theory. It is likely this increase in accuracy is caused by overfitting on the dataset, as the simulation data showed limited variation in demand patterns. It important to carefully assess the temporal horizon, also with respect to traffic flow theory, and to not only rely on accuracy when making a choice. The maximum travel time on a segment and shockwave speed of congestion can provide traffic flow theory-based guidelines on which maximum input sequence length can provide relevant information.

Main Contributions

The **contributions to the industry** can be summarised as follows. The conclusions drawn regarding the model accuracy give practical insights in several ways. Firstly, the feedback to traffic flow theory provides some theoretical limitations on the model accuracy. Secondly, by assessing the errors of different neural network architectures for different congestion patterns, guidelines are given on which traffic scenarios will be suitable for traffic forecasting. Thirdly, the proposed spatiotemporal error metrics are good to assess the practical usefulness of the predictions and can aid users in the interpretations of the model results. Besides, several other elements of this thesis can contain valuable information on designing traffic forecasting neural networks. An introductory chapter in both macroscopic traffic flow theory as well as (recurrent) neural networks gives potential developers an introduction in the theory behind both subjects. The proposed baseline model can be helpful to assess the usefulness of the predictions and the proposed GRU RNN provides a good starting point when designing a neural network.

The following **contributions to science** can be identified. Firstly, the taxonomy proposed in the literature analysis of RNNs can aid the development of new traffic forecasting RNNs, as it provides clear categories on which new neural network architectures can be designed. Secondly, the newly proposed baseline and GRU RNN models, as well as the dataset generated in the experimental setup, can be used as benchmarks to test more complex neural networks with. Thirdly, the results of the tested models give some guidelines on which neural network architectures, input data, loss functions and temporal horizons provides a good starting point when testing or proposing new neural network structures. Fourthly, by relating the prediction results to traffic flow theory, information is given on how predictable different congestion patterns are and what theoretical limitations exist on traffic forecasting. Lastly, the new spatiotemporal error metrics and model assessment could be used to assess and compare the results of a new neural network architecture in more depth.

10.2. Recommendations

As with the conclusions, the recommendations have been categorised into several categories. First, recommendations will be given on how the contents of this thesis can be used for proposing new neural network architectures for traffic forecasting. Secondly, options will be assessed to overcome the shortcomings of the simulation data used in this thesis. Thirdly, extensions to the spatiotemporal error metrics will be proposed, also with respect to using empirical data. Fourthly, an explanation is given on how the explainability of neural networks can help to validate whether a neural network predicts traffic in accordance with traffic flow theory.

Proposing New Neural Network Architectures

In case researchers want to design and implement new neural network architectures, this thesis can be used as a guideline for the design process or to evaluate its accuracy into more depth. Therefore, some recommendations are done which can help the development of new neural network architectures. First of all, the taxonomy proposed in chapter 3 can be used to pinpoint the areas in which one wants to make changes to the neural network architecture or to suggest new additions. It can also guide the process of designing better traffic forecasting neural networks, as it makes it easier to pinpoint the areas to which other researchers have made changes. In this way, one can easier use and extend the work done by other researchers. A recommendation would therefore be to further use and develop this taxonomy when designing new neural network structures. Secondly, the newly proposed error metrics can give more insight in the accuracy of a neural network. This enables a more critical assessment of the added value of a newly proposed neural network architecture. Other researchers are therefore encourage to adopt these error metrics. Thirdly, the neural network theory and discussed theoretical limitations of traffic flow prediction can set boundaries on whether an even higher accuracy is feasible. Its insights can also better guide the search to certain specific neural network elements which are likely to increase accuracy. Stricter guidelines on neural network architecture, possibly based on traffic flow theory or a thorough data analysis, might tame the proliferation of different neural network architectures which is happening at the moment. I would encourage other researchers to also take this into account when proposing new neural network architectures. Lastly, it would be interesting to test whether the conclusions made regarding the inclusion of both flow and speed data and the effect of the loss function also hold for other neural network types and architectures, as already large differences could be observed between the one-shot and sequence-to-sequence architecture.

Shortcomings of Data
It was concluded that using simulation data poses some shortcomings as the limited variation of congestion patterns can lead to overfitting and the resulting flow and speed patterns might miss some details which can be observed in empirically obtained data. First, it would be interesting to thoroughly assess the accuracy of a neural network trained on actual loop detector data using the methodology proposed in this thesis. Also, since actual loop detector data will contain a larger diversity of congestion patterns, it would be interesting to analyse a couple of these patterns in depth and to assess the accuracy on that level.

However, since the quality of the flow and speed data retrieved from the loop detectors in chapter 4 was quite low, questions were raised regarding the usefulness of this data for predictions. In case the loop detector data does not accurately represent the actual traffic situation, the predictions would still be poor regardless of a high accuracy. It would therefore be interesting to see what the effect is of the data quality of loop detector data on the predictions.

Spatiotemporal Error Metrics
With regard to the spatiotemporal error metrics, some recommendations can be made regarding improving these error metrics. First of all, the IQR gave insight in how the model accuracy can deviate from day to day. Although the depiction of it as an error band already provides some insight, it would be interesting to see whether this deviation can be quantified. For example, the median absolute deviation (MAD), defined as median$\left(\left|y_i - \bar{y}\right|\right)$, can be added for the MSE, to see how the error deviates over different days. Another way of visualising this would be to make a boxplot of the MAE values for each day, as this would provide even more information on the distribution. This would also lead to more quantitative results.

Besides assessing the error over both space and time or per day, it would also be interesting to do this per congestion pattern. Although this is quite easy when simulation data is used containing only a couple of congestion patterns, this is more difficult for empirical data. This could be done by first detecting and classifying congestion patterns, for example using the method proposed by [113] and then assessing the error per congestion pattern. In this way, the error per congestion pattern can also be assessed when using empirical data.

Explainability of Neural Networks
Besides, it would also be interesting to see to what extent explainable AI (xAI) can be used to get more insight on the factors the neural network bases its predictions on. Some researches also used xAI tools in the traffic domain (e.g. [16, 17]), however no clear conclusions could be drawn from the interpretation. Techniques like (a) visualisation techniques such as surrogate models such as SHAP [114] and LIME or partial dependence plots, (b) knowledge extraction techniques such as

rule extraction techniques and (c) influence methods such as sensitivity analysis, layer-wise relevance propagation and feature importance [115] could provide some relevant insights. However, as spatiotemporal predictions are quite a niche within AI applications, using these techniques in a traffic prediction context might be challenging.

Bibliography

[1] V. Knoop, A. Hegyi, M. Salomons, J. Lint, Y. Yuan, and R. Landman, *Traffic Flow Modelling and Control* (TU Delft, Transport & Planning, Delft, 2019) p. 546.

[2] A. Kestting, *Traffic jams: A collective phenomenon in space and time,* traffic-flow-dynamics.org/traffic-states (2021), traffic-flow-dynamics.org/traffic-states,

[3] B. S. Kerner, *Three-phase traffic theory and highway capacity,* Physica A: Statistical Mechanics and its Applications **333**, 379 (2004).

[4] B. S. Kerner, *Introduction to modern traffic flow theory and control: the long road to three-phase traffic theory* (Springer Science & Business Media, 2009).

[5] F. Bre, J. M. Gimenez, and V. D. Fachinotti, *Prediction of wind pressure coefficients on building surfaces using artificial neural networks,* Energy and Buildings **158**, 1429 (2018).

[6] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning* (2020).

[7] X. Zhou, *Understanding the convolutional neural networks with gradient descent and back-propagation,* in *Journal of Physics: Conference Series*, Vol. 1004 (IOP Publishing, 2018) p. 012028.

[8] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning,* Vol. 1 (MIT press Cambridge, 2016).

[9] G. Li, V. L. Knoop, and H. van Lint, *Multistep traffic forecasting using dynamic graph convolution: Interpretations of real-time spatial correlations,* Transportation Research Part C: Emerging Technologies [PREPRINT] (2020).

[10] M. G. Karlaftis and E. I. Vlahogianni, *Statistical methods versus neural networks in transportation research: Differences, similarities and some insights,* Transportation Research Part C: Emerging Technologies **19**, 387 (2011).

[11] I. Lana, J. J. Sanchez-Medina, E. I. Vlahogianni, and J. Del Ser, *From data to actions in intelligent transportation systems: a prescription of functional requirements for model actionability,* arXiv preprint arXiv:2002.02210 (2020).

[12] J. Van Lint and C. Van Hinsbergen, *Short-term traffic and travel time prediction models,* Artificial Intelligence Applications to Critical Transportation Issues **22**, 22 (2012).

[13] C. Hinsbergen, J. W. C. Lint, and F. Sanders, *Short term traffic prediction models,* 14th World Congress on Intelligent Transport Systems, ITS 2007 **7** (2007).

[14] L. Breiman, *Statistical modeling: The two cultures (with comments and a rejoinder by the author),* Statistical Science **16**, 199 (2001).

[15] S. Oh, Y.-J. Byon, K. Jang, and H. Yeo, *Short-term travel-time prediction on highway: A review of the data-driven approach,* Transport Reviews **35**, 4 (2015).

[16] Y. Wu, H. Tan, L. Qin, B. Ran, and Z. Jiang, *A hybrid deep learning based traffic flow prediction method and its understanding,* Transportation Research Part C: Emerging Technologies **90**, 166 (2018).

[17] A. Barredo-Arrieta, I. Laña, and J. Del Ser, *What lies beneath: A note on the explainability of black-box machine learning models for road traffic forecasting,* in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (IEEE, 2019) pp. 2232–2237.

[18] M. Treiber, A. Kesting, and D. Helbing, *Three-phase traffic theory and two-phase models with a fundamental diagram in the light of empirical stylized facts,* Transportation Research Part B: Methodological **44**, 983 (2010).

[19] B. Greenshields, J. Bibbins, W. Channing, and H. Miller, *A study of traffic capacity,* in *Highway research board proceedings*, Vol. 1935 (National Research Council (USA), Highway Research Board, 1935).

[20] M. Treiber and A. Kesting, *Traffic flow dynamics,* Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg (2013).

[21] M. Treiber, A. Hennecke, and D. Helbing, *Congested traffic states in empirical observations and microscopic simulations,* Physical review E **62**, 1805 (2000).

[22] B. S. Kerner, *The physics of traffic: empirical freeway pattern features, engineering applications, and theory* (Springer Science & Business Media, 2004).

[23] T. Schreiter, H. Van Lint, Y. Yuan, and S. Hoogendoorn, *Propagation wave speed estimation of freeway traffic with image processing tools,* Report (2010).

[24] B. S. Kerner, *Criticism of generally accepted fundamentals and methodologies of traffic and transportation theory: A brief review,* Physica A: Statistical Mechanics and its Applications **392**, 5261 (2013).

[25] B. S. Kerner, *A theory of traffic congestion at heavy bottlenecks,* Journal of Physics A: Mathematical and Theoretical **41**, 215101 (2008).

[26] W. Brilon, J. Geistefeldt, and M. Regler, *Reliability of freeway traffic flow: a stochastic concept of capacity,* in *Proceedings of the 16th International symposium on transportation and traffic theory*, Vol. 125143 (College Park Maryland, 2005).

[27] J. van Lint, *Reliable travel time prediction for freeways: Bridging artificial neural networks and traffic flow theory,* Doctoral thesis (2004).

[28] S. P. Hoogendoorn, H. van Lint, and V. L. Knoop, *Macroscopic modeling framework unifying kinematic wave modeling and three-phase traffic theory,* Transportation research record **2088**, 102 (2008).

[29] A. Ermagun and D. Levinson, *Spatiotemporal traffic forecasting: review and proposed directions,* Transport Reviews **38**, 786 (2018).

[30] C. M. Bishop, *Pattern recognition and machine learning* (springer, 2006).

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors,* nature **323**, 533 (1986).

[32] K. Koutroumbas and S. Theodoridis, *Pattern Recognition, 4th Edition* (Academic Press, Cambridge, MA, 2008) p. 984.

[33] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, *Scheduled sampling for sequence prediction with recurrent neural networks,* in *Advances in Neural Information Processing Systems* (2015) pp. 1171–1179.

[34] A. Graves, *Long short-term memory,* in *Supervised sequence labelling with recurrent neural networks* (Springer, 2012) pp. 37–45.

[35] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,* (2001).

[36] Y. Bengio, P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult,* IEEE transactions on neural networks **5**, 157 (1994).

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

[38] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, *On the properties of neural machine translation: Encoder-decoder approaches,* arXiv preprint arXiv:1409.1259 (2014).

[39] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation,* arXiv preprint arXiv:1406.1078 (2014).

[40] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators,* Neural networks **2**, 359 (1989).

[41] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to sequence learning with neural networks,* in *Advances in neural information processing systems* (2014) pp. 3104–3112.

[42] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, *Short-term traffic forecasting: Where we are and where we're going,* Transportation Research Part C: Emerging Technologies **43**, 3 (2014).

[43] I. Lana, J. Del Ser, M. Velez, and E. I. Vlahogianni, *Road traffic forecasting: Recent advances and new challenges,* IEEE Intelligent Transportation Systems Magazine **10**, 93 (2018).

[44] L. N. N. Do, N. Taherifar, and H. L. Vu, *Survey of neural network-based models for short-term traffic state prediction,* WIREs Data Mining and Knowledge Discovery **9** (2018), 10.1002/widm.1285.

[45] M. Cao, V. O. K. Li, and V. W. S. Chan, *A cnn-lstm model for traffic speed prediction,* (2020).

[46] J. Mackenzie, J. F. Roddick, and R. Zito, *An evaluation of htm and lstm for short-term arterial traffic flow prediction,* IEEE Transactions on Intelligent Transportation Systems **20**, 1847 (2019).

[47] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, *Lc-rnn: A deep learning model for traffic speed prediction,* in *IJCAI* (2018) pp. 3470–3476.

[48] Z. Zhao and Y. Zhang, *A traffic flow prediction approach: Lstm with detrending,* (2018) pp. 101–105.

[49] R. Asadi and A. C. Regan, *A spatio-temporal decomposition based deep neural network for time series forecasting,* Applied Soft Computing Journal **87** (2020), 10.1016/j.asoc.2019.105963.

[50] N. G. Polson and V. O. Sokolov, *Deep learning for short-term traffic flow prediction,* Transportation Research Part C: Emerging Technologies **79**, 1 (2017).

[51] Y. Tian and L. Pan, *Predicting short-term traffic flow by long short-term memory recurrent neural network,* 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity) , 153 (2015).

[52] R. Fu, Z. Zhang, and L. Li, *Using lstm and gru neural network methods for traffic flow prediction,* (2017) pp. 324–328.

[53] B. Liu, J. Cheng, K. Cai, P. Shi, and X. Tang, *Singular point probability improve lstm network performance for long-term traffic flow prediction,* (2017) pp. 328–340.

[54] H. Shao and B. H. Soong, *Traffic flow prediction with long short-term memory networks (lstms),* (2017) pp. 2986–2989.

[55] Z. Bartlett, L. Han, T. T. Nguyen, and P. Johnson, *Prediction of road traffic flow based on deep recurrent neural networks,* (2019) pp. 102–109.

[56] Z. Xue and Y. Xue, *Multi long-short term memory models for short term traffic flow prediction,* IEICE Transactions on Information and Systems **E101D**, 3272 (2018).

[57] Z. Li, G. Xiong, Y. Chen, Y. Lv, B. Hu, F. Zhu, and F. Y. Wang, *A hybrid deep learning approach with gcn and lstm for traffic flow prediction,* (2019) pp. 1929–1933.

[58] C. Zhang, J. J. Q. Yu, and Y. Liu, *Spatial-temporal graph attention networks: A deep learning approach for traffic forecasting,* IEEE Access **7**, 166246 (2019).

[59] J. Guo, Z. Wang, and H. Chen, *On-line multi-step prediction of short term traffic flow based on gru neural network,* (2017).

[60] M. Chen, G. Yu, P. Chen, and Y. Wang, *Traffic congestion prediction based on long-short term memory neural network models,* (2018) pp. 673–681.

[61] Y. Ci, G. Xiu, and L. Wu, *A short-term traffic flow prediction method based on long short-term memory network,* (2019) pp. 601–608.

[62] J. Li, L. Gao, W. Song, L. Wei, and Y. Shi, *Short term traffic flow prediction based on lstm,* (2019) pp. 251–255.

[63] B. Yang, S. Sun, J. Li, X. Lin, and Y. Tian, *Traffic flow prediction using lstm with feature enhancement,* Neurocomputing **332**, 320 (2019).

[64] M. F. Dixon, N. G. Polson, and V. O. Sokolov, *Deep learning for spatio-temporal modeling: Dynamic traffic flows and high frequency trading,* arXiv preprint arXiv:1705.09851 (2017).

[65] D. Kang, Y. Lv, and Y. Chen, *Short-term traffic flow prediction with lstm recurrent neural network,* in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)* (2017) pp. 1–6.

[66] A. S. Mihaita, H. Li, Z. He, and M. A. Rizoiu, *Motorway traffic flow prediction using advanced deep learning,* (2019) pp. 1683–1690.

[67] Z. Cui, R. Ke, Z. Pu, and Y. Wang, *Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction,* arXiv preprint arXiv:1801.02143 (2018).

[68] B. Yu, H. Yin, and Z. Zhu, *Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,* Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (2018), 10.24963/ijcai.2018/505.

[69] Z. Lu, W. Lv, Z. Xie, B. Du, and R. Huang, *Leveraging graph neural network with lstm for traffic speed prediction,* (2019) pp. 74–81.

[70] M. A. Chattha, S. A. Siddiqui, M. I. Malik, L. van Elst, A. Dengel, and S. Ahmed, *Kinn: Incorporating expert knowledge in neural networks,* (2019).

[71] Y. Ma, Z. Zhang, and A. Ihler, *Multi-lane short-term traffic forecasting with convolutional lstm network,* IEEE Access **8**, 34629 (2020).

[72] D. Han, J. Chen, and J. Sun, *A parallel spatiotemporal deep learning network for highway traffic flow forecasting,* International Journal of Distributed Sensor Networks **15** (2019), 10.1177/1550147719832792.

[73] J. Qu, X. Gu, and L. Zhang, *Improved ugrnn for short-term traffic flow prediction with multi-feature sequence inputs,* (2018) pp. 13–17.

[74] Y. Li, R. Yu, C. Shahabi, and Y. Liu, *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,* arXiv preprint arXiv:1707.01926 (2017).

[75] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, *Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting,* IEEE Transactions on Intelligent Transportation Systems (2019).

[76] T. Bogaerts, A. D. Masegosa, J. S. Angarita-Zapata, E. Onieva, and P. Hellinckx, *A graph cnn-lstm neural network for short and long-term traffic forecasting based on trajectory data,* Transportation Research Part C: Emerging Technologies **112**, 62 (2020).

[77] Z. Zheng, Y. Yang, J. Liu, H. N. Dai, and Y. Zhang, *Deep and embedded learning approach for traffic flow prediction in urban informatics,* IEEE Transactions on Intelligent Transportation Systems **20**, 3927 (2019).

[78] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, *Lstm network: a deep learning approach for short-term traffic forecast,* IET Intelligent Transport Systems **11**, 68 (2017).

[79] Q. Li, H. Tan, Y. Wu, L. Ye, and F. Ding, *Traffic flow prediction with missing data imputed by tensor completion methods,* IEEE Access **8**, 63188 (2020).

[80] X. Yang, Y. Yuan, and Z. Liu, *Short-term traffic speed prediction of urban road with multi-source data,* IEEE Access **8**, 87541 (2020).

[81] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, *Long short-term memory neural network for traffic speed prediction using remote microwave sensor data,* Transportation Research Part C: Emerging Technologies **54**, 187 (2015).

[82] J. Wang, F. Hu, and L. Li, *Deep bi-directional long short-term memory model for short-term traffic flow prediction,* (2017) pp. 306–316.

[83] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, *Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks,* Sensors **17**, 1501 (2017).

[84] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, *Lstm-based traffic flow prediction with missing data,* Neurocomputing **318**, 297 (2018).

[85] G. Dai, C. Ma, and X. Xu, *Short-term traffic flow prediction method for urban road sections based on space-time analysis and gru,* IEEE Access **7**, 143025 (2019).

[86] S. Du, T. Li, Y. Yang, X. Gong, and S. J. Horng, *An lstm based encoder-decoder model for multi-step traffic flow prediction,* (2019).

[87] A. E. Essien, G. Chukwkelu, and C. Giannetti, *A scalable deep convolutional lstm neural network for large-scale urban traffic flow prediction using recurrence plots,* (2019).

[88] Z. Guo, W. Zhong, F. Zhu, X. Li, F. Y. Wang, and G. Xiong, *Short term traffic flow forecast based on cm-gru networks,* (2019) pp. 163–168.

[89] Y. Kim, P. Wang, and L. Mihaylova, *Structural recurrent neural network for traffic speed prediction,* (2019) pp. 5207–5211.

[90] Y. Li and X. Wu, *Traffic flow prediction based on long short term memory network,* (2019) pp. 1157–1162.

[91] X. Luo, D. Li, Y. Yang, and S. Zhang, *Spatiotemporal traffic flow prediction with knn and lstm,* Journal of Advanced Transportation **2019** (2019), 10.1155/2019/4145353.

[92] L. Mou, P. Zhao, and Y. Chen, *Short-term traffic flow prediction: A long short-term memory model enhanced by temporal information,* (2019) pp. 2411–2422.

[93] P. Peng, D. Xu, H. Gao, Q. Xuan, Y. Liu, H. Guo, and D. He, *Short-term traffic flow prediction using attention-based long short-term memory network,* (2019) pp. 403–409.

[94] Z. Wang, R. Zhu, M. Zheng, X. Jia, R. Wang, and T. Li, *A regularized lstm network for short-term traffic flow prediction,* (2019) pp. 100–105.

[95] W. Wei, H. Wu, and H. Ma, *An autoencoder and lstm-based traffic flow prediction method,* Sensors (Switzerland) **19** (2019), 10.3390/s19132946.

[96] W. Xiangxue, X. Lunhui, and C. Kaixun, *Data-driven short-term forecasting for urban road network traffic based on data processing and lstm-rnn,* Arabian Journal for Science and Engineering **44**, 3043 (2019).

[97] D. Yang, H. Yang, P. Wang, and S. Li, *Multi-step traffic flow prediction using recurrent neural network,* (2019) pp. 803–808.

[98] G. Zheng, W. K. Chai, and V. Katos, *An ensemble model for short-term traffic prediction in smart city transportation system,* (2019).

[99] D. Chen, H. Wang, and M. Zhong, *A short-term traffic flow prediction model based on autoencoder and gru,* (2020) pp. 550–557.

[100] D. Chen, C. Xiong, and M. Zhong, *Improved lstm based on attention mechanism for short-term traffic flow prediction,* (2020) pp. 71–76.

[101] X. Chen, X. Xie, and D. Teng, *Short-term traffic flow prediction based on convlstm model,* (2020) pp. 846–850.

[102] M. C. Lee and J. C. Lin, *Dalc: Distributed automatic lstm customization for fine-grained traffic speed prediction,* (2020) pp. 164–175.

[103] Z. Lu, W. Lv, Y. Cao, Z. Xie, H. Peng, and B. Du, *Lstm variants meet graph neural networks for road speed prediction,* Neurocomputing **400**, 34 (2020).

[104] J. Zhou, H. Chang, X. Cheng, and X. Zhao, *A multiscale and high-precision lstm-gasvr short-term traffic flow prediction model,* Complexity **2020** (2020), 10.1155/2020/1434080.

[105] H. Zhu, Y. Xie, W. He, C. Sun, K. Zhu, G. Zhou, and N. Ma, *A novel traffic flow forecasting method based on rnn-gcn and brb,* Journal of Advanced Transportation **2020** (2020), 10.1155/2020/7586154.

[106] J. Van Lint, S. Hoogendoorn, and H. J. van Zuylen, *Accurate freeway travel time prediction with state-space neural networks under missing data,* Transportation Research Part C: Emerging Technologies **13**, 347 (2005).

[107] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization,* arXiv preprint arXiv:1412.6980 (2014).

[108] D. Masters and C. Luschi, *Revisiting small batch training for deep neural networks,* arXiv preprint arXiv:1804.07612 (2018).

[109] Y. Bengio, *Practical recommendations for gradient-based training of deep architectures,* in *Neural networks: Tricks of the trade* (Springer, 2012) pp. 437–478.

[110] Z. Zhang, M. Li, X. Lin, Y. Wang, and F. He, *Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies,* Transportation research part C: emerging technologies **105**, 297 (2019).

[111] B. L. Smith and M. J. Demetsky, *Traffic flow forecasting: Comparison of modeling approaches,* Journal of Transportation Engineering **123**, 261 (1997).

[112] I. Žliobaitė, M. Pechenizkiy, and J. Gama, *An overview of concept drift applications,* in *Big data analysis: new algorithms for a new society* (Springer, 2016) pp. 91–114.

[113] P. Krishnakumari, T. Nguyen, L. Heydenrijk-Ottens, H. L. Vu, and H. van Lint, *Traffic congestion pattern classification using multiclass active shape models,* Transportation Research Record **2645**, 94 (2017).

[114] S. M. Lundberg and S.-I. Lee, *A unified approach to interpreting model predictions,* in *Advances in neural information processing systems* (2017) pp. 4765–4774.

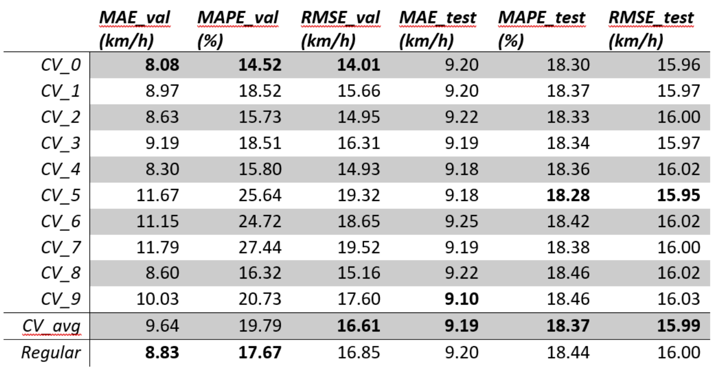
[115] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (2020).

Appendices

A
Results Obsolete Experiments

A.1. K-Fold Cross-Validation

Cross validation is often used in machine learning when the validation set is small, to cope with the fact that validation results can be noisy [30]. By dividing the training set into K folds, having K datasets where K-1 folds are the training set and 1 fold is the validation set, the model will be trained K times. In the end, either the performance result can be averaged or the best performing model can be chosen. A 1-layer 16-neuron GRU model has been trained on speed of the first dataset, using 10-fold cross-validation. When assessing the results of the cross-validation, shown in figure A.1,



	MAE_val (km/h)	*MAPE_val (%)*	*RMSE_val (km/h)*	*MAE_test (km/h)*	*MAPE_test (%)*	*RMSE_test (km/h)*
CV_0	**8.08**	**14.52**	**14.01**	9.20	18.30	15.96
CV_1	8.97	18.52	15.66	9.20	18.37	15.97
CV_2	8.63	15.73	14.95	9.22	18.33	16.00
CV_3	9.19	18.51	16.31	9.19	18.34	15.97
CV_4	8.30	15.80	14.93	9.18	18.36	16.02
CV_5	11.67	25.64	19.32	9.18	**18.28**	**15.95**
CV_6	11.15	24.72	18.65	9.25	18.42	16.02
CV_7	11.79	27.44	19.52	9.19	18.38	16.00
CV_8	8.60	16.32	15.16	9.22	18.46	16.02
CV_9	10.03	20.73	17.60	**9.10**	18.46	16.03
CV_avg	9.64	19.79	**16.61**	**9.19**	**18.37**	**15.99**
Regular	**8.83**	**17.67**	16.85	9.20	18.44	16.00

Figure A.1: Results Cross-Validation

one can see that there are large differences in the error metrics on the validation testset, however, the distances for the test set are rather small. When comparing the average cross-validation results versus a model which was not cross-validated, the differences in errors are also negligible.

A.2. Output activation one-shot

To assess which output activation function works best for the one-shot model, both a linear and a sigmoidal activation function have been tested. The results, shown in table A.1, show that the lowest error metric values are observed for the linear output activation, although the difference is quite small.

Table A.1: Results of linear and sigmoidal output activation function for one-shot model dataset 1.

		RMSE (km/h)		MAE (km/h)		MAPE (%)	
Output activation	Layers	128	256	128	256	128	256
Sigmoid	1	16.93	16.90	7.72	7.70	15.05	15.10
	2	16.87	16.82	7.68	7.66	15.17	15.20
	4	16.79	16.71	7.67	7.73	15.50	15.65
Linear	1	16.91	**16.86**	7.68	7.65	15.07	15.05
	2	16.89	16.94	7.61	7.62	15.01	15.10
	4	16.99	17.12	**7.56**	7.61	**14.99**	15.39

A.3. Output activation sequence-to-sequence

To find the best output function for the sequence-to-sequence model, both a sigmoidal and a linear output function have been tried. The sequence-to-sequence model was trained on speed data of the second dataset. The results in table A.2 show that a sigmoidal output function leads to significantly better results.

Table A.2: Results of linear and sigmoidal output activation function for sequence-to-sequence model dataset 2.

	RMSE (km/h)			MAE (km/h)			MAPE (%)		
Output activation	108	216	432	108	216	432	108	216	432
Sigmoid	7.13	7.11	**6.96**	4.32	4.27	**4.28**	6.47	6.37	6.33
Linear	8.04	7.35	7.03	4.89	4.97	4.38	7.56	7.56	6.52

B
Heatmaps of Best Test Models

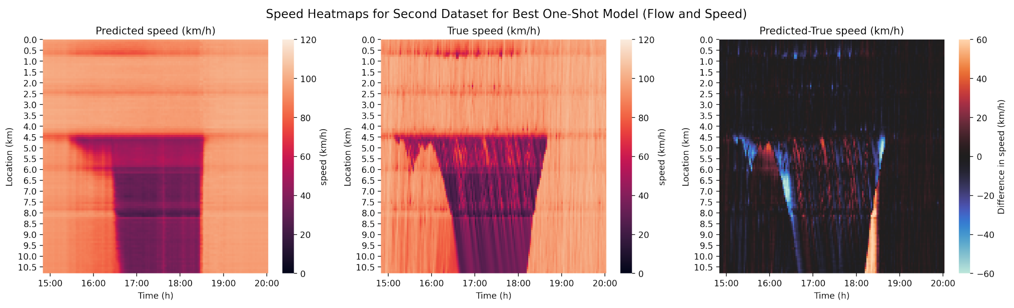
In this appendix, the resulting speed heatmaps are presented for the best performing models resulting from section 7.2. They are split out per dataset, and the results for both the best one-shot as well as the best sequence-to-sequence model (in terms of accuracy) are shown.

B.1. Dataset 2

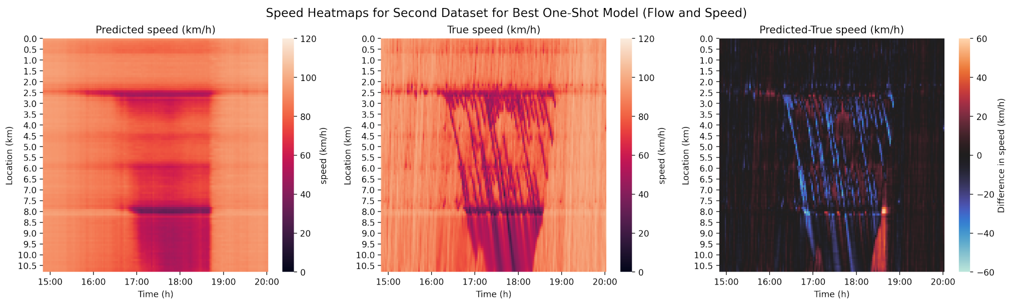
When looking at the results, in general, one can conclude that the finer patterns in the speed predictions are not captured by the neural network. Furthermore, both GRU RNNs could not accurately capture the shock waves separating either the general pattern or synchronised pattern phase with the free-flow phase. However, some differences were observed too. The best one-shot model consisted of 2 layers, each having 216 nodes, having a MAE loss function and trained on both flow and speed data. When assessing the results, it can be seen that the shockwaves separating the free-flow and MSP or GP phase are not predicted quite accurately, as it seems they propagate with an infinite speed (represented by the horizontal borders).

The best sequence-to-sequence model was trained on speed data only, having a MAE loss function and 216 units per hidden layer. Compared to the one-shot model, the shockwave speeds are much better captured since the spatiotemporal border along which a traffic phase change takes place (between SP/GP and free-flow) propagates upstream in the speed prediction instead of propagating (almost) instantaneously for the one-shot models. Within the free-flow conditions, almost no difference could be observed between the models.

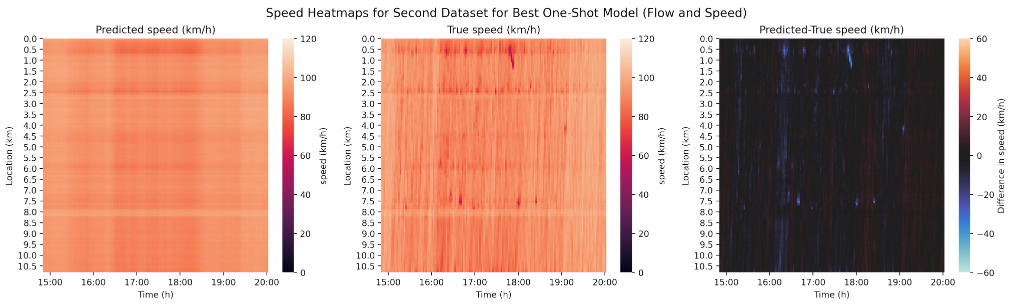
However, for the GP in the first scenario, artefacts in the congestion solution were also visible, as several narrow spikes of free-flow intruded these heatmaps. This indicates that for this particular setting, the mdoel could have trouble accurately predicting the solution of this jam wave.



(a) Speed heatmaps first scenario.

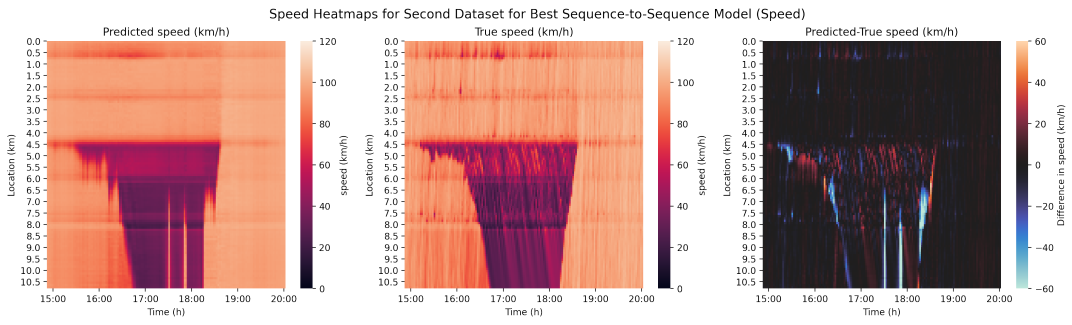


(b) Speed heatmaps second scenario.

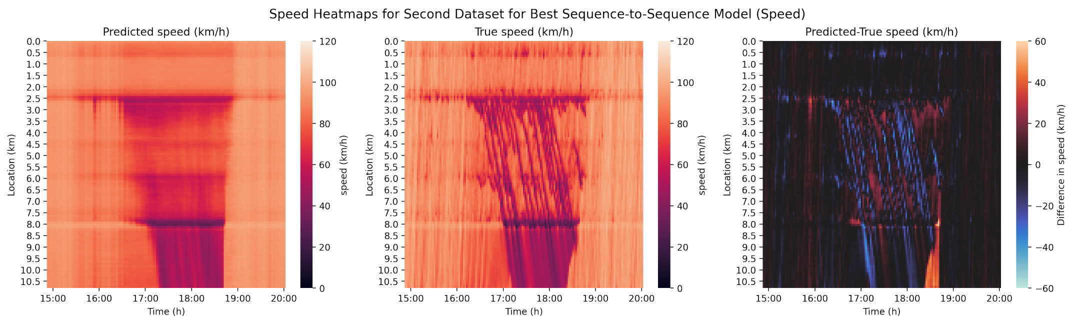


(c) Speed heatmaps third scenario.

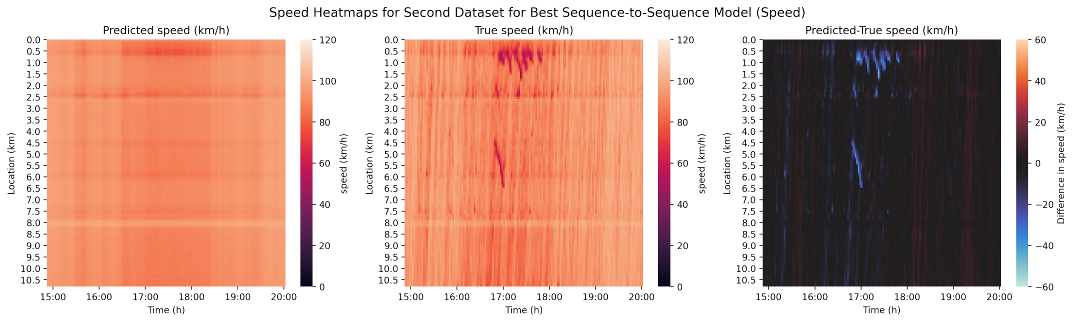
Figure B.1: Speed heatmaps of prediction result best one-shot model trained on dataset 2.



(a) Speed heatmaps first scenario.



(b) Speed heatmaps second scenario.

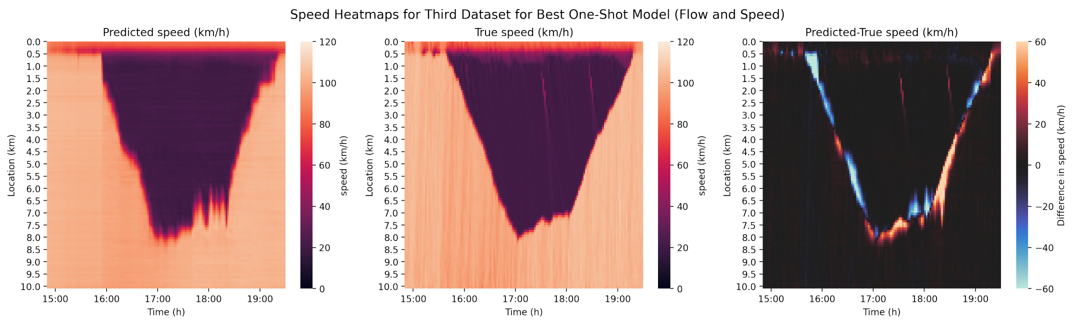


(c) Speed heatmaps third scenario.

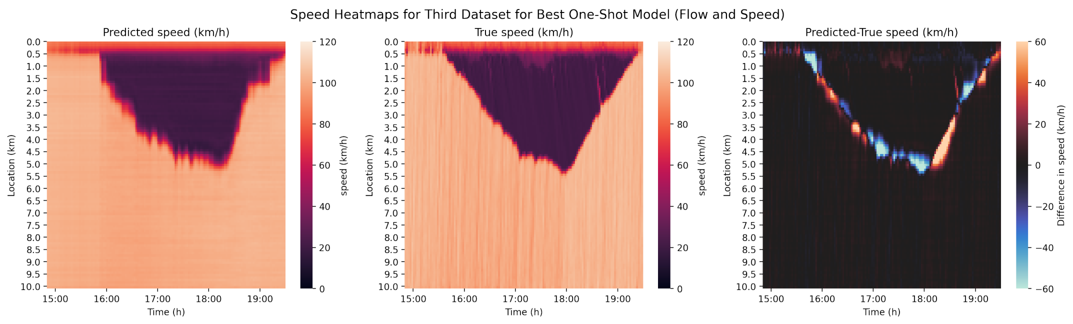
Figure B.2: Speed heatmaps of predictions best sequence-to-sequence model trained on dataset 2.

B.2. Dataset 3

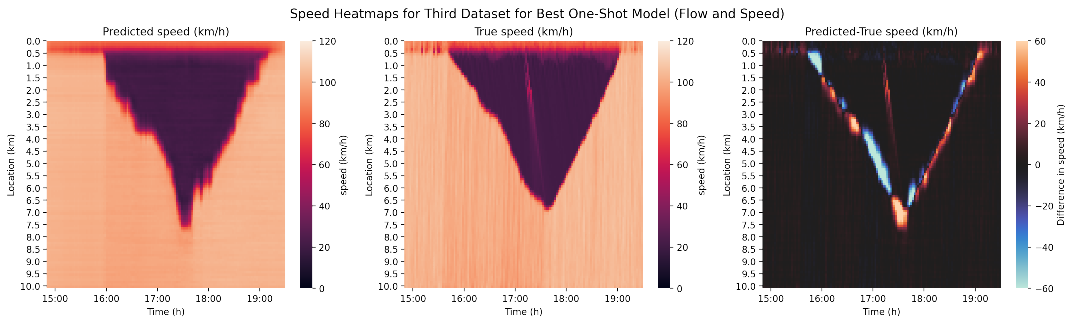
In general, the predictions of the models from the third dataset are of much higher accuracy compared to the second dataset. The predicted congestion patterns are quite accurate as the shock-waves between the free-flow and congested state are tracked quite accurately



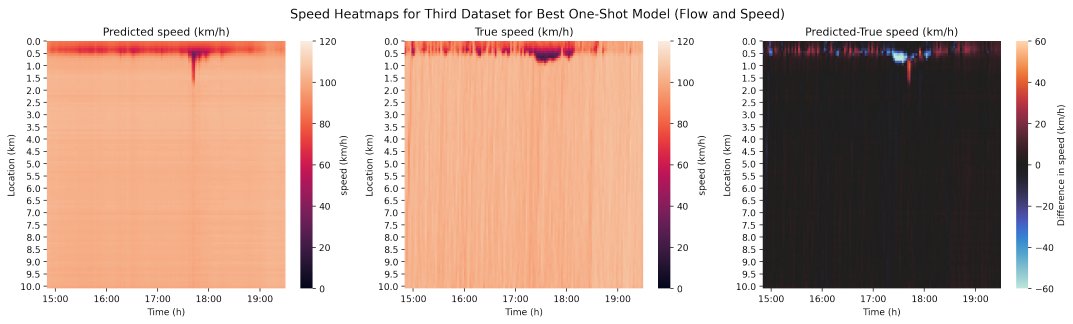
(a) Speed heatmaps first scenario.



(b) Speed heatmaps second scenario.

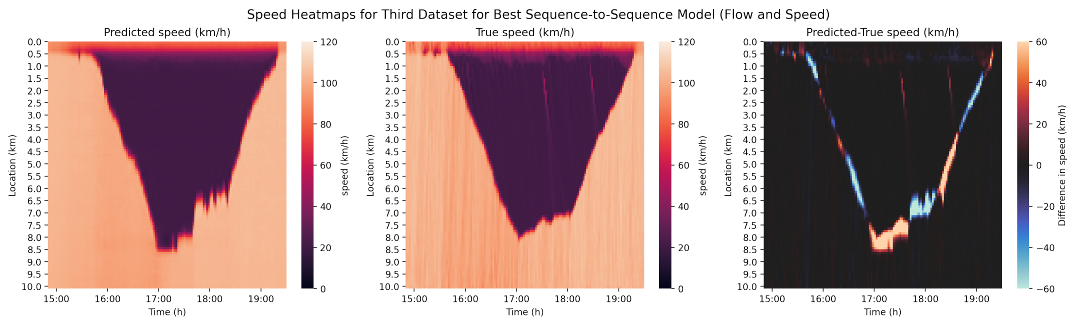


(c) Speed heatmaps third scenario.

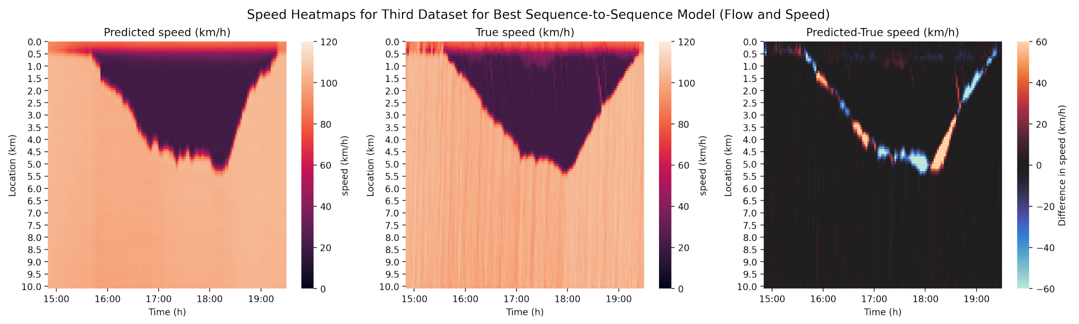


(d) Speed heatmaps fourth scenario.

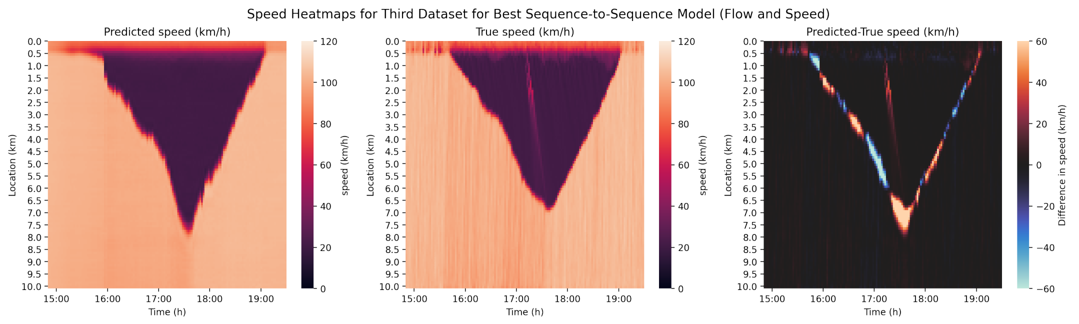
Figure B.3: Speed heatmaps of prediction result of best one-shot model trained on dataset 3.



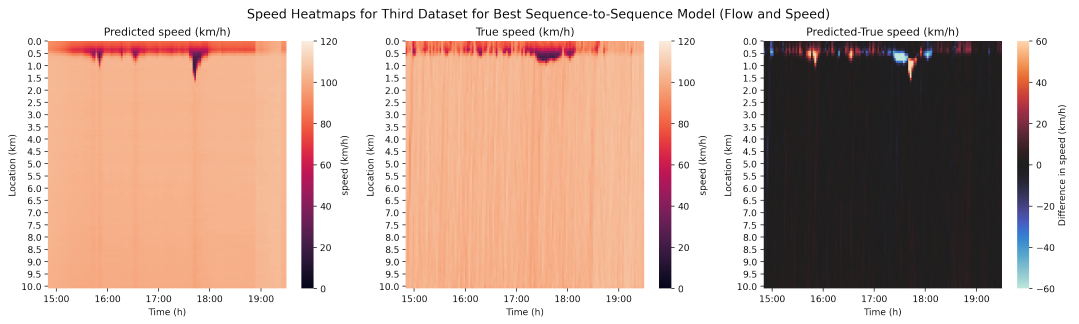
(a) Speed heatmaps first scenario.



(b) Speed heatmaps second scenario.



(c) Speed heatmaps third scenario.



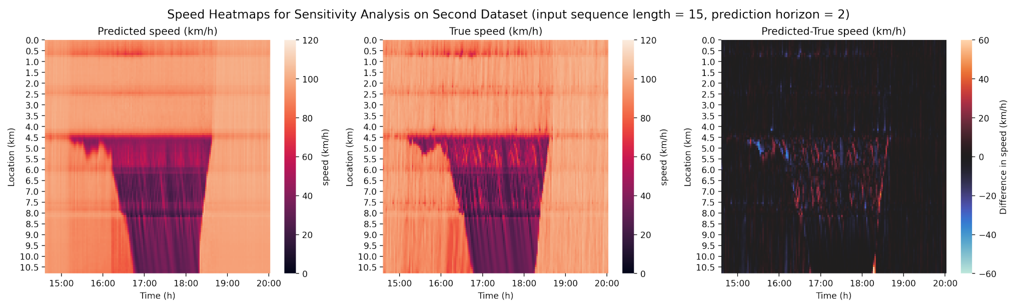
(d) Speed heatmaps fourth scenario.

Figure B.4: Speed heatmaps of prediction result of best sequence-to-sequence model trained on dataset 3.

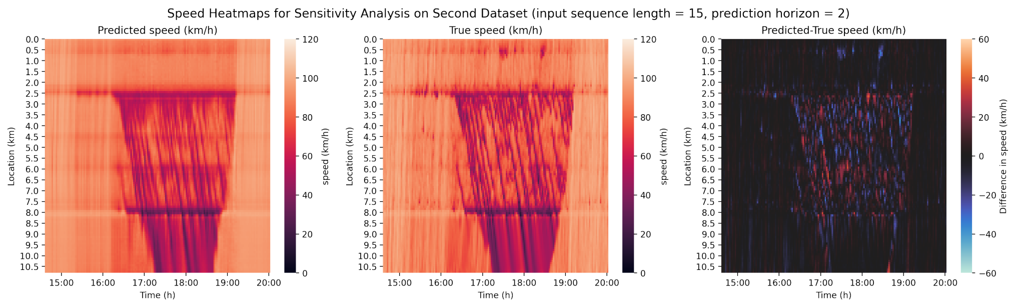
C
Heatmaps of Sensitivity Analysis

In this appendix, exemplary predicted speed heatmaps are presented resulting from the sensitivity analysis proposed in section 5.5. They are split out per dataset and prediction horizon, presenting only the results of the input sequence length which lead to the highest accuracy for that prediction horizon.

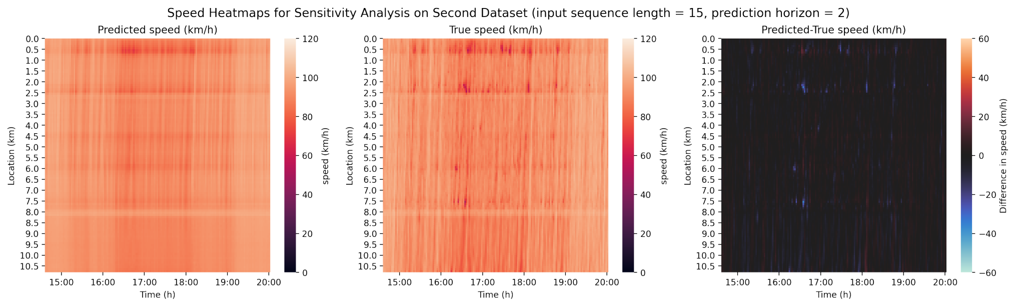
C.1. Heatmaps Dataset 2



(a) Speed heatmaps first scenario.

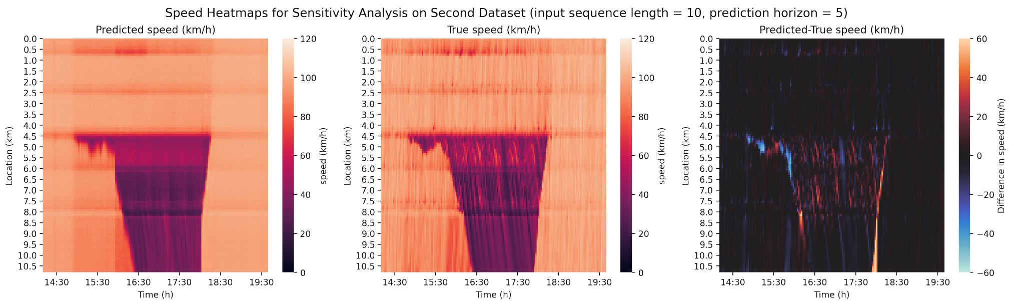


(b) Speed heatmaps second scenario.

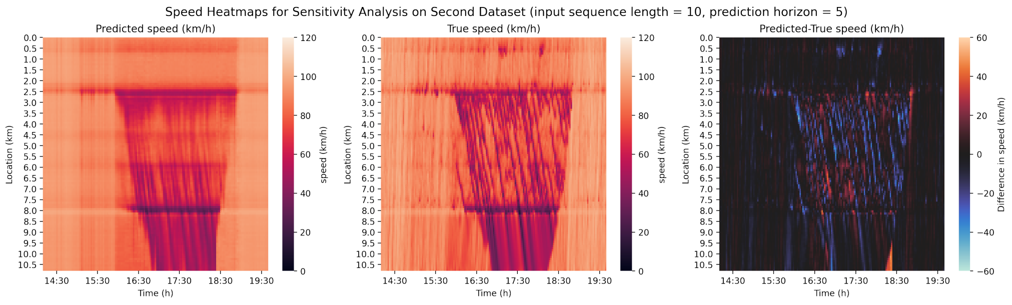


(c) Speed heatmaps third scenario.

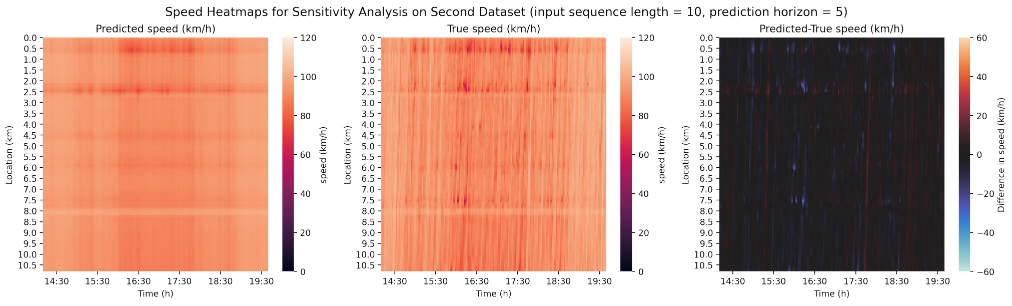
Figure C.1: Speed heatmaps of sensitivity analysis with prediction horizon = 2 steps on dataset 2.



(a) Speed heatmaps first scenario.

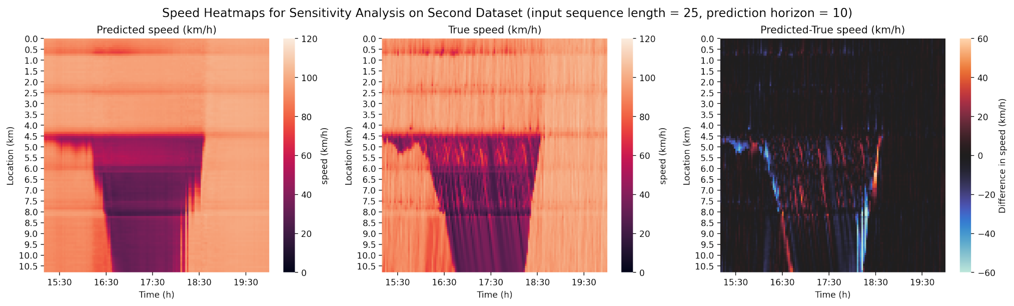


(b) Speed heatmaps second scenario.

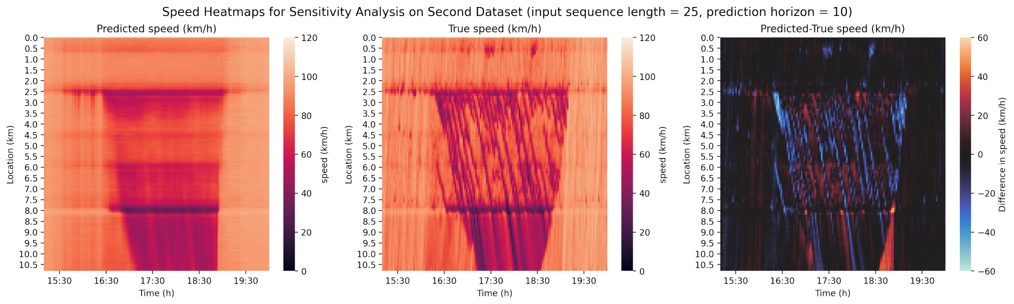


(c) Speed heatmaps third scenario.

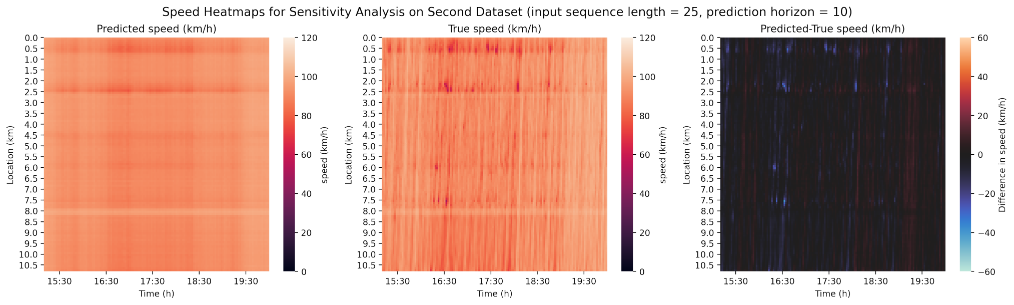
Figure C.2: Speed heatmaps of sensitivity analysis with prediction horizon = 5 steps on dataset 2.



(a) Speed heatmaps first scenario.



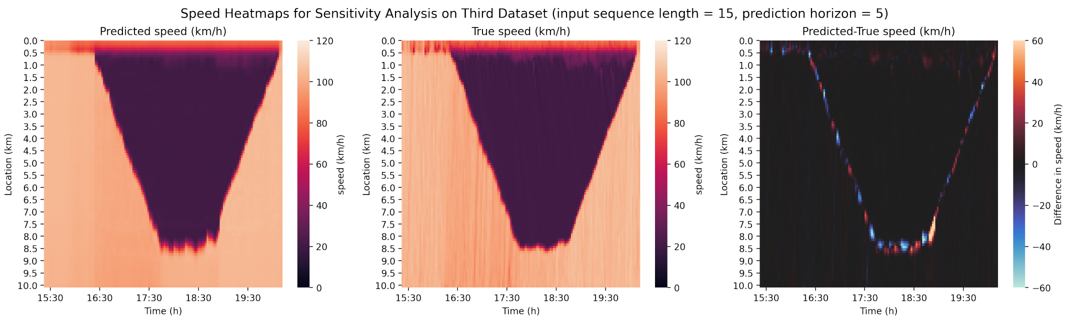
(b) Speed heatmaps second scenario.



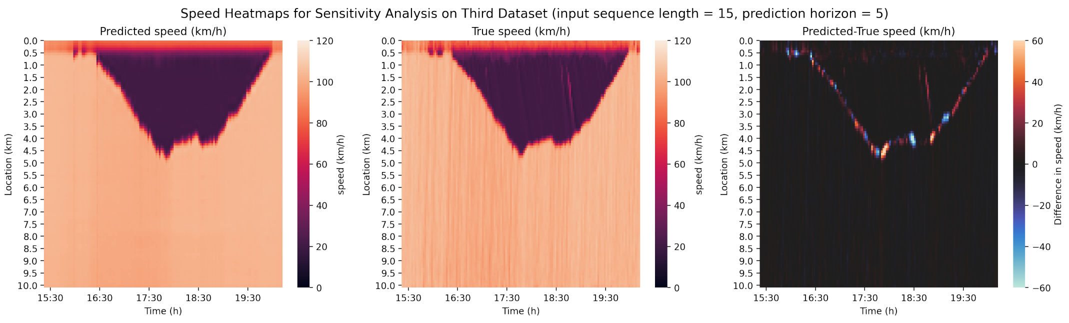
(c) Speed heatmaps third scenario.

Figure C.3: Speed heatmaps of sensitivity analysis with prediction horizon = 10 steps on dataset 2.

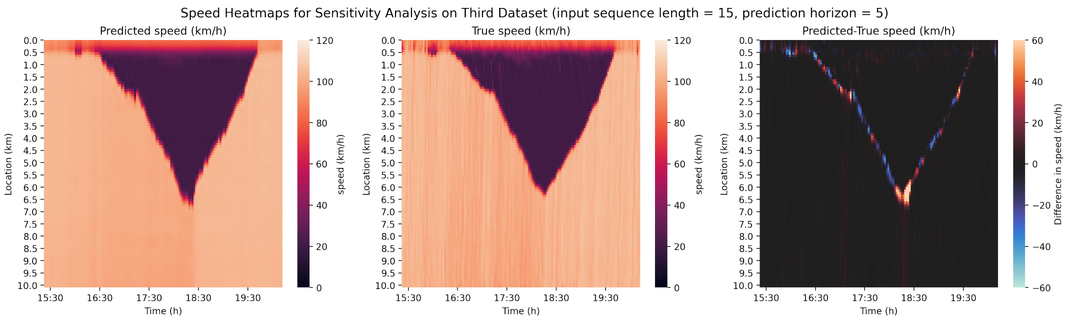
C.2. Heatmaps Dataset 3



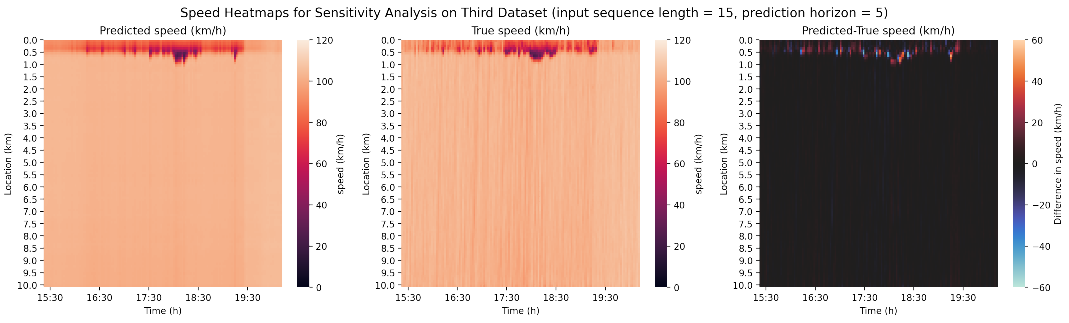
(a) Speed heatmaps first scenario.



(b) Speed heatmaps second scenario.

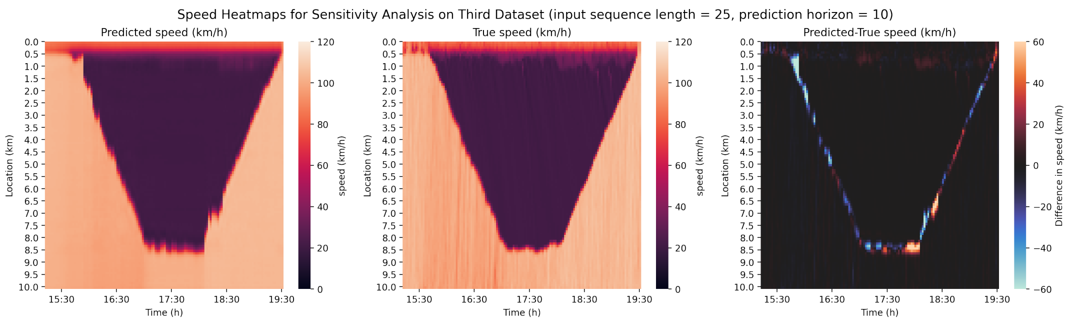


(c) Speed heatmaps third scenario.

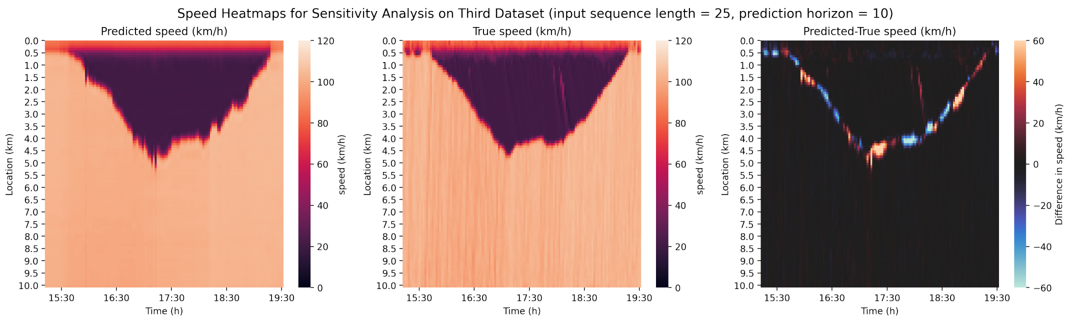


(d) Speed heatmaps fourth scenario.

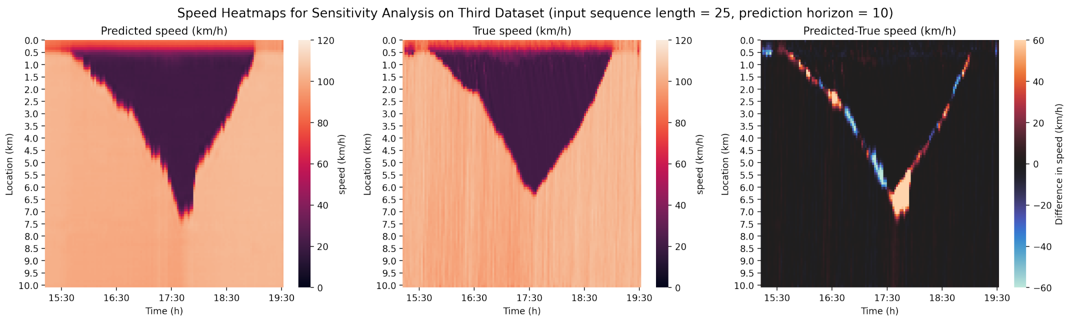
Figure C.4: Speed heatmaps of sensitivity analysis with prediction horizon = 5 steps on dataset 3.



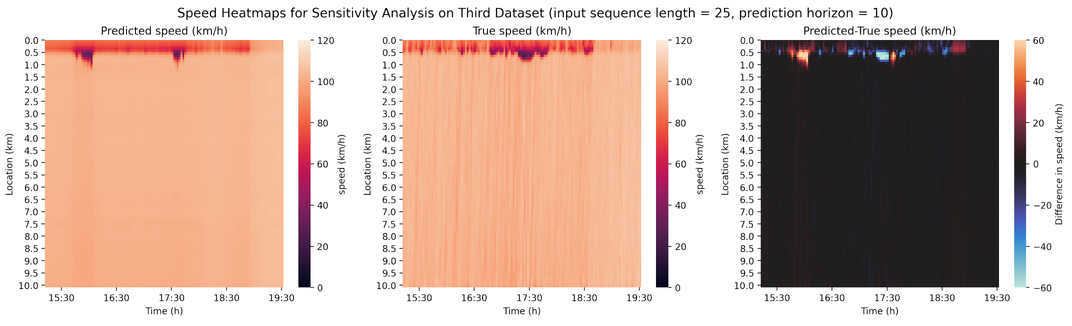
(a) Speed heatmaps first scenario.



(b) Speed heatmaps second scenario.

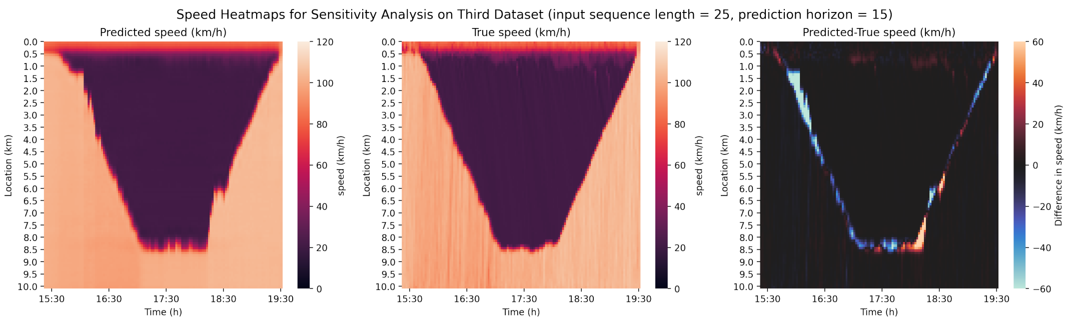


(c) Speed heatmaps third scenario.

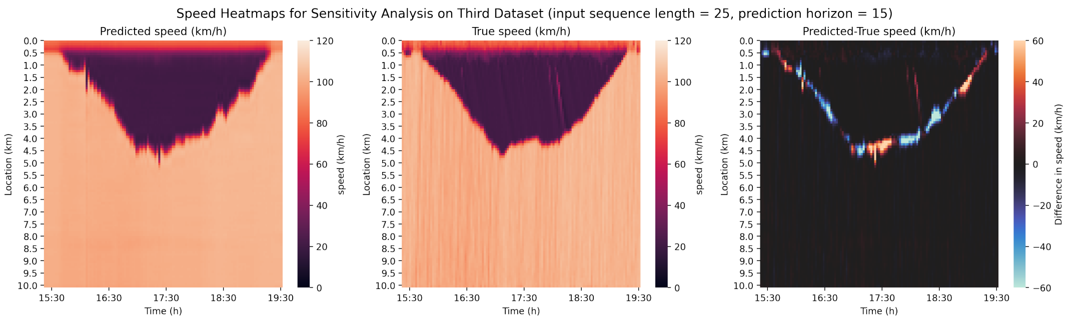


(d) Speed heatmaps fourth scenario.

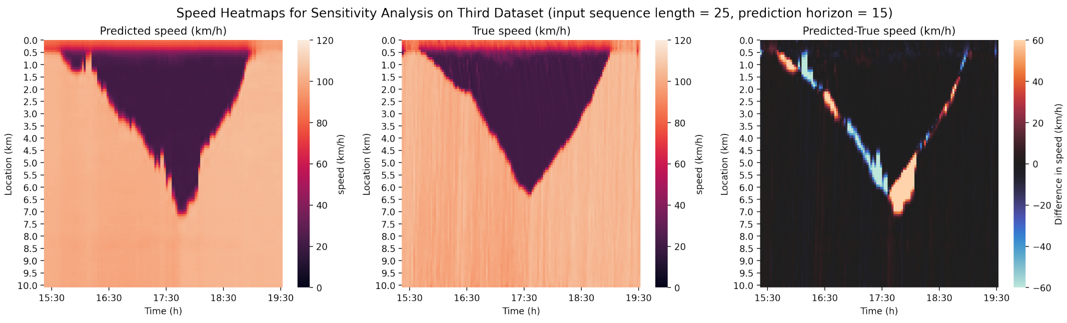
Figure C.5: Speed heatmaps of sensitivity analysis with prediction horizon = 10 steps on dataset 3.



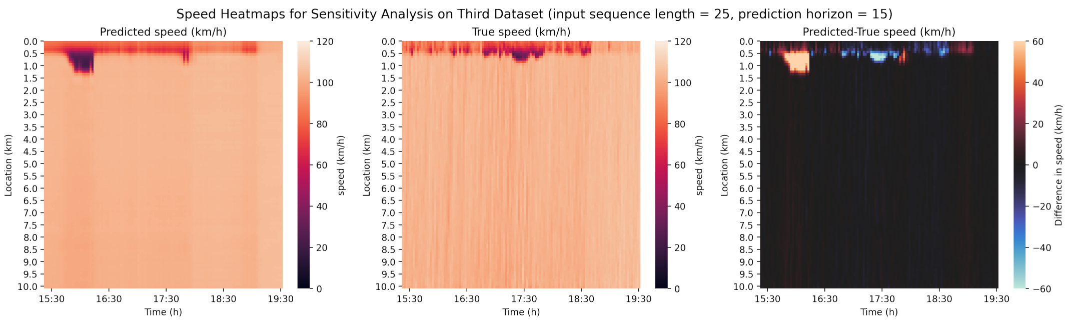
(a) Speed heatmaps first scenario.



(b) Speed heatmaps second scenario.



(c) Speed heatmaps third scenario.



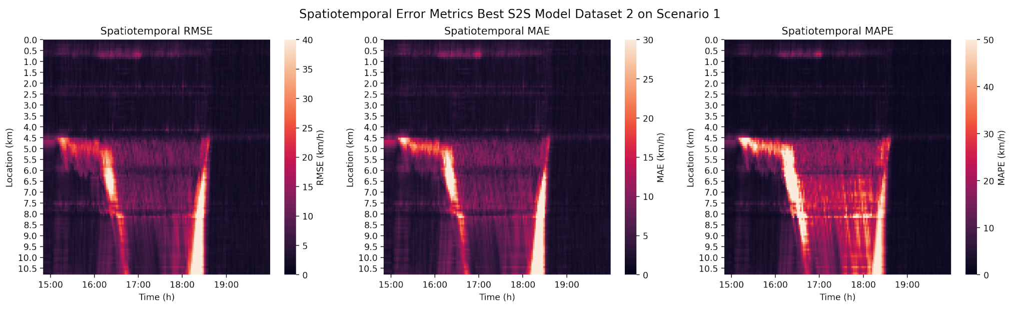
(d) Speed heatmaps fourth scenario.

Figure C.6: Speed heatmaps of sensitivity analysis with prediction horizon = 15 steps on dataset 3.

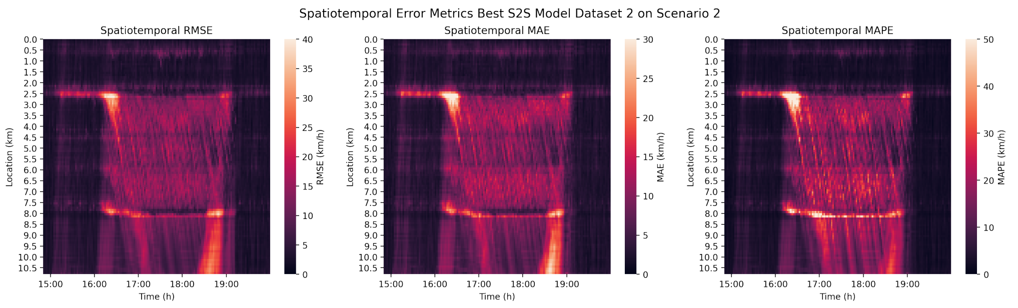
D Spatiotemporal Error Heatmaps per Scenario

In this appendix, the spatiotemporal error heatmaps are presented for each individual scenario. Using these heatmaps, it is easier to identify per congestion pattern which points in space and time had high prediction errors. The error metrics are split out per dataset.

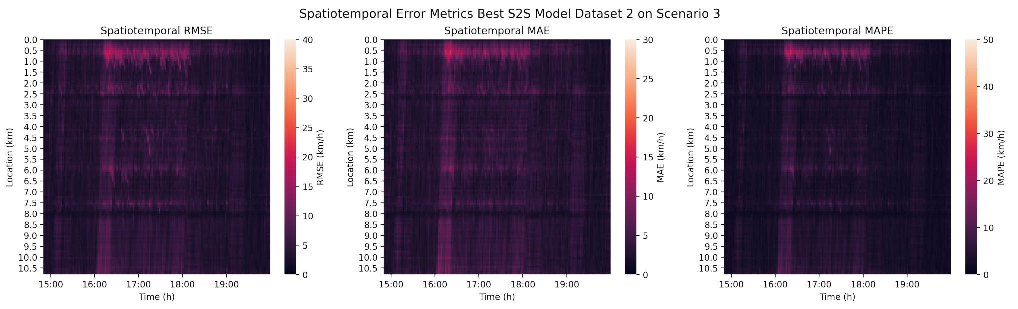
D.1. Heatmaps Dataset 2



(a) Spatiotemporal error first scenario.



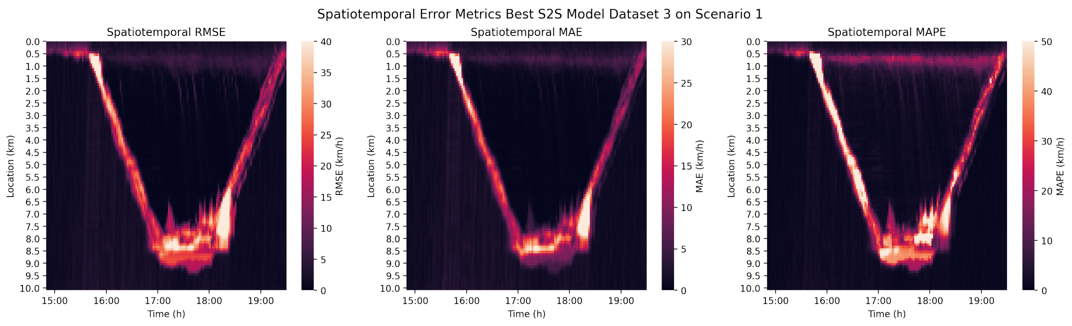
(b) Spatiotemporal error second scenario.



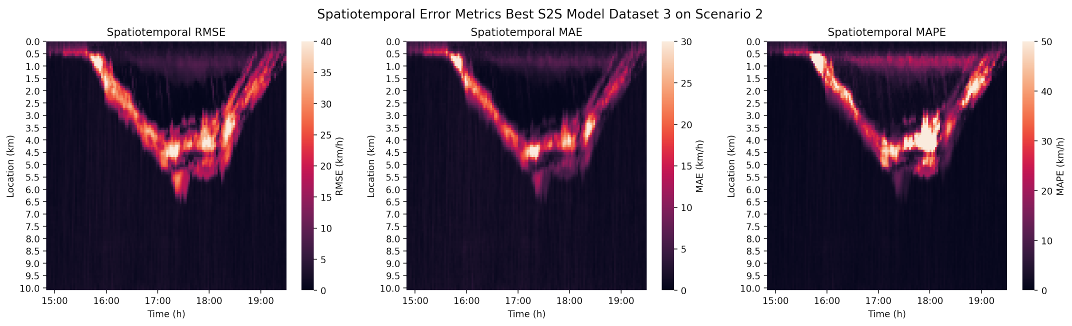
(c) Spatiotemporal error third scenario.

Figure D.1: Spatiotemporal error of different scenarios for the second dataset.

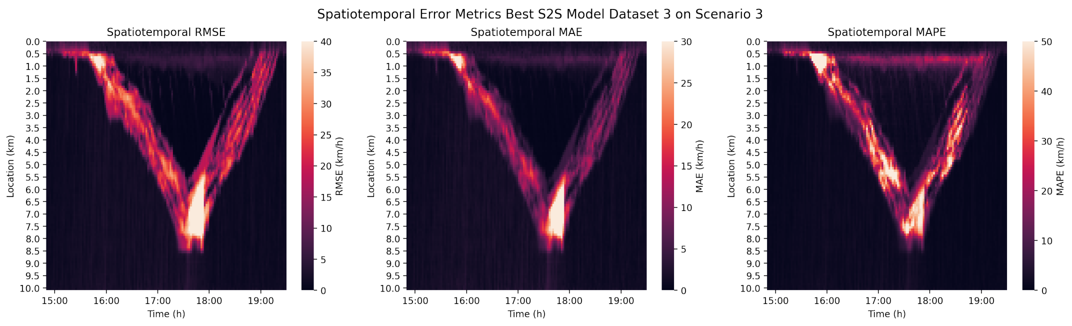
D.2. Heatmaps Dataset 3



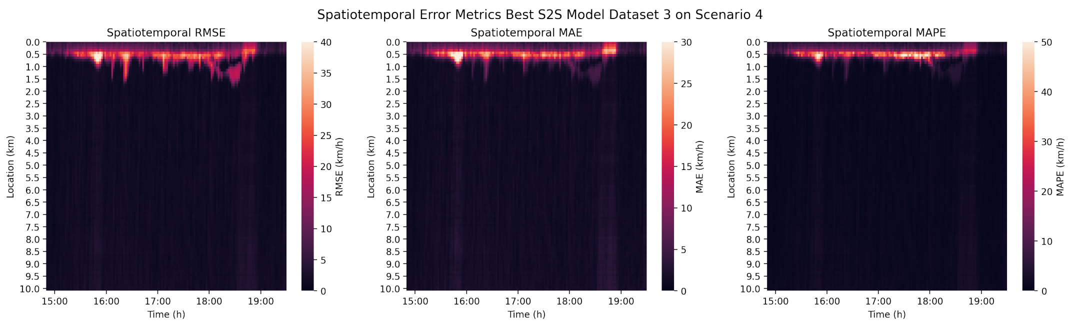
(a) Spatiotemporal error first scenario.



(b) Spatiotemporal error second scenario.



(c) Spatiotemporal error third scenario.



(d) Spatiotemporal error fourth scenario.

Figure D.2: Spatiotemporal error of different scenarios for the second dataset.

Delft
University of
Technology