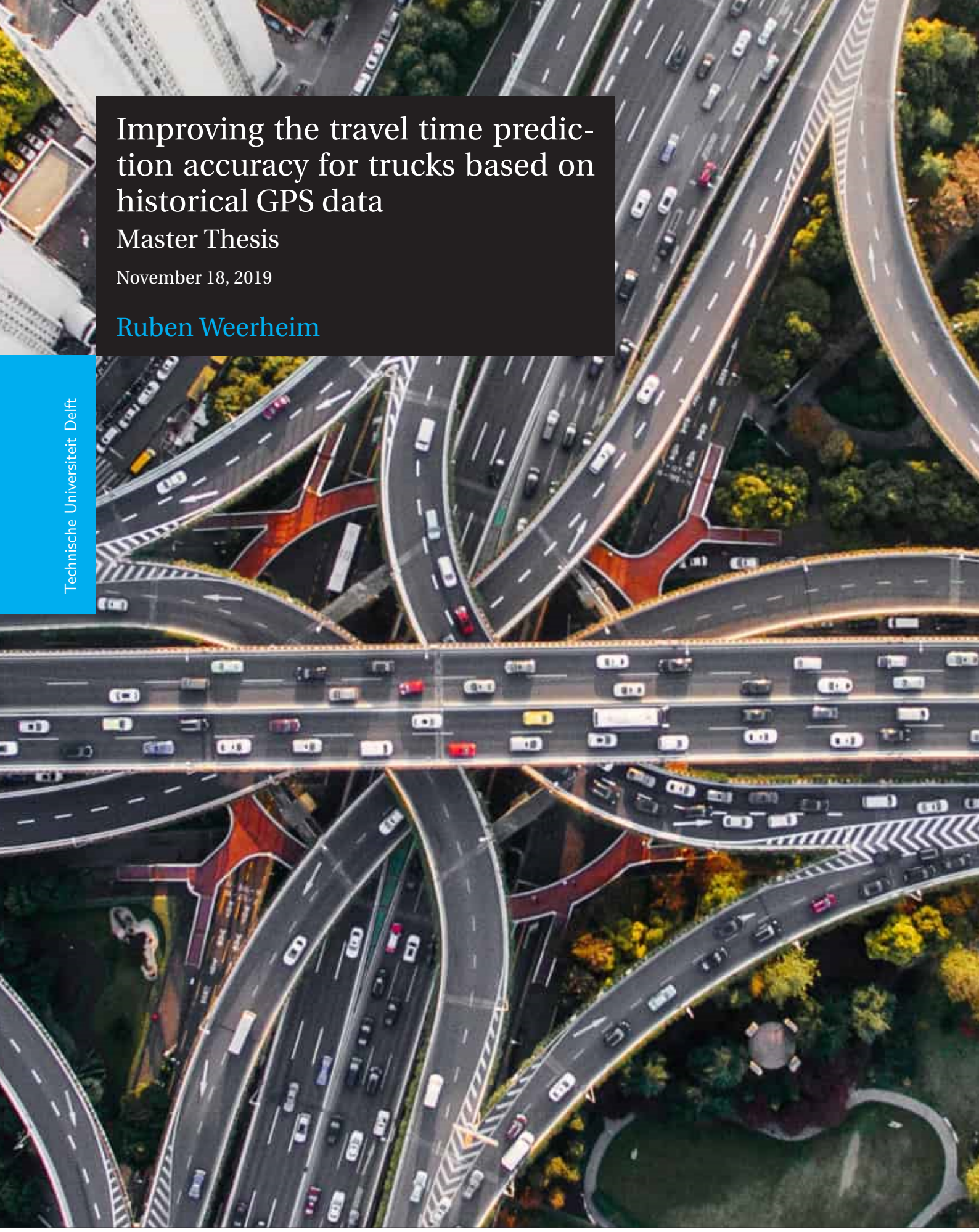

Improving the travel time prediction accuracy for trucks based on historical GPS data

Master Thesis

November 18, 2019

Ruben Weerheim

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

IMPROVING THE TRAVEL TIME PREDICTION ACCURACY FOR TRUCKS BASED ON HISTORICAL GPS DATA

MASTER THESIS
NOVEMBER 18, 2019

by

Ruben Weerheim

Faculty of Mechanical, Maritime and Materials Engineering
Department of Marine and Transport Technology,
to be defended publicly on Wednesday November 27, 2019 at 14:15.

Author:	Ruben Weerheim	
Student number:	4358295	
Report number:	2019.TEL.8380	
Thesis committee:	Prof. dr. R. R. Negenborn	TU Delft, Committee chair
	Dr. ir. X. Jiang	TU Delft, supervisor
	P. Agterberg	Company supervisor
	Dr. V.L. Knoop	TU Delft, Committee member
	Dr. ir. J.W. Frouws	TU Delft, Committee member

PREFACE

This report provides a master thesis about improving the travel time prediction accuracy for trucks based on historical GPS data. This master thesis is part of the master track Transport Engineering and Logistics (TEL) at the faculty of 3mE. The research has been conducted at ORTEC, who also provided me the graduation assignment. This assignment is a following up of a previous research done at ORTEC, where this research aims to further improve the travel time prediction accuracy of the previous research. The graduation project has been an intense period, but ultimately gives me a satisfied feeling about the result that I am proud of. This brings me to the end of a little more than five years studying at the TU Delft, where I also obtained my bachelor degree in Aerospace Engineering. It has been a time of hard working, but the obtained experience and knowledge is priceless which I will take with me for the rest of my life.

I would like to thank everyone who has supported me during my master thesis. First, I would like to thank my daily supervisor Pim Agterberg from ORTEC. I really appreciate the endless effort you have put into helping me at every moment during the research. Secondly, I would like to thank my supervisors Dr. Ir. X. Jiang and Prof. dr. R. R. Negenborn from the TU Delft for their valuable advice and guidance, from an academic perspective, during the graduation project. Lastly, I would like to thank my family and friends for the support during the master thesis, but also during all other years that I have spent at TU Delft.

Ruben Weerheim
Delft, November 2019

SUMMARY

This master thesis has been made at ORTEC in Zoetermeer. ORTEC is one of the world's leaders in optimization software and analytics solutions. The research study is conducted at ORTEC Routing and is one of the domains of the planning and scheduling software. ORTEC Routing aims to develop software that optimizes route planning in order to reduce transportation cost. This optimization also include the accuracy of the travel time predictions from point A to B. If there is a too tight schedule, then the schedule would lead to pressure to the driver, late fees and reputation damage, but a too loose schedule would result in wasted capacity.

To calculate the travel time, first the shortest path has to be found. From the shortest path, the travel time can be derived. This is the sum of the length, divided by the speed, of each road that is included in the shortest path. To calculate the shortest path, detailed map data provided by the company *HERE Technologies* is used. This map data includes road networks of different countries and many properties of each road, such as lane width, speed limit, whether it is a bridge, etc. To find a correct shortest path and to obtain accurate travel time predictions, it is important that the assigned speed to each road in the road network is predicted accurately. Currently, each road in the map belongs to one of the 20 road types, where each road type has a speed that is assigned by the customer. These speeds are based on the experience of the implementation consultant (customer). However, dividing the roads in the map into 20 road types, is quite a rough road classification. Also, the speeds that are assigned to each road type is based on the experience from the customer. This reduces the accuracy of the travel time predictions.

To improve the travel time predictions at ORTEC, den Heijer [1] introduced a new method. This method learns from realized GPS data, which was obtained from a customer that operates with trucks. The coordinates of the GPS data are used to match each GPS point to a road in the map. In this way, the corresponding road properties (speed limit, lane width, tunnel, ramp, region, etc.) of the matched road can be looked up and linked to the driven speed of the GPS point. This results in point-based data. By finding relations between the road properties (independent variables) and the driven speed (dependent variable), the speed of each road in the map can be predicted more accurately. Besides the point-based data, den Heijer also introduced trip-based data, for training the model and to predict the speed of each road in the map. Trip-based data is based on the average road properties and travel time between two GPS points.

Different models were developed by den Heijer and were either based on linear regression or random forest. The random forest model, which was trained on point-based data with dependent variable logspeed, had the best travel time prediction accuracy. This accuracy is $sMdAPE_{TT}$ 13.8% compared to $sMdAPE_{TT}$ 18.4% of the current method (20 road types). This is an enormous improvement. However, the shortcoming of the random forest method is that it is not able to learn from trip-based data, which could have improved the travel time prediction accuracy further. The linear regression model can be trained on both point- and trip-based data, but can only model linear relationships. This limits the prediction accuracy. Therefore, this research focuses on developing a model with a higher prediction accuracy than linear regression and random forest that can be used for both point- and trip-based data.

The travel time prediction methods in literature, except the method from den Heijer, focus on improving the travel time predictions for one route or road. This research study focuses on improving travel time predictions for an entire road network. Therefore, the travel time prediction method, developed by den Heijer, is the only method that is applicable to this research. However, to improve the method of den Heijer, prediction methods that are used by the travel time prediction methods in literature, can be adapted to this research problem. The prediction methods that can be used are linear regression, random forest, support vector regression, gradient boosting and neural networks. These methods are able to predict a continuous output (speed) and are able to learn from a data set, to generalize this to an entire road network. Subsequently, a method trade-off was performed to find the best prediction method that overcomes the shortcomings of the linear and random forest models. After performing the method trade-off and a sensitivity analysis, it was found that the neural network is the most suitable prediction method, for both point- and trip-based data.

The factors that influence the travel time can be categorized into five classes: vehicle classification, temporal factors, weather factors, road engineering factors and unpredictable factors. The weather and unpredictable factors were considered to be out of scope for this research. The vehicle classification is trucks, and is fixed because the GPS data is collected from trucks. The road engineering factors (road properties) are firstly used to find relationships between the road properties and speeds. At the end of this research, the influence of the temporal factor was researched by splitting the data set into rush and non-rush hours. This is because the temporal factor could not be used as independent variable, since only one speed can be assigned to each road in the digital map.

The neural network could be used as prediction method by finding relationships between the independent variables (speed limit, region, lane width, etc.) and the dependent variable (speed). In total, 21 independent variables were used to find relationships between the road engineering factors and speeds in the data sets. This is the training process. After the model is trained, the speed of each road in the map can be predicted individually. In total, 21 different neural network models were developed. These models differ in type of training data (point- or trip-based data), loss function (MSE_{LF}, $MAPE_{LF}$, MAE_{LF}, $sMAPE_{LF}$ and $sMdAPE_{LF}$) and dependent variable (speed, logspeed and pace (1/speed)). During hyperparameter tuning, it was found that the best number of neurons and hidden layers differ between the models. This depends on whether the dependent variable pace or loss function $sMdAPE_{LF}$ was used.

After the best set of hyperparameters was found, 21 different neural network models were trained on old and new data. The old data was collected and preprocessed by den Heijer, with a frequency of 5 minutes. The new data was collected and preprocessed during this research, with a frequency of 2 minutes. The data sets were obtained from different customers that operate with trucks in the Benelux or the Netherlands. It was expected that with new data, the quality of the trip-based data would improve and as a result the speed and travel time prediction accuracy of the neural networks.

From the results, it was concluded that the random forest models, from den Heijer, had the best travel time prediction accuracy with $sMdAPE_{TT}$ 13.8% for old data, and $sMdAPE_{TT}$ 12.4% for new data. This means that none of the neural networks did outperform the $sMdAPE_{TT}$ of den Heijer's models. This means that the neural network is not able to learn and predict the speeds in such a way, that the travel time prediction accuracy is improved. Furthermore, possible relationships between the speed and travel time prediction accuracy were investigated. It was found that the speed prediction accuracy in MSE_{Speed} was mostly related to the travel time prediction accuracy in $sMdAPE_{TT}$. The model with the lowest MSE_{Speed} had the lowest $sMdAPE_{TT}$ for new data, which was not true for the old data. However, the MSE_{Speed} can be used as a first estimation of the $sMdAPE_{TT}$, but not to find the model with the best $sMdAPE_{TT}$.

It was expected that by increasing the data frequency from 5 to 2 minutes, a relatively higher travel time prediction accuracy would be achieved, compared to den Heijer's models. However, from the results, it is unclear whether a higher frequency of 2 minutes, compared to 5 minutes, improves the travel time prediction accuracy. Therefore, it is recommended to research the influence of the data frequency on the same data set. Furthermore, the temporal factor was researched by splitting the new data set into a rush and non-rush hour data set. The results showed that the travel time prediction accuracy can be improved even further, by taking the temporal factor into account.

After all, based on the results that were obtained in this research, a new speed prediction model that outperforms the current speed prediction models, with respect to $sMdAPE_{TT}$, could not be developed. However, it could be neither concluded that the neural network does not provide better travel time predictions for other data sets. Also, it cannot be concluded that neural networks, which are trained on trip-based data, will not outperform the random forest models with a higher data frequency than 2 minutes.

SAMENVATTING

Deze master thesis is gemaakt bij ORTEC in Zoetermeer. ORTEC is één van de wereldleiders in optimalisatie software and analytische oplossingen. Het onderzoek is uitgevoerd bij ORTEC Routing en is één van de domeinen van de planning- en schedulingsoftware van ORTEC. ORTEC Routing streeft ernaar om software te ontwikkelen dat de route planning optimaliseert om zo de transport kosten te verminderen. Deze optimalisatie bevat ook de nauwkeurigheid van de reistijdvoorspellingen voor voertuigen van punt A naar B. Als de planning te strak is, kan dit leiden tot druk voor de bestuurder, vertragings kosten en/of reputatieschade. Echter, een te ruime planning leidt tot onbenutte capaciteit van het voertuig waardoor mogelijk te veel voertuigen zijn ingezet.

Om de reistijd te berekenen, moet er eerst het kortste pad worden gevonden. Hierna kan de reistijd worden berekend aan de hand van het kortste pad. Dit is de som van de lengte, gedeeld door de snelheid, van elke weg die is opgenomen in het kortste pad. Om het kortste pad te berekenen is er gebruik gemaakt van gedetailleerde kaartgegevens. Deze zijn geleverd door het bedrijf *HERE Technologies* en bevatten wegnetwerken van verschillende landen en veel eigenschappen van elke weg. Voorbeelden van wegeigenschappen zijn de breedte van een weg, het snelheidslimiet, of de weg een brug is, enz. Om een correcte kortste pad te vinden en om nauwkeurige reistijdvoorspellingen te verkrijgen, is het belangrijk dat de toegewezen snelheid aan elke weg in het wegennet nauwkeurig wordt voorspeld. Momenteel behoort elke weg op de kaart tot één van de 20 wegtypen, waarbij elk wegtype een snelheid heeft die kan worden toegewezen door de klant. Deze snelheden zijn gebaseerd op de ervaring van de klant van ORTEC. Het opdelen van de wegen in de digitale kaart in 20 wegtypen is echter een vrij groffe verdeling van de wegen. Ook zijn de snelheden die aan elk wegtype worden toegewezen, gebaseerd op de ervaring van de klant. Deze twee tekortkomingen verminderen de nauwkeurigheid van de reistijdvoorspellingen.

Om de reistijdvoorspellingen bij ORTEC te verbeteren, introduceerde den Heijer[1] een nieuwe methode die leert van gerealiseerde GPS-data. De GPS-data werd verkregen van een klant die opereert met vrachtwagens. De coördinaten van de GPS-data kunnen worden gebruikt om elk GPS-punt te matchen met een weg in de digitale kaart. Op deze manier kunnen de wegeigenschappen (snelheidslimiet, rijstrookbreedte, tunnel, op/afrit, regio, enz.) van de gematchte weg worden opgezocht en gekoppeld aan de gereden snelheid van het GPS-punt. Dit resulteert in point-based data. Door relaties te vinden tussen de wegeigenschappen (onafhankelijke variabelen) en de gereden snelheid (afhankelijke variabele), kan de snelheid van elk weg op de kaart hierna nauwkeuriger worden voorspeld. Naast de point-based data, introduceerde den Heijer ook trip-based data voor het trainen van het model, en om de snelheid van elke weg op de kaart te voorspellen. Trip-based data is gebaseerd op de gemiddelde wegeigenschappen en de reistijd tussen twee GPS-punten.

Verschillende modellen werden ontwikkeld door den Heijer en waren gebaseerd op lineaire regressie of random forest. Het random forest model, dat werd getraind met point-based data en de afhankelijke variabele logspeed, had de beste nauwkeurigheid van de reistijdvoorspellingen. Deze nauwkeurigheid was $\text{sMdAPE}_{\text{TT}}$ 13,8%, vergeleken met $\text{sMdAPE}_{\text{TT}}$ 18,4% van de huidige methode (20 wegtypen). Dit is een enorme verbetering. Echter, de tekortkoming van de random forest is dat het niet in staat is om te leren van trip-based data. Dit had de nauwkeurigheid van de reistijdvoorspellingen verder kunnen verbeteren. Het lineaire regressiemodel kan worden getraind op zowel point- als trip-based data, maar kan alleen lineaire relaties modelleren. Dit beperkt de voorspellingsnauwkeurigheid. Daarom richt dit onderzoek op het ontwikkelen van een model met een hogere voorspellingsnauwkeurigheid dan lineaire regressie en random forest en dat kan worden gebruikt voor zowel point- als trip-based data.

Uitgezonderd van de methode van den Heijer, zijn alle andere methodes voor reistijdvoorspellingen gericht op het verbeteren van de reistijdvoorspellingen voor één route of weg. Dit onderzoek richt zich op het verbeteren van reistijdvoorspellingen voor een heel wegennet. Daarom is de door den Heijer ontwikkelde reistijdvoorspellingsmethode de enige methode die kan worden toegepast op dit onderzoek. Om de methode van den Heijer te verbeteren voor dit onderzoek, kunnen voorspellingsmethoden worden gebruikt van de reistijdvoorspellingsmethoden in de literatuur. De voorspellingsmethoden die kunnen worden gebruikt zijn

lineaire regressie, random forest, support vector regression, gradient boosting and neurale netwerken. Deze methoden kunnen een continue output (snelheid) voorspellen en zijn in staat om te leren van een dataset en deze vervolgens te generaliseren voor een heel wegennet. Vervolgens was een methode trade-off uitgevoerd om de beste voorspellingsmethode te vinden, die niet de tekortkomingen van de lineaire en random forest modellen heeft. Na het uitvoeren van de methode trade-off en een gevoeligheids analyse, bleek dat de neurale netwerk de meest geschikte voorspellingsmethode voor zowel point- als trip-based data is.

De factoren die de reistijd beïnvloeden, kunnen worden onderverdeeld in vijf klassen: voertuigclassificatie, tijdsfactoren, weersfactoren, wegfactoren en onvoorspelbare factoren. Het weer en onvoorspelbare factoren werden niet onderzocht in dit onderzoek. De voertuigclassificatie voor dit onderzoek is vrachtwagens, omdat de GPS-data werden verzameld van vrachtwagens. De wegfactoren (wegeigenschappen) werden gebruikt om relaties te vinden tussen de wegeigenschappen en snelheden. Aan het einde van dit onderzoek is ook de invloed van de tijdsfactoren onderzocht door de dataset op te splitsen in spitsuren en niet-spitsuren. De tijdsfactor is op deze manier onderzocht, omdat het niet als onafhankelijke variabele kon worden gebruikt. Dit is omdat aan elke weg slechts één snelheid kan worden toegewezen in de digitale kaart.

Het neurale netwerk kan als voorspellingsmethode worden gebruikt door relaties tussen de onafhankelijke variabelen (snelheidslimiet, regio, rijstrookbreedte, etc.) en de afhankelijke variabele (snelheid) te vinden. In totaal werden er 21 onafhankelijke variabelen gebruikt om verbanden te vinden tussen de wegfactoren en snelheden in de datasets. Dit wordt het trainingsproces genoemd. Nadat het model is getraind, kan de snelheid van elke weg op de kaart individueel worden voorspeld. In totaal zijn er 21 verschillende neurale netwerkmodellen ontwikkeld. Deze modellen verschillen in het type training data (point- or trip-based data), verliesfunctie (MSE_{LF}, $MAPE_{LF}$, MAE_{LF}, $sMAPE_{LF}$ and $sMdAPE_{LF}$) en afhankelijke variabele (snelheid, logsnelheid en tempo (1/snelheid)). Tijdens het afstemmen van de hyperparameters van de verschillende modellen, werd er gevonden dat het beste aantal neuronen en verborgen lagen tussen de modellen verschillen. Dit hangt af of de afhankelijke variabele tempo of verliesfunctie $sMdAPE_{LF}$ was gebruikt.

Nadat de beste set hyperparameters was gevonden voor elk model, werden 21 verschillende neurale netwerk modellen getraind op oude en nieuwe data. De oude data was verzameld en voorverwerkt door den Heijer, met een frequentie van 5 minuten. De nieuwe data werd verzameld en voorverwerkt tijdens dit onderzoek, met een frequentie van 2 minuten. De datasets werden verkregen van verschillende klanten die met vrachtwagens in de Benelux of Nederland rijden. Het werd verwacht dat met de nieuwe data set de kwaliteit van de trip-based data zou verbeteren, en als gevolg daarvan een verbetering van de nauwkeurigheid van de snelheid en reistijdvoorspellingen van de neurale netwerken.

Uit de resultaten kan worden geconcludeerd dat de random forest models, van den Heijer, de beste reistijdvoorspellingsnauwkeurigheid. De nauwkeurigheid van de reistijdvoorspelling zijn $sMdAPE_{TT}$ 13,8% voor de oude data en $sMdAPE_{TT}$ 12,4% voor de nieuwe data. Dit betekent dat geen van de neurale netwerken de $sMdAPE_{TT}$ van de modellen van den Heijer kon verbeteren. Dit betekent dat de neurale netwerk niet in staat is om zo van de snelheden te leren en te voorspellen, dat de nauwkeurigheid van de reistijdvoorspellingen werd verbeterd. Verders werden mogelijke relaties tussen de voorspellings nauwkeurigheid van de snelheden en reistijden onderzocht. Het bleek dat de nauwkeurigheid van de snelheidsvoorspelling in MSE_{Speed} het meest gerelateerd was aan de nauwkeurigheid van de reistijdvoorspelling in $sMdAPE_{TT}$. Het model met de laagste MSE_{Speed} had de laagste $sMdAPE_{TT}$ voor de nieuwe data, maar dit was niet waar voor de oude data. Daarom kan de MSE_{Speed} worden gebruikt als een eerste schatting van de $sMdAPE_{TT}$ en niet om de beste $sMdAPE_{TT}$ te vinden.

Er werd verwacht dat door het verhogen van de datafrequentie, van 5 naar 2 minuten, een relatief hogere nauwkeurigheid voor de reistijdvoorspellingen zou worden bereikt, vergeleken met de modellen van den Heijer. Uit de resultaten was het onduidelijk of een hogere frequentie van 2 minuten, vergeleken met 5 minuten, de nauwkeurigheid van de reistijdvoorspellingen verbeterd. Daarom wordt aanbevolen de invloed van de datafrequentie op een zelfde datast te onderzoeken. Verders werd de tijdsfactor nog onderzocht door de nieuwe dataset op te splitsen in een spits- en niet-spitsuur dataset. De resultaten toonden aan dat de nauwkeurigheid van de reistijdvoorspellingen nog verder konden worden verbeterd door de tijdsfactor mee te nemen in het model.

Contents

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AVI	Automatic Vehicle Identification
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
BGD	Batch Gradient Descent
BPNN	Back Propagation Neural Network
CPU	Central Processing Unit
CTM	Cell Transmission Model
CV	Cross-Validation
DMI	Distance Measuring Instrument
DOD	Department of Defense
ELU	Exponential Linear Unit
ER	Error Rate
EU	European Union
GB	Gradient Boosting
GBM	Gradient Boosting Methods
GLM	Generalized Linear Model
GMRAE	Geometric Mean Relative Absolute Error
GPR	Generalized Linear Model
GPS	Global Positioning System
GPU	Graphics Processing Unit
HNR	Highway Node Routing
IQR	Inter-Quartile Range
ITS	Intelligent Transport System
KF	Kalman Filtering
k-NN	k-Nearest Neighbors
lr	Learning Rate
LR	Linear Regression
LS	Least Squares
LSTM	Long Short-Term Network
LSTM-DNN	Long Short-Term Deep Neural Network
MAE	Mean Absolute Error
MAE_{LF}	Loss Function Mean Absolute Error
MAE_{Speed}	Mean Absolute Error for evaluation Speeds
MAPE	Mean Absolute Percentage Error
$MAPE_{LF}$	Loss Function Mean Absolute Percentage Error
$MAPE_{Speed}$	Mean Absolute Percentage Error for evaluation Speeds
MARE	Mean Absolute Relative Error
MBGD	Mini-Batch Gradient Descent
MdAPE	Median Absolute Percentage Error
MdRAE	Median Relative Absolute Error
ME	Mean Error
Mk-NN	Multilevel k-Nearest Neighbors
ML	Machine Learning
MLR	Multiple Linear Regression
MPE	Mean Percentage Error
MRE	Mean Relative Error

MSE	Mean Squared Error
MSE_{LF}	Loss Function Mean Squared Error
MSE_{Speed}	Mean Squared Error for evaluation Speeds
MBGD	Stochastic Gradient Descent
NN	Neural Network
ReLU	Rectified Linear Unit
RF	Random Forest
RFID	Radio Frequency Identification
RFNN	Random Forest k-Nearest Neighbors
RME	Relative Mean Error
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RSS	Residual Sum of Squares
SARIMA	Seasonal Autoregressive Integrated Moving Average
SELU	Scaled Exponential Linear Unit
SES	Simple Exponential Smoothing
SGD	Stochastic Gradient Descent
sMAPE	Symmetric Mean Absolute Percentage Error
$sMAPE_{LF}$	Loss Function Symmetric Mean Absolute Percentage Error
$sMAPE_{Speed}$	Symmetric Mean Absolute Percentage Error for evaluation Speeds
sMdAPE	Symmetric Median Absolute Percentage Error
$sMdAPE_{LF}$	Loss Function Symmetric Median Absolute Percentage Error
$sMdAPE_{Speed}$	Symmetric Median Absolute Percentage Error for evaluation Speeds
$sMdAPE_{TT}$	Symmetric Median Absolute Percentage Error for evaluation Travel Times
$sMdPE_{TT}$	Symmetric Median Percentage Error for evaluation Travel Times
sPE_{TT}	Symmetric Percentage Error for evaluation Travel Times
SSNN	State-Space Neural Network
SVM	Support Vector Machine
SVR	Support Vector Regression
TLS	Total Least Squares
VRP	Vehicle Routing Problem

1

INTRODUCTION

1.1. BACKGROUND TO ORTEC

ORTEC is one of the world's leaders in optimization software and analytics solutions. ORTEC makes operations more efficient, more predictable and more effective. The core activities of ORTEC are developing advanced planning and scheduling software, providing advanced analytics and optimization solutions (consultancy) and developing optimization software for real-time data like in sports. This research study is conducted at ORTEC Routing and is one of the domains of the planning and scheduling software. ORTEC Routing aims to develop software that optimizes route planning in order to reduce transportation cost.

1.2. PROBLEM STATEMENT

Route planning is a major research focus at ORTEC. To optimize the route planning, it is important that the travel time-distance calculations come close to reality. If there is a too tight schedule, then this schedule would lead to pressure to the driver, late fees and reputation damage, but a too loose schedule would result in wasted capacity.

To calculate the shortest route, detailed map data provided by the company *HERE Technologies* is used. This map data includes road networks of different countries and many properties of each road such as lane width, speed limit, whether it is a bridge, etc. Currently, the travel time prediction of a calculated route for a vehicle depends on assigned speeds (speed profile) to 20 different road type. These are based on the experience of the implementation consultant (customer). The problems that can be identified with the current planning and scheduling process are as follows:

1. Currently, roads in the map are assigned to one of the 20 different road types. After this, an average speed is assigned to each road type by the implementation consultant. A heuristic is used to assign each road in the map to a road type, but this heuristic is hard to understand, since it is unclear what each road type represent. Also, there is never thorough research done whether the classification of the 20 road types is a good approach. Experience has shown that the 20 road types are not a good division of the roads and seems to be too coarse[1]. Therefore, a better classification of the roads is needed by taking into account factors that influence the average driven speed on the roads.
2. The average speed that is assigned to each road type for a specific vehicle is hard to estimate by the implementation consultant. Therefore, it would be more accurate to use data sources that contain the vehicle's driven speed on many roads.

The consequence of the two mentioned problems above is that some customer experience two shortcomings with the current planning and scheduling process: inaccurate travel time predictions and bad calculated routes. These two shortcomings can be blamed to the inaccurate assignment of the travel speeds to the roads. This is due to the bad classification of the roads and the assigned speeds by the customer as explained above. By improving the classification and assigned speeds, the two shortcomings faced by the customers can be mitigated.

An easy and obvious option to improve the travel time prediction would be to use available routing software such as *Google maps*, *Bing maps*, etc., which can provide accurate travel time predictions. However, these routing softwares cannot be used by ORTEC, because they do not support many-to-many queries (shortest paths between all sources s and targets t), which is needed to solve the vehicle routing problem (VRP). Also, the computational time of these routing software is high and undesired, since the customers have a limited amount of time for the planning and scheduling process. Lastly, these routing softwares do only route calculations for cars, while route calculations for trucks are desired due to the majority of customers that operate with trucks.

At ORTEC, a first attempt was made to improve the travel time predictions based on GPS data and machine learning (ML) algorithms by den Heijer[1]. GPS data is used, since it contains information about the driven speed of the vehicles at many roads and plenty of GPS data can be obtained from the customer. During the research of den Heijer, GPS data from one customer, who is active in the Benelux and operates with trucks, could be obtained and was used. Two ML algorithms, linear regression and random forest, were used to handle the big data sets and to find patterns between multiple road properties (independent variables) and the driven speed (dependent variable). By finding these patterns, the assigned speeds to the roads in the map can be predicted more accurately and come closer to reality. As a result, the travel time prediction accuracy can be improved.

This first attempt showed promising results. Den Heijer's best speed prediction model, based on random forest, improved the $sMdAPE_{TT}$ (symmetrical median absolute percentage error of the travel time) from 18.4% (current model) to 13.8%. The predicted speeds from this random forest model have already been implemented in the map of halve of the customers. However, den Heijer concluded that the travel time predictions may be improved further based on the shortcomings of the used methods as described below.

1. The linear regression model can only model linear relationships between the independent and dependent variables. Many independent variables are not linear related to the speed, which limits the **speed prediction accuracy**. This can also be concluded from the research of den Heijer [1]. It was concluded that the travel time prediction accuracy of the random forest models outperformed the linear regression models due to a higher speed prediction accuracy of the random forest models.
2. Random forest is a more accurate prediction method than linear regression, since it allows to model non-linearities. However, random forest models are bad at predicting new data that differs from the trained data. This is the case when the model learns from data that is based on combinations of multiple roads (trip-based data) and predicts the speed of single roads in the map. Therefore, the random forest can only learn from point-based data. The linear regression model is able to learn from **trip-based data**, however it is unclear whether a more accurate prediction method than linear regression is able to outperform the random forest model with trip-based data.

1.3. RESEARCH OBJECTIVE

The aim of the research is to develop a new speed prediction model for trucks that outperforms the travel time prediction accuracy of the current speed prediction models for a given road network. These current speed prediction models are the models developed by den Heijer [1] based on historical GPS data.

In order to achieve the research objective, main and sub research questions were formulated to provide guidance during the research. The main research question that will be answered in this research assignment is as following:

Can a new speed prediction model be developed for trucks that outperforms the travel time prediction accuracy of the current speed prediction models for a given road network?

The sub research questions that will support the answering of the main research question are the following:

1. How are travel times currently estimated at ORTEC and what are the shortcomings?
2. Which related literature and researches are available, including available travel time prediction methods and influential travel time factors?
3. Which prediction method is most suitable to predict the speeds?

4. How can a prediction method be developed for this problem?
5. How does the new model compare to the current models?

1.4. SCOPE

The scope of this research is described by the following:

- For this research, there is access to two GPS data sets. These data sets are obtained from two different customers of ORTEC. Both data sets will be used for this research. The first GPS data set has already been collected and preprocessed by den Heijer and has a data frequency of 5 minutes. The second GPS data set, that will be collected and preprocessed during this research, has a data frequency of 2 minutes.
- The GPS data that is obtained, comes from two customers of ORTEC that operate in the Benelux and the Netherlands. This means that the given road network for this problem is limited to the Benelux and the Netherlands.
- The collected GPS data is obtained from customers that operate with trucks. This means that this research will focus on improving the travel time predictions for trucks.
- Real-time information that might be available during route planning will not be included such as non-recurrent congestions, road accidents etc. This is because the routing software is used offline by the customer. This means that ORTEC is not able to update the speeds in the map frequently to improve the travel times throughout the day.
- Furthermore, the GPS data, that will be obtained from ORTEC's customers, is collected during their working hours. These working hours are mainly between 5:00 and 16:00 during the midweek. This means that the travel time predictions are applicable for these working hours.
- The travel times will be predicted by ORTEC's routing software. It calculates the shortest path from the first to the last GPS point for each travel, including the travel time. Therefore, the calculation of the shortest path and travel time is considered to be out of scope for this research.

1.5. REPORT OUTLINE

After the introduction, a problem description will be provided in chapter 2. First, the current process for the travel time predictions at ORTEC is analyzed. Secondly, the improved process proposed by den Heijer[1] is analyzed. After this, a comparison of the travel time prediction accuracy, between these two processes, will be shown. Lastly, the focus of this research will be discussed. This is based on the recommendations by den Heijer to improve the travel time prediction accuracy further.

In chapter 3, the prediction method and methodology for this research will be discussed. First, available travel time prediction methods from literature are researched. This will be done to find out whether another method, than developed by den Heijer, can be used for this research problem. After this, a method trade-off will be performed between suitable prediction methods. Subsequently, an literature review will be conducted for the neural network. Lastly, the methodology for this research will be discussed.

In chapter 4, the data used in this research will be discussed. First, a description will be provided of the used GPS data. Secondly, the data cleaning steps and map matching process will be discussed. Thirdly, the feature extraction of the GPS data will be discussed. This includes the process of deriving values (features) that are informative to facilitate the learning and predicting process of the model. Lastly, the data will be sampled into a training and test data set.

In chapter 5, the model design will be discussed. First, the model objective will be determined. Secondly, the independent (input) and dependent (output) variables of the model are provided. Subsequently, model design choices will be made. After this, the implementation of the model will be discussed. Lastly, the hyper-parameters of the neural network model will be tuned to obtain the best model performance.

In chapter 6, the experiments and results are discussed. First, an overview of the experiments will be provided. Then, the speed and travel time prediction accuracy of the new developed and current models will be shown. Lastly, the results of the influence of the temporal factor, on the travel time prediction accuracy, will be shown and discussed.

2

PROBLEM DESCRIPTION

In this chapter, the problem of this research study will be described. Firstly, the current steps for route calculation are discussed. After this, the current steps that can be improved to improve the travel time prediction accuracy are identified. Then a process developed by den Heijer[1] is explained, which uses GPS data to improve the travel time predictions. After this, the travel time prediction accuracies of the Current and den Heijer's models are provided and compared. Lastly, the motivation for this research will be discussed, to develop a new model that improve the travel time prediction accuracy even further.

2.1. DESCRIPTION OF THE CURRENT PROCESS

In Figure 2.1, an illustration of the currently followed steps, to calculate a route and subsequently the travel time, is shown. The process consists of four steps where the two first steps are executed at ORTEC and the last two at the customer itself. Step 1 to 3 are executed once and basically form the foundation for the route calculation in step 4. This last step is repeated as many times as needed by the customer to calculate the shortest route from point A to B. After this, the predicted travel time is used as input for the route planning. A description of each step can be found below[1]:

- **Step 1. Get map data:** To calculate routes, a map of the road network is required to know all existing roads which can be used by the route optimizer. This map is obtained from the company *HERE Technologies* and also contains properties of each road segment such as length, advisory speed, speed limit, speed bumps, etc. This map data is the underlying foundation for the route calculation.
- **Step 2. Convert, add types:** After access is permitted to the map data by *HERE*, this raw data can be adjusted to a format that is understandable by ORTEC's logistics planning software. After this, a simple heuristic is used to assign each road segment to a 'road type'. There are 20 'road types' to which each road segment can be assigned and is based on a few properties of the road segment. These properties are obtained from the map data. This means that all road segments from the entire road network have a speed that corresponds to the speed of the assigned 'road type'.
- **Step 3. Configure speeds:** Now all road segments are assigned to a 'road type', the implementation consultant can easily adjust the speed of all road segments that are categorized by the 20 'road types'. The assignment of the speeds to the 'road types' is based on experience of the implementation consultant.
- **Step 4. Calculate route:** The last step is performed by the customer and can be executed after the previous steps are completed. Here, the ORTEC's logistics planning software calculates the optimal route based on an improved Dijkstra algorithm called Highway Node Routing [2]. This algorithm preserves the same optimal route as Dijkstra's algorithm, but with an improved computational time.

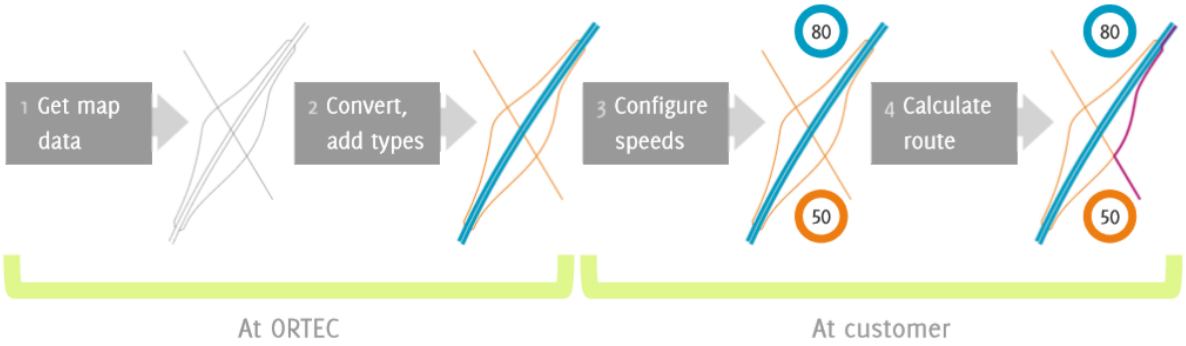

Figure 2.1: Current process to calculate a route[1].

The Highway Node Routing algorithm can be applied statically and dynamically to incorporate for example congestions by updating individual edge weights. ORTEC's logistics planning software uses the static version, since this results in considerably faster computation time, which is beneficial for the customer. This means that all edges are static and cannot be changed to incorporate time-dependent variables such as congestions, road incidents, etc.

Other routing softwares such as *Google maps*, *Bing maps*, etc. provide accurate route calculations and travel time predictions. This would be an easy and obvious solution to improve the travel time predictions by ORTEC. However, they are not used by ORTEC because:

- The routing software does not support **many-to-many queries**. This is needed to know all routes between all origins s and destinations t and is required to solve the VRP.
- The **computational time** of the routing software is significantly higher. For example, the calculation of 2314 trips in *Google maps*, with an average travel time of 18 minutes in the city Baton Rouge (US), takes about 15 minutes[3]. ORTEC's customer have many-to-many query sizes up to 2800x2800, which means that this would lead to unacceptable computational times.
- The routing software does only calculate routes and travel times for cars and not other **vehicle types**. Since many customers of ORTEC operate with trucks, the route calculation and travel time prediction may not be accurate. This is due to a different driving behavior of trucks compared to cars. For example, trucks have a different speed limit than cars on highways due to regulations. Also, the time to pass a roundabout is higher due to the relatively large length and high weight, which makes insertion more difficult.

SHORTCOMINGS CURRENT PREDICTION MODEL

The current planning and scheduling process designed by ORTEC has, as described in section 1.2, two problems. The first is rough classification of the roads (20 road types), and the second is inaccurate assigned speeds by the customer. To mitigate these two problems, the steps shown in Figure 2.1 have to be revised. This can be done by the following question: *how can the average speed assigned to the road segments in the road network be optimized to improve the travel time predictions?* After revision of the steps by den Heijer, it was concluded that an improvement can be made at step 2 and 3. This means that the classification of the 20 'road types' are too rough and the assigned speeds are inaccurate, since it is based on experience. In the following section, the developed process by den Heijer's, to improve the travel time prediction accuracy, is discussed.

2.2. DESCRIPTION OF DEN HEIJER'S PROCESS

As explained in section 2.1, the accuracy of the travel time predictions can be improved by improving step 2 and 3 as illustrated in Figure 2.1. Den Heijer developed a model, that replaces step 2 and 3. This model learns from driven speeds obtained from realized GPS data. These speeds are related to the road properties, obtained from the map data, to predict the speeds in the map more accurately. These road properties can be found in Appendix B. The predicted speeds from this random forest have already been implemented in the map of halve of the customers.

2.2.1. GPS DATA

There are different ways to obtain information about the vehicle's location and speed and are among others: loop detectors, vehicle identification devices or floating car observers[4]. However, these methods are limited to one specific location and/or are too expensive. Therefore, GPS equipment is a great solution, because it is:

- **Cheap:** Access to GPS is free for public and purchase cost of the GPS equipment are relatively low. This makes the collection of location and speed data of vehicles through GPS a cheap solution.
- **Easy to implement:** GPS equipment can be installed in all vehicles and is relatively easy to implement. This makes the threshold for the installation of GPS equipment lower and results in more data collection.
- **Complete:** While data collection through other equipment is restricted to specific locations, GPS equipment is flexible. This allows data collection at all locations in a road network. In this way, factors that influence the driving speed can be identified and can be used to predict average driven speeds on roads in the entire road network.
- **Available:** Since the installation of GPS equipment is relatively easy and has low cost, many companies are currently using GPS, to track there fleet of vehicles. Therefore, plenty of GPS data is available. This can be used to learn from the driven routes and to improve the travel time predictions.

Besides the many advantages of using GPS, there is also a disadvantage, which is a poor accuracy. On average, a GPS data point is obtained within 10 meters accuracy of the real location where the GPS location is recorded. The accuracy is even worse when the GPS signal is blocked by for example tunnels or reflected in urban areas and mountainous regions. This adds some inaccuracies to the prediction model. However, these inaccuracies can be largely mitigated by applying map matching techniques. This will be elaborated in the next chapter.

2.2.2. SPEED PREDICTION MODEL

To improve the travel time predictions at ORTEC, den Heijer decided to use available GPS data from a customer. This contains a lot of useful information about the driven speeds and travel times throughout the entire road network. A first attempt to improve the travel time predictions for trucks, is described in [1]. This showed promising results and proved that travel time predictions can be considerably improved by learning from GPS data.

In Figure 2.2, an illustration of the process of the developed speed prediction model by den Heijer is shown. This process replaces step 2 and 3 in Figure 2.1 and consists of 2 steps. First the speed prediction model needs to be trained, which means that the model tries to find correlations between the independent and dependent variables:

- **Independent variables:** The independent variables are also called explanatory variables, features, indicators, input and predictor variables. As independent variables, den Heijer used the road properties of each road such as speed limit, traffic lights, speed bumps, etc. to predict the speed. The road properties of each road in the road network were obtained from the map data, supplied by *HERE*.
- **Dependent variable:** The dependent variable is also known as target, predicted or output variable. The dependent variable of den Heijer is the speed and is obtained from the collected GPS data.

After the model is trained successfully, the model can be used as prediction model to predict the speed for each road in the map. For the map of the Netherlands, this is 3,304,031 roads. Then the predicted speeds are

assigned to each road in the map, after which the shortest paths and travel times from point A to B are calculated. The calculated travel times are compared with the actual travel times, using the symmetric median absolute percentage error ($sMdAPE_{TT}$).

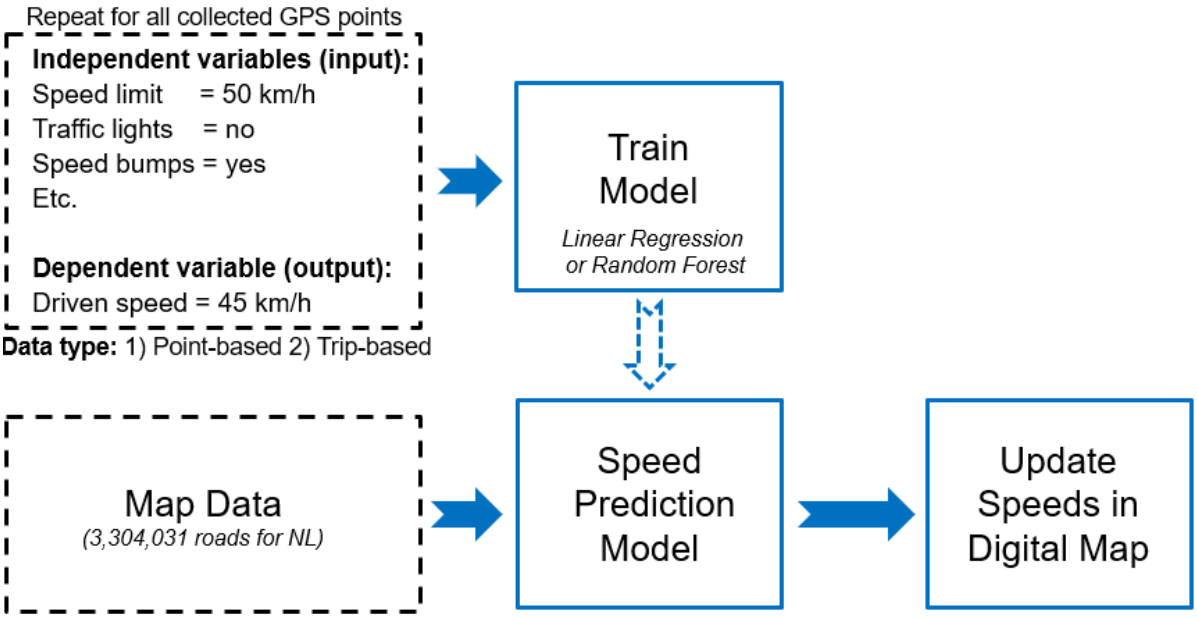

Figure 2.2: First the speed prediction model is trained based on independent variables and dependent variables. After this, the average speed for each road in the map based is predicted based on its road properties (independent variables). This speed prediction model replaces step 2 and 3 in Figure 2.1.

SHORTCOMINGS DEN HEIJER'S SPEED PREDICTION MODELS

Den Heijer identified some shortcomings with the linear regression and random forest models. By overcoming these shortcomings, the travel time prediction accuracy might be further improved. The identified shortcomings are as following:

1. The linear regression model can only model linear relationships between the independent and dependent variables. Many independent variables are not linear related to the speed, which limits the **speed prediction accuracy**. This can also be concluded from the research of den Heijer [1]. It was concluded that the travel time prediction accuracy of the random forest models outperformed the linear regression models due to a higher speed prediction accuracy of the random forest models.

2. Random forest is a more accurate prediction method than linear regression, since it allows to model non-linearities. However, random forest models are bad at predicting new data that differs from the trained data. This is the case when the model learns from data that is based on combinations of multiple roads (trip-based data) and predicts the speed of single roads in the map. Therefore, the random forest can only learn from point-based data. The linear regression model is able to learn from **trip-based data**, however it is unclear whether a more accurate prediction method than linear regression is able to outperform the random forest model with trip-based data.

   Bad performance random forest with trip-based data
   Most independent variables (properties) of single roads are Boolean, such as speed bumps, traffic lights, ramp, paved and priority road. 1 means that the road has this property and 0 not. When the model is first trained on a combination of multiple roads, the average of the properties is taken. The value of the independent variables is often somewhere between 0 and 1 instead of being a Boolean. A simple example of a decision tree that is trained on a combination of multiple roads is shown in Figure 2.3. Many of these kind of trees are used by the random forest prediction method. After the decision trees are trained, the speed for a single road in the map is predicted based on the Boolean value of X. When X=0, then the average speed of leaf node Speeds($0 \geq X < 0.25$) is taken and when X=1, the average speed of leaf node Speeds($0.75 \geq X \geq 1$) is taken. However, this means that speeds that are not equal to

a Boolean value are also taken into account to predict the speed for X=0 and X=1. This causes a difference in the desired output for Speeds(X=0) and Speeds(X=1). Because this difference is present at each predicted road in the map, the final difference is large and the travel time prediction accuracy will be low.

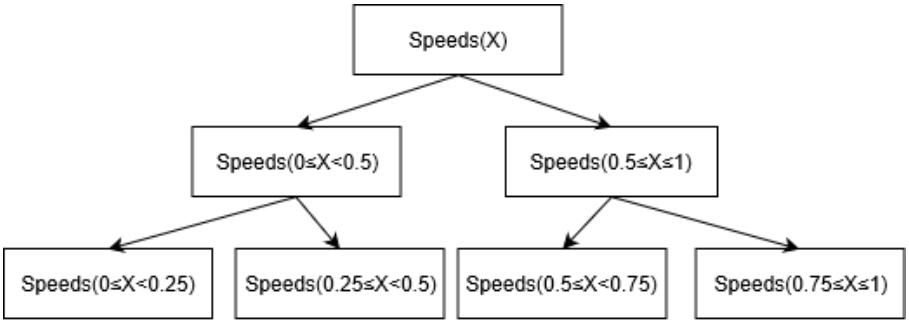

Figure 2.3: Example of decision tree that predicts the speed based on independent variable X that ranges between 0 and 1.

2.3. COMPARISON CURRENT AND DEN HEIJER'S MODELS

In this section, a comparison between the current and den Heijer' models is provided. Den Heijer developed 8 different models and are based on two different ML prediction methods, linear regression and random forest. These prediction methods are used to find relationships between the input variables and the average speed on the roads in the map. These prediction methods were applied to two different types of training data, point- and trip-based data. The trip-based data is derived from the point-based data and needs an additional preprocessing step. These types of data are visualized in Figure 2.4 and Figure 2.5, respectively. Additionally, the prediction methods were trained on four different types of dependent variables: *speed, logspeed, pace and time*. These were after prediction converted back to speed to be able to calculate the travel time with a shortest path algorithm. The reasoning for using point- and trip-based data as well as different dependent variables is provided below.

2.3.1. POINT-BASED AND TRIP-BASED DATA

POINT-BASED DATA

Point-based data is the simplest training data. This data is based on the road properties and the driven speed of the locations where the GPS points were recorded. In Figure 2.4, an example is shown of three recorded GPS points where the driven speed, obtained from the GPS data, is indicated between brackets. After the GPS points are matched to a road in the digital map, the properties of the roads can be looked up in the map data. A simple example with two road properties, speed limit and speed bumps, is shown in Table 2.1. In reality, many more road properties are available and can be used to find relationships between the road properties (independent variables) and the speed (dependent variable).

TRIP-BASED DATA

When point-based data is used, only the data from the roads of the GPS locations are used. This means that information from only a few roads of the travel are used to train the model. All other roads, where no GPS data is collected, are therefore not used to train the model. As a consequence, travel information of a very large part of the travel is not used to finally predict the speeds. To still include these roads, where no GPS data is recorded, den Heijer introduced trip-based data. In Figure 2.5, an example is shown of trip-based data. In this figure two trips are shown, trip 1-2 and trip 2-3, which are both between two GPS points. Since the trip-based data uses all roads between two consecutive GPS points, the variables have to be rewritten to a form that can be easily trained on. This is an additional preprocessing step. This is done by taking the average of the road properties of all roads that are included the trip. The average of each road property is based on the relative length that the property occurs in the trip.

Preprocessing Step Trip-Based Data

An example of the preprocessing step for trip-based data is shown in Table 2.1. In this table, the average is calculated for the speed limit and speed bumps for trip 1-2 and trip 2-3. For simplicity, trip 1-2 is divided into two roads and trip 2-3 into three roads, which are all straight roads. For trip 1-2, the first road is 85% and

the second road 15 % of the total trip length. For trip 2-3, the first road is 75%, the second road 20% and the third road 5% of the total trip length. Subsequently, these percentages are used to determine the values of the independent variables. For trip 1-2, the first road has a speed limit of 30 km/h and the second road 50 km/h. This results in a speed limit for trip 1-2 of 0.85*30 + 0.15*50 = 33 km/h. This approach can be applied to all independent variables. The dependent variable is the average speed of the trip, which is $\frac{\text{trip length}}{\text{trip time}}$.

Table 2.1: Overview of how the independent variable values are used and calculated for point-based and trip-based data.

	Independent Variables		**Dependent Variable**
Point/Trip	**Speed Limit [km/h]**	**Speed Bumps [y/n (1/0)]**	**Driven Speed [km/h]**
Point 1	30	1	28
Point 2	50	0	52
Point 3	30	1	33
Trip 1-2	33 (0.85*30 + 0.15*50)	0.85 (0.85*1 + 0.15*0)	23
Trip 2-3	45 (0.75*50 + 0.20*30 + 0.05*30)	0.25 (0.75*0 + 0.20*1 + 0.05*1)	25

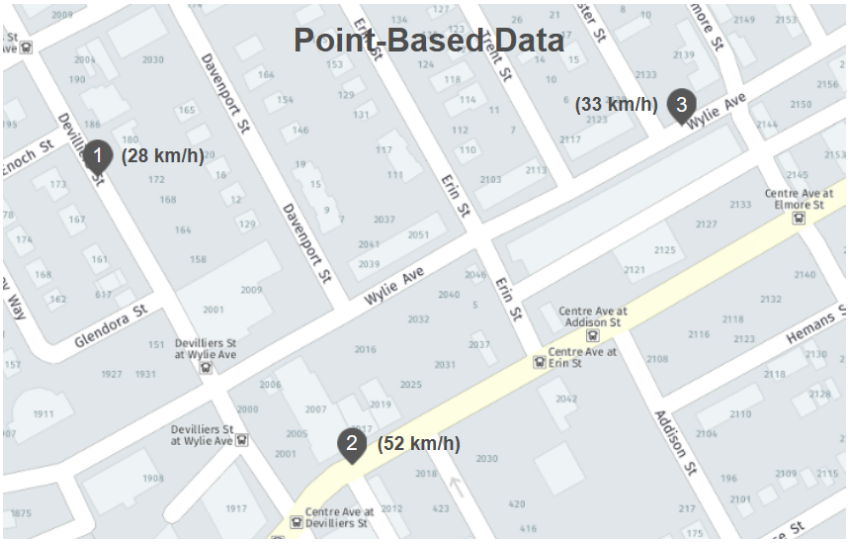

Figure 2.4: Example of GPS data points used to train the speeds with point-based data. Also the driven speed in km/h is included to each GPS point.

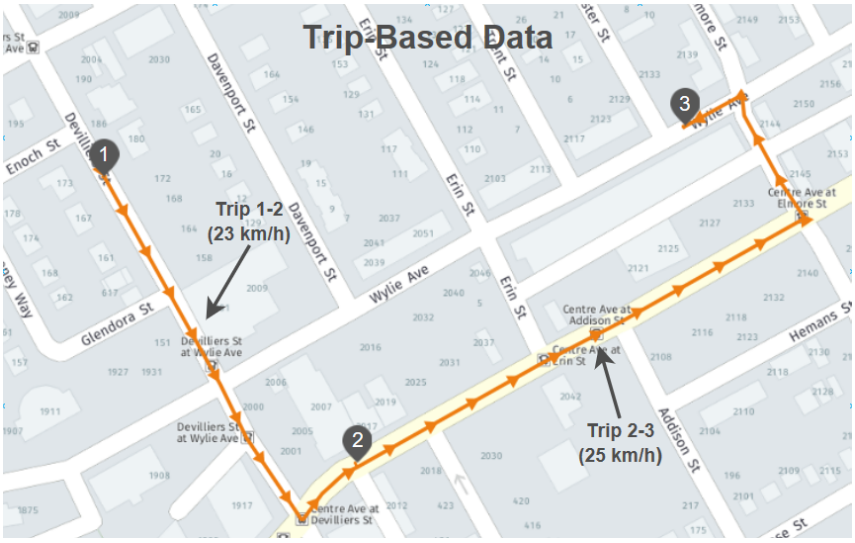

Figure 2.5: Example of trips between the GPS data points used to train the speeds with trip-based data. Also the average speed in km/h is included to each trip.

2.3.2. DEPENDENT VARIABLES

It was decided to train on different dependent variables. den Heijer concluded that training on the speed, is not necessarily the best choice to improve the travel time prediction accuracy. The reason to choose the following dependent variables to train on are as following[1]:

Dependent variables for point-based data

- **Speed:** An obvious choice is to train on the dependent variable speed. This is because the collected GPS data contains the driven speed and the speeds assigned to each road in the map need to be improved. Therefore speed is a logical choice to train the prediction model on.
- **Logspeed:** Linear regression and random forest both use the squared error to optimize the model. This is not always ideal, because it results in a model that better fits to larger actual values and an underestimation of the actual value. A better fit to larger actual values can be explained by Table 2.2, where an absolute difference for a low and high actual speed is equal to 10 km/h. Intuitively, 10 km/h difference on the low actual speed (20 km/h) is a much larger difference (100%) than on the high actual speed of 80 km/h (12.5%). This results in a prediction model that better fits to larger actual values.

Table 2.2: Example to show that squared errors favor to fit the model to more accurate predictions of large actual speeds.

Actual Speed	Predicted Speed	Squared Error
10 km/h	20 km/h	$(20-10)^2 = 100$
80 km/h	90 km/h	$(90-80)^2 = 100$

Additionally, a second example shows that the same low and high absolute predicted speed compared to the actual speed give the same error. However, when calculating the travel time for a 10 km road, the prediction of 5 km/h results in a travel time of 2 hours, while a 15 km/h prediction results in 40 minutes (Table 2.3). The actual travel time would be 1 hour, which means that the prediction of 15 km/h is closer, while 5 km/h and 10 km/h have the same squared error. The overestimation is preferred, while both have the same error. This results in a underestimation of the actual speed. To avoid both problems, the logarithmic speed can be used. This moves the speeds to a relative space, which agrees better with the research goal to optimize the travel time predictions. Moving to the logarithmic speed gives the same error for low and large actual speeds, as well as under- and overestimations. Eventually, whether training on speed or logspeed is preferred, should be tested.

Table 2.3: Example to show that the same absolute under- and overestimation of the actual value give same results.

Actual Speed	Predicted Speed	Squared Error
10 km/h	5 km/h	$(5-10)^2 = 25$
10 km/h	15 km/h	$(15-10)^2 = 25$

Dependent variables for trip-based data

The challenge of using trip-based data is: how can a sequence of multiple roads be written to a form that can be used by the model, which is in this case linear regression. Some choices used by den Heijer, based on different dependent variables, are discussed below:

- **Time:** Training on time between two GPS points is an obvious choice, since the aim of the research is to improve the travel time prediction accuracy. To train on the trip time and to predict the average time for each road in the map, the independent variables have to be weighed in some way. This can be done by multiplying each coefficient by the distance that the corresponding variable occurs during the trip. A simple example is shown in Equation 2.3.1, with one Boolean variable. This is 'IsHighway' and is multiplied by the length of occurrence in the trip.

$$[\text{trip time}] = \beta_1 \cdot [\text{meters on IsHighway}] + \beta_2 \cdot [\text{meters not on IsHighway}] \tag{2.3.1}$$

This multiplication by distance can also be applied to categories, such as countries shown in Equation 2.3.2:

$$[\text{trip time}] = \beta_1 \cdot [\text{m NL}] + \beta_2 \cdot [\text{m LU}] + \beta_3 \cdot [\text{m BE}] \tag{2.3.2}$$

Independent variables that occur at a specific site (point-data), such as traffic lights, can also be added. This is shown in Equation 2.3.3, where β_4 represents the average waiting time at a traffic light.

$$[\text{trip time}] = \beta_1 \cdot [\text{m NL}] + \beta_2 \cdot [\text{m LU}] + \beta_3 \cdot [\text{m BE}] + \beta_4 \cdot [\text{number of traffic lights}] \tag{2.3.3}$$

- **Pace:** An alternative way of training the model, instead of using 'meters driven', is using 'fraction of meters' with respect to the full trip. The conversion to fraction of meters can be simply applied by dividing each independent and dependent variable by 'total meters driven' for all trips. This is shown in Equation 2.3.4. This results in a dependent variable dimension of time/length (1/speed), which is called average *pace*. An advantage of moving to the pace is that the data is normalized, so long trip times are not favored by the squared errors.

$$\begin{aligned}[\text{average pace}] = \beta_1 \cdot [\text{fraction of meters on IsHighway}] + \\ \beta_2 \cdot [\text{fraction of meters not on IsHighway}]\end{aligned} \tag{2.3.4}$$

- **Speed:** By using the pace, the model favors a high pace (low speed) due to the squared errors. It is unknown whether it is preferred that the model favors low or high speeds. Therefore, the speed is also used as dependent variable. This results in a multiplication of the coefficients by the 'fraction in seconds'. However, the time that is spent on each road between two GPS points is not known, therefore the fraction of meters is used. An example is shown in Equation 2.3.5 where all β's represent the speed. In addition, training on the speed is, as with point-based data, an obvious choice. This is because the speeds, assigned to each road in the map, need to be improved.

$$\begin{aligned}[\text{average speed}] = \beta_1 \cdot [\text{fraction of seconds on IsHighway}] + \\ \beta_2 \cdot [\text{fraction of seconds not on IsHighway}]\end{aligned} \tag{2.3.5}$$

- **Logspeed:** Also, when using trip-based data, the model is optimized by using the squared error for linear regression. By optimizing the model with the squared error, larger actual values are favored as well as an underestimation of the actual value. Therefore, also the logspeed, which is basically the same as -logpace, will be used as dependent variable. The coefficients are multiplied by the 'fraction of meters'.

2.3.3. MODEL COMPARISON

In total, 8 speed prediction models were developed by den Heijer and compared to two benchmarks. The first benchmark is the travel time prediction by ORTEC's Current model. The second benchmark is the advanced travel time predictor for trucks from *HERE Technologies*. The models are compared with the $\text{sMdAPE}_{\text{TT}}$ function and is a representation of the error between the predicted and actual travel times. In Figure 2.6, it can be seen that the Current model, which uses 20 different 'road types' and the experience from the implementation consultant, has the worst performance with $\text{sMdAPE}_{\text{TT}}$ 18.4%. All 8 developed models by den Heijer have a better performance than the Current model. The random forest model Point-RF-Logspeed, trained on point-based data and dependent variable logspeed, performs the best with $\text{sMdAPE}_{\text{TT}}$ 13.8%. This speed prediction model even outperforms the predicted travel times by *HERE-truck*, which is seen as an advanced travel time prediction model.

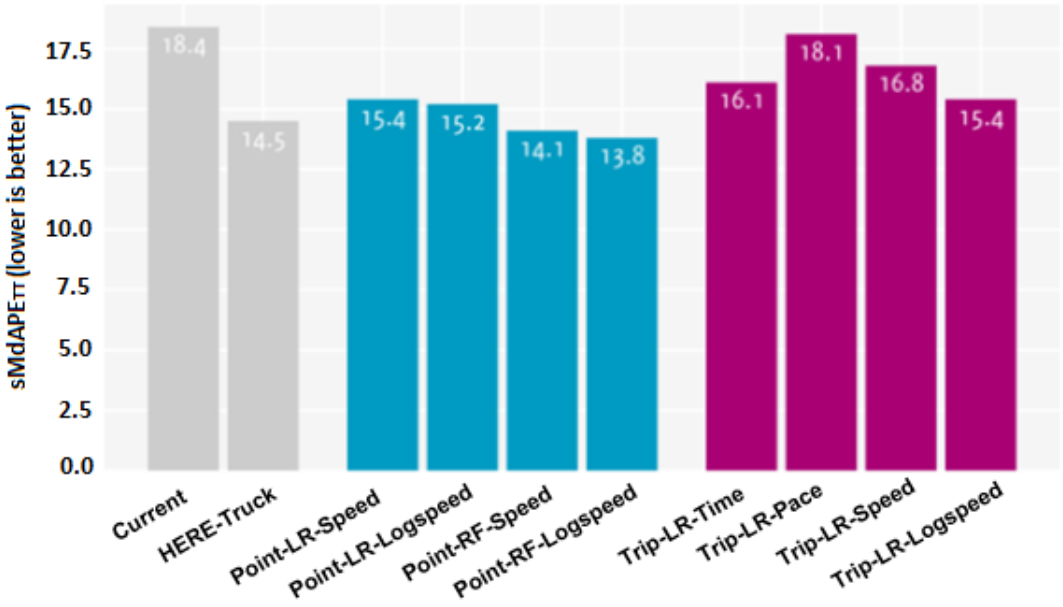

Figure 2.6: Travel time error from different models evaluated by $sMdAPE_{TT}$ in %. The closer to 0 the better the model[1].

In Table 2.4, the Current model and den Heijer's best model are summarized. In the table the input, method, output, type of training data and travel time prediction accuracy are included. Also, the vehicle and road network are added for which the models are compared.

Table 2.4: Comparison between Current model and den Heijer's best model.

	Current Model	**den Heijer's Model**
Input	Speeds based on experience customers	GPS data (5 minutes frequency)
Method	20 Road Types	Random Forest (logspeed)
Output	Speed	Speed
Type of Training Data	None	Point-based data
Travel Time Accuracy ($sMdAPE_{TT}$)	18.4%	13.8%
Vehicle	Truck (chemical packages)	Truck (chemical packages)
Road Network	Benelux	Benelux

2.4. RESEARCH FOCUS

From the research of den Heijer, it was found that the optimization of the travel time predictions is quite a challenging task. This is because first the speed of each road in the digital map has to be predicted in order to improve the travel time predictions. Den Heijer concluded that the travel time prediction accuracy, expressed in $sMdAPE_{TT}$, does not go hand in hand in hand with the speed prediction accuracy expressed in $sMdAPE_{Speed}$[1]. The problem is that the calculation of the travel time is based on multiple roads instead of one road. If the calculation of the travel time would be based on one road, then the predictions of the speeds would be directly correlated to the travel time, since the road length of each road is fixed. This is shown in Equation 2.4.1, where accurate speed predictions, expressed in $sMdAPE_{Speed}$, would directly lead to accurate travel time predictions in $sMdAPE_{TT}$.

$$\text{Travel Time} = \frac{\text{Road Length}}{\text{Travel Speed}} \qquad \text{(One Road)} \qquad (2.4.1)$$

In this research, the goal is to improve the travel time prediction accuracy in $sMdAPE_{TT}$, where a lower $sMdAPE_{TT}$ means a higher accuracy. This research goal can be written in a objective function shown in Equation 2.4.2. In this equation A_t is the actual travel time, P_t the predicted travel time, t the travel number and n

the total number of travels. The predicted travel time P_t for each travel can be calculated by Equation 2.4.3, which is based on multiple roads. In this equation, m indicates the total number of roads that are included to calculate travel time P_t and differs between travels. m is based on the roads that are included in the shortest path, which are found by a shortest path algorithm. To minimize the objective function, P_t has to be predicted as close as possible to A_t, where A_t is obtained from the travel time data.

$$\text{Minimize} \quad \text{sMdAPE}_{\text{TT}} = \text{median}\left(200\% \cdot \left|\frac{A_t - P_t}{A_t + P_t}\right|\right) \qquad t \in 1, ..., n \tag{2.4.2}$$

$$P_t = \left(\frac{\text{Road Length}}{\text{Travel Speed}}\right)_{1t} + \left(\frac{\text{Road Length}}{\text{Travel Speed}}\right)_{2t} + ... + \left(\frac{\text{Road Length}}{\text{Travel Speed}}\right)_{mt} \qquad \text{(Multiple Roads)} \tag{2.4.3}$$

An obvious approach would be to optimize the objective function in Equation 2.4.2 by finding the optimal speeds for each travel in Equation 2.4.3. However, this approach is not suitable for this research problem due to the following reasons:

- The travel time prediction P_t, of each travel, is derived from the shortest path that is found by a shortest path algorithm. Before optimizing Equation 2.4.2, the speeds are not known, which means that Equation 2.4.3 cannot be derived for each travel, since a shortest path cannot be found. Therefore, $\text{sMdAPE}_{\text{TT}}$ in Equation 2.4.2 cannot be optimized based on Equation 2.4.3.
- By minimizing Equation 2.4.2, only the speeds of the roads that are included in the travels are predicted. Therefore, many speeds of other roads in the map are not predicted, which may be needed for other travels in the future. Therefore, the optimization of the objective function in Equation 2.4.2 with Equation 2.4.3 is not suitable for this research.

As a conclusion, an improved travel time prediction accuracy has to be found in a different way. Den Heijer made a first attempt to predict the speeds such that the travel time predictions were improved based on GPS data. This was done by finding relationships between the driven speeds and road properties, after which the speed for each road in the road network was predicted. The random forest model, with output logspeed and trained on point-based data, had the best travel time prediction accuracy. In the chapter, *conclusions and recommendations* of den Heijer's research report[1], it was discussed that the travel time prediction accuracy might be improved even further. The recommendations from den Heijer, to further improve the travel time prediction accuracy, will be focused on in this research. These recommendations are:

- **Prediction method:** The ML methods used by den Heijer had some limitations as explained in section 2.2. The linear regression method can be trained on point- and trip-based data, however it can only model linear relationships. This limits the prediction accuracy of the speeds. The shortcoming of the random forest method is that it cannot be used for trip-based data. When the model is trained on trip-based data, the speeds may be predicted more accurately, such that the travel time prediction accuracy is improved. Therefore, a model with a higher prediction accuracy than linear regression and random forest for point- and trip based data is preferred.
- **Data:** The research study conducted by den Heijer had access to GPS data from trucks with a frequency of 5 minutes. However, the frequency of 5 minutes between two GPS points is rather low and reduces the quality of the trip-based data. Therefore, it is preferred to use GPS data with a higher frequency than 5 minutes. It is expected that a higher data frequency will lead to improved speed and travel time predictions for models that are trained on trip-based data.
- **Influential factors:** Den Heijer did not conduct a literature survey about factors that influence the travel time. Only road properties from the map data, provided by *HERE*, were used to train the model. Including more factors such as part of the day, season or weather might further improve the travel time prediction accuracy.

2.5. CONCLUSION

Currently, the steps that are followed to calculate a route, from which the travel time can be derived, are:

- **Step 1:** Map data of the road network and the properties of each road segment are obtained from *HERE Technologies*.
- **Step 2:** The map data is converted to a format that ORTEC's logistic routing software understands and all road segments are assigned to one of the 20 'road types'.
- **Step 3:** The implementation consultant assigns a speed to each 'road type'.
- **Step 4:** Based on the assigned speed to each 'road type', the shortest route from point A to B is calculated.

From these four steps, den Heijer[1] identified that by improving the speed predictions in step 2 and 3, the travel time prediction accuracy may be improved. This is because of the rough road classification of 20 road types and possibly inaccurate speeds that are assigned to each road type, by the customer. Den Heijer developed eight different models, which are based on linear regression and random forest. The models learn from historical GPS data to make more accurate speed predictions. First, the models try to find relationships between the road properties and the driven speeds, from the roads in the GPS data set. Secondly, the speed for each road in the road network is predicted based on its road properties. The GPS data set that was used by den Heijer, to improve the speed predictions, is obtained from a customer of ORTEC that operate with trucks throughout the Benelux. The GPS data was recorded with a data frequency of 5 minutes.

The eight developed speed prediction models by den Heijer, differ in prediction method (linear regression or random forest), type of training data (point- or trip-based) and dependent variable (speed, logspeed or pace). After learning and predicting the speed for each road in the digital map, the travel times could be predicted and evaluated. The random forest model, trained on point-based data with dependent variable logspeed, showed the largest improvement from $\text{sMdAPE}_{\text{TT}}$ 18.4% (Current model) to 13.8%.

From the research of den Heijer, it was found that the optimization of the travel time predictions is quite a challenging task. This is because first the speed of each road in the digital map has to be predicted in order to improve the travel time predictions. In the chapter, *conclusions and recommendations* of den Heijer's research report[1], it was discussed that the travel time prediction accuracy, of den Heijer's best model, might be improved even further. The recommendations to improve this accuracy, which will also be the research focus, are the following:

- Developing a model with a higher prediction accuracy than linear regression and random forest that can be used for point- and trip-based data.
- Using GPS data with a higher frequency than 5 minutes to improve the quality of trip-based data, which may result in a higher speed and travel time prediction accuracy.
- Finding factors that influence the speed and travel time, which can be included in the model to improve the speed and travel time predictions.

3

Prediction Methods

In this chapter, the prediction method and methodology for this research are discussed. Firstly, travel time prediction methods in literature are examined and summarized in tables. This will be done to find out if there is another method than den Heijer's method that can be applied to this research problem. After this, a method trade-off will be done of the prediction methods that can used for this research problem. Subsequently, an literature review is conducted for the most suitable prediction method, which is the neural network. Lastly, the methodology for this research is described which is based on den Heijer's methodology. This is visualized through a flowchart.

3.1. Available Travel Time Prediction Methods in Literature

In this section, available travel time prediction methods in literature are discussed. This is done to find out if there are other methods that can be used for this research problem than the method developed by den Heijer[1]. In [5], a recent overview of different available travel time prediction methods in literature is provided. In Figure 3.1, an overview of these travel time prediction methods is shown and can be divided into model-based and data-driven methods. The difference between these two classified methods are:

- **Model-based:** Model-based methods are used when a deep understanding of the system or process is known. The method is simplified and is limited to a certain complexity. To find and build an appropriate model can be a time-consuming and expensive process.
- **Data-driven:** Data-driven methods learn from data, where more input data results in better predictions. These methods allow high complexity, while the implementation is relatively simple with low cost. Data-driven methods are able to find relationships between variables without knowing the physical behavior of the system. However, the precision of the data-driven methods is in general less than the model-based methods. M. Bai et al.[5] divided the data-driven methods into parametric and non-parametric. The differences between these two methods are:
  - **Parametric methods** simplify the function to a known form. The parametric methods assume that the data will follow a certain distribution or 'shape' of the data. For example, it is known that a linear line will be fitted to the data if a linear regression model is used.
  - **Non-parametric methods** do not make strong assumptions about the 'shape' of the data. They have an indefinite number of parameters and make fewer assumptions about the data. Non-parametric methods are flexible and have often a higher prediction performance.

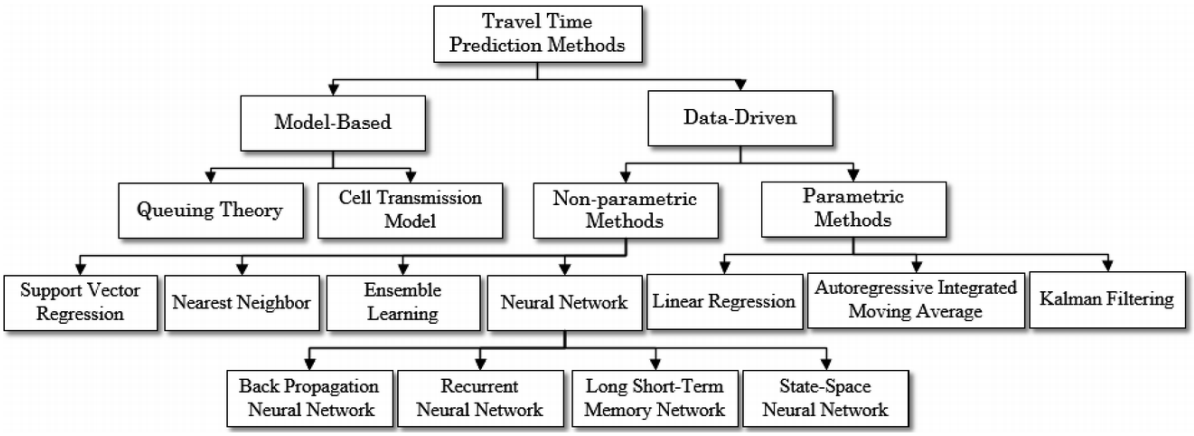

Figure 3.1: Classification of travel time prediction methods[5].

Each travel time prediction method in literature is applied to a different research problem. To assess if one or more methods are suitable for this research, different criteria are used in Table 3.1, Table 3.2 and Table 3.3. The criteria that are used are:

- **Input:** The input of the prediction method can be divided into site-based and vehicle-based data. Site-based data contains information about a specific road segment. Vehicle-based data contains information about different points throughout the whole road network. Examples of site-based data are license plate matching and loop detectors. Mostly, vehicle-based data consists of GPS data. For this research, GPS data will be used as input of the prediction method.
- **Prediction Method:** The prediction method is the method used to predict an output based on a given input. For this research, the method needs to be able to predict the speed based on a given input.
- **Output:** The output is the prediction of the prediction method based on the input. For this research, the travel speed needs to be the output.
- **Accuracy:** The accuracy is important to know how accurate a prediction method is. Different error functions can be used to express the accuracy, where each error function has its own benefit.
- **Scope:** The scope indicates how many roads or routes are predicted with the prediction method. The scope of this research is a road network, which means that different travel time predictions throughout an entire road network have to be improved.
- **Prediction Horizon:** The prediction horizon indicates how far in the future predictions can be made. For this research, there is no prediction horizon since the speeds are predicted only once. This is because the routing software, developed by ORTEC, is used offline by the customers. This means that the speed of each road in the map cannot be changed.

3.1.1. MODEL-BASED METHODS

The model-based methods that are commonly used to predict the travel time are based on the queuing theory and cell transmission model (CTM). These methods are built by traffic parameters such as flow, speed and density, which are obtained from historical traffic data. Model-based methods predict the behavior for one or numerous vehicles and are suitable for short-term travel time predictions. In Table 3.1, an overview of model-based methods that are found in literature is shown. It can be concluded that all prediction methods use site-based data as input and the travel time as output. The accuracy differs per model and is hard to compare due to the different error functions used. The scope for each prediction method is limited to a single road and all methods have a relatively low prediction horizon.

- **Queuing Theory:** The queuing theory calculates the waiting time, length of the queue and number of vehicles in the queue. Subsequently, these outcomes can be used to calculate the expected travel time. Historical data is used to tune the model parameters.
- **Cell Transmission Model:** The CTM is another model that can be used to simulate the traffic behavior.

Roads are divided into homogeneous sections, called cells, where vehicles travel from one cell to the other. The flow and density of each cell are calculated to evaluate the system and are based on the number of vehicles in each cell at time t.

Table 3.1: Overview model-based methods from literature.

Source	Input	Prediction Method	Output	Accuracy	Scope	Prediction Horizon
[6]	Site-Based Data	Queuing theory	Travel Time	RMSE = 6-18%	Single Road	5 min
[7]	Site-Based Data	Queuing theory	Travel Time	-	Single Road	-
[8]	Site-Based Data	Queuing theory	Travel Time	ER <5%	Single Road	7 min
[9]	Site-Based Data	CTM	Travel Time	RMSE <15%	Single Road	5 min
[10]	Site-Based Data	CTM	Travel Time	-	Single Road	5 min
[11]	Site-Based Data	CTM	Travel Time	MAPE = 18 - 19%	Single Road	-

3.1.2. DATA-DRIVEN METHODS: PARAMETRIC

Where model-based methods can be used for short-term prediction, data-driven methods can be used for short-term as well as long-term predictions. This makes these methods widely applicable. Data-driven methods require historical data to find patterns from the past to predict the future. These methods can be divided into non-parametric and parametric methods. The main difference between these two methods is that parametric methods assume that the data follows a certain distribution, while non-parametric methods do not.

In literature, three non-parametric methods were found that predict travel times. These methods are: linear regression (LR), autoregressive integrated moving average (ARIMA) and kalman filtering (KF). In Table 3.2, an overview of these parametric methods found in literature is shown. It can be concluded that one prediction method uses GPS data as input. All other prediction methods use site-based data as input. All prediction methods have as output the travel time. The accuracy differs per model and is hard to compare due to the different error functions used. The scope for each prediction method is limited to a single road. All methods have a prediction horizon and ranges between 5 and 90 minutes.

- **Linear Regression:** A linear regression method can be used when the relationships between independent (traffic conditions) and dependent (travel time) variables are expected to be linear. An advantage of linear regression is that the independent variables that mostly influence the dependent variable can be identified.
- **ARIMA:** ARIMA is a statistical analysis method that uses time series data to predict the future. Before prediction, the time series are made stationary called differencing, which means that the mean and variance are constant and do not change over time. An extension of ARIMA is a seasonal autoregressive integrated moving average (SARIMA) model. This model supports seasonal trends where ARIMA does not. However, the ARIMA model is sensitive to outliers and should not be used for data with a large variance.
- **Kalman filtering:** Kalman filtering estimates an output based of observations that might be uncertain and inaccurate and is an appropriate method for dynamic information forecasts[12]. Kalman filtering is a method that uses a prediction and correction mechanism. The method adds a correction term, which is proportional to the predicted error, to the previous estimation. By doing this, the state of the system is optimized and the error is minimized.

Table 3.2: Overview parametric methods from literature.

Source	Input	Prediction Method	Output	Accuracy	Scope	Prediction Horizon
[13]	Site-Based Data	LR	Travel Time	MAPE = 7.7 - 23 %	Single Road	60 min
[14]	Site-Based Data	LR	Travel Time	MAPE = 7 - 14%	Single Road	90 min
[15]	Site-Based Data	LR	Travel Time	MAPE = 8.5 - 11.4%	Single Road	25 min
[16]	Site-Based Data	ARIMA	Travel Time	ME <13.9%	Single Road	5 min
[17]	Site-Based Data	ARIMA	Travel Time	MAPE = 5.3%	Single Road	5 min
[18]	Site-Based Data	KF	Travel Time	MARE <2.1%	Single Road	5 min
[12]	Vehicle-Based Data (GPS Data)	KF	Travel Time	MARE = 2.1	Single Road	-
[19]	Site-Based Data	KF	Travel Time	ER <9%	Single Road	45 min
[20]	Site-Based Data	KF	Travel Time	MARE = 1.3 - 4.1%	Single Road	5 min

3.1.3. DATA-DRIVEN METHODS: NON-PARAMETRIC

In general, non-parametric data-driven methods are more frequently used than parametric methods. Non-parametric methods do not make assumptions about the data distribution like the parametric methods. The non-parametric methods that can be found in literature are: support vector regression (SVR), nearest neighbor, ensemble learning and neural network (NN) methods. In Table 3.3, an overview of these methods is shown. It can be concluded a few methods used GPS data as input. All all prediction methods use site-based data as input. The output of all prediction methods is the travel time. The accuracy differs per model and is hard to compare due to the different error functions used. The scope for most prediction methods is limited to a single road. One method predicts for multiple roads and two methods for a single route. All methods have a prediction horizon and ranges between 1 minute and 6 hours.

- **Neural Network:** Neural networks are based on the human brain to recognize patterns. A neural network consist of different layers of nodes where each node combines inputs and weights to calculate the output. Neural networks can be applied to any nonlinear problem and is therefore a powerful model. In literature, four different types of neural networks were found:
  - **Feed-Forward Neural Network:** The feed-forward neural network (FFNN) is the simplest of the four different types. For this type of neural network, the connections between the neurons are only 'fed forward' and are not for example cycles, like for recurrent neural networks. In Figure 3.1, the term *back-propagation neural network* is used, where actually the term *feed-forward neural network* should have been used. This is because back propagation is a training algorithm that is used to optimize the neural network. Back propagation can also be used for the other three types of neural networks and is therefore incorrect to use to indicate feed-forward neural networks.
  - **Recurrent Neural Network:** A recurrent neural network (RNN) allows a bi-directional flow of data between the nodes, which results in a more complex architecture. The RNN has loops in the architecture to keep the information in the neural network and to use it for the following predictions. RNN is able to model dynamic temporal behavior and uses its internal memory to optimize the model. The RNN is widely used for data with a sequential structure.
  - **Long Short-Term Memory Network:** The long short-term memory (LSTM) network is family of the RNN and is also used for sequential data. Instead of a single neural network layer, four layers are used and interact in a unique way.
  - **State-Space Neural Network:** The state-space neural network (SSNN) is also family of the RNN and uses a state-space model. This model can be used for multiple inputs - multiple outputs systems.
- **Support Vector Regression:** Support vector regression (SVR) is a regression method and works with continuous values instead of classification like the support vector machine (SVM) method. The SVR method minimizes the error by individualizing the hyperplane with a maximum margin and a toleration up to a certain error.
- **Nearest Neighbor:** Another term for the nearest neighbors algorithm is k-nearest neighbors (k-NN).

The k-NN put the samples in a feature space and divides these in classes, where the historical data, 'neighbors', is used to predict the travel time.

- **Ensemble Learning:**An ensemble method combines multiple 'weaker' models to achieve one strong model. Examples of ensemble learning methods are random forest (RF) and gradient boosting methods (GBM).

Table 3.3: Overview non-parametric methods from literature.

Source	Input	Prediction Method	Output	Accuracy	Scope	Prediction Horizon
[21]	Site-Based Data	FFNN	Travel Time	MAPE = 7.4 - 17.9%	Single Road	5-25 min
[22]	Vehicle-Based Data (GPS Data)	FFNN	Travel Time	MSE <3%	Single Road	3 h
[23]	Site-Based Data	SSNN	Travel Time	MRE = 1.6%	Single Road	1 min
[24]	Site-Based Data	SSNN	Travel Time	MAPE = 7.3%	Single Road	2 min
[25]	Site-Based Data	RNN/FFNN	Travel Time	MAPE = 4.0 - 17.3%	Single Road	-
[26]	Site-Based Data	RNN	Travel Time	MPE <15%	Single Road	15 min
[27]	Vehicle-Based Data (GPS Data)	LSTMN	Travel Time	MRE = 7 - 11.3%	Single Road	15-60 min
[28]	Site-Based Data	LSTMN	Travel Time	MAPE = 1 - 7.3%	Single Road	5-60 min
[29]	Site-Based Data	SVR	Travel Time	RME = 1 - 4%	Single Road	3 min
[30]	Site-Based Data	SVR	Travel Time	MAPE = 5.9 - 13.1%	Single Road	5 min
[31]	Site-Based Data	SVR	Travel Time	MRE = 9.7%	Single Road	5 min
[32]	Site-Based Data	k-NN	Travel Time	MAPE = 4.3 - 14.8%	Single Road	5-30 min
[33]	Vehicle-Based Data (GPS Data)	1-NN	Travel Time	MAPE = 1.5 - 8.6%	Single Road	5 min
[34]	Site-Based Data	Mk-NN	Travel Time	MAPE = 5.9%	Single Road	6 h
[35]	Vehicle-Based Data (GPS Data)	RF	Travel Time	RMSE <7.5%	Multiple Roads	6-30 min
[36]	Site-Based Data	GB	Travel Time	MAPE = 2.3 - 18.4%	Single Road	5-30 min
[37]	Vehicle-Based Data (GPS Data)	RF/k-NN	Travel Time	MAPE = 6.9 - 14.3%	Single Route	1 h
[38]	Vehicle-Based Data (GPS Data)	RF/GB	Travel Time	MRE = 17 - 29%	Single Route	-

Conclusion

In the literature review of Bai et al. [5], many different travel time prediction methods were reviewed. A couple of these methods use GPS data as input, which will also be used for this research. However, based on the criteria, none of these methods is suitable for this research problem. This is because of the following reasons:

- The scope of the methods are limited to single/multiple roads or a single route. This means that the travel times are not improved for an entire road network, which is required for this research.
- The output of all prediction methods is the travel time. Despite the goal of this research is to improve the travel time predictions, the speed is required as output. This is because first the speed of each road in the map needs to be improved, to improve the travel time prediction accuracy.
- All prediction methods have a prediction horizon. This means that the method predicts travel times in the future, based on historical data. For this research, no prediction horizon is required, since the routing software, developed by ORTEC, is used offline by the customers. This means that the speeds have to be predicted once, in such a way, that the travel time predictions are predicted well and can be used throughout the year.

Because there are many travel time prediction methods available in literature, some methods were not included in the literature study from Bai et al. [5]. Therefore, an additional literature review was done, to find travel time prediction methods that might fit to this research problem. It is required that the travel time prediction method 1) focuses on a road network, 2) has as output speed to improve the travel time predictions

and 3) predict average speeds once that can be used throughout the year. Additional travel time prediction methods that were found and seems to comply with these criteria, based on the research title, are:

- A research from Jensen and Larsen [4] was found with the title: *Travel-Time Estimation in Road Networks Using GPS Data*. This seems to be a perfect reference for this research, since GPS data is used for travel time estimations in a road network. However, this method only focuses on improving travel time estimations of two roads, Vesterbro and Sohngardsholmsvej, in a road network in Aalborg. Also, the GPS data that was only collected from these two roads instead throughout a road network. Therefore, this travel time prediction method is not applicable to this research.
- Another research, from Anderson [39], was found with the title: *Travel Time Prediction in Urban Road Networks*. This research focuses on improving travel time predictions on nine links (roads) in a road network. Also, a site-based data source was used, called license plate matching. Therefore, also this travel time prediction method is not applicable to this research.
- Asif et al. [40] researched the speed prediction of roads in a large road network with the title: *Unsupervised Learning Based Performance Analysis of ν-Support Vector Regression for Speed Prediction of A Large Road Network*. In this research, a speed prediction model is developed to predict speeds in a road network through clustering of the roads. This comes close to the research in this report. However, the speed predictions are based on current and past speed trends, where speeds are predicted with a prediction horizon between 5 and 60 minutes. Also, the roads are only divided into four clusters, which is a rough road classification. Therefore, this developed speed prediction model is also not applicable.

As a result, there is no complete travel time prediction method in literature, except the one developed by den Heijer, suitable for this research problem. The travel time prediction method of den Heijer has been explained in section 2.2. Collected GPS points are matched to a road in the road network from which the road properties are looked up. After this, relationships between the road properties and the driven speed are found using a prediction method such as linear regression and random forest. Subsequently, the speed of each road in the road network is predicted based on its properties and the relationships found by the prediction method. Lastly, the travel times can be predicted using a shortest path algorithm.

Despite other travel time prediction methods in literature are not applicable to this research, some prediction methods in Figure 3.1 can be adapted and implemented into the methodology from den Heijer. The prediction methods that can be used are linear regression, random forest, support vector regression, gradient boosting and neural networks. These prediction methods are suitable, since they can predict a continuous output (speed) and are able to generalize speed predictions to an entire road network. All five prediction methods are machine learning (ML) methods, which is a subset of data-driven methods. In the following section, a method trade-off of these prediction methods will be performed.

3.2. METHOD TRADE-OFF

In section 3.1, it was concluded that there are five prediction methods suitable to predict the speeds in this research problem. These methods are: linear regression, random forest, support vector regression, gradient boosting and neural networks (Appendix C). Den Heijer already used, as discussed in section 2.2, linear regression and random forest. In this section a trade-off between all five prediction methods will be made to find the most suitable prediction method for this research. Linear regression and random forest are also included in the method trade-off to compare the other models with.

3.2.1. TRADE-OFF CRITERIA

To select the right prediction method, a multiple-criteria analysis will be used. This is a common way to make an appropriate selection. Surprisingly, there are no machine learning method trade-offs available in literature that can be used as reference for this research. Therefore, the criteria will be defined such that it helps to select the best prediction method for this research. The criteria that are used to find the best prediction method for this research are the following:

- **Prediction accuracy & interpretability:** To distinguish machine learning methods, often the properties prediction accuracy and interpretability are used. Different references, such as [41] and [42], explain that a higher prediction accuracy comes at the expense of a lower interpretability of the model. The interpretability of the model indicates how well it is understood why something is predicted. Or in

other words, how well can the relationships between the independent and dependent variables be understood. Both are important criteria to select a prediction method and will be both included in the method trade-off.

- **Hyperparameter tuning:** Hyperparameters are parameters that need to be tuned by the user to get the best performance of the model. More hyperparameters make the model more complex and increase the computational time for hyperparameter tuning. Also, a higher training time of the model increases the computational time needed for hyperparameter tuning. Both the number of hyperparameters and training time of the model are important indicators for this criterion. Because the hyperparameter tuning process differs between the prediction methods, hyperparameter tuning will also be used as criterion.
- **Large dataset:** The dataset that is used for this research is 'large' (~1,000,000 data points). It is important that the prediction method is able to handle this large amount of data to find the underlying patterns and to obtain an increased speed prediction accuracy. Therefore, 'large dataset' will also be used as criterion to select the right prediction method.

Criteria Importance

In the method trade-off, the importance of the criteria will be distinguished by assigning a weight to each criterion between 1 and 5. A higher weight means a higher importance of the criterion. From the criteria above, the *prediction accuracy* and *large dataset* are the most important criteria and get a weight equal to 5. The criterion *prediction accuracy* is important, since this leads to a model with accurate speed predictions and possibly to an improved travel time prediction accuracy. The criterion *large dataset* is also and important criterion, since a large data set allows the model to find more underlying relationships between the independent and dependent variables. This may also increase the speed prediction accuracy. *Hyperparameter tuning* and *interpretability* are the least important criteria with a weight of 2 and 1 respectively, since they have the lowest impact on the main research objective. The criterion *interpretability* has the lowest weight, since the independent variables are carefully chosen through a literature study and reasoning. Also, the goal of the research is not to understand why something is predicted.

3.2.2. TRADE-OFF PREDICTION METHODS

In Table 3.4, a trade-off between five different ML methods is shown. In the first column, the criteria used to find the most suitable method is shown. In the second column, the importance of each criterion is indicated by a weight that ranges between 1 and 5, where 1 means not important and 5 very important. In the last five columns, a score can be found for each prediction method with respect to each criterion. This score ranges between 1 and 5, where 1 means bad performance and 5 great performance for that specific criterion. The total score, which is the sum of the multiplication between the weight and scores, can be found in the last row. From Table 3.4, it can be concluded that the neural network has the best total score, while the SVR has the lowest total score.

Table 3.4: Trade-off between multiple machine learning methods based on different criteria for point-based data.

Criteria	**Weight**	**LR**	**RF**	**SVR**	**GB**	**NN**
Prediction Accuracy	5	1	3	4	4	5
Hyperparameter Tuning	2	5	4	2	2	1
Interpretability	1	5	3	2	3	1
Large Dataset	5	4	4	1	4	5
Total		40	46	31	47	**53**

Prediction Accuracy

The prediction accuracy for each prediction method depends on the problem. From the research of den Heijer, it was concluded that the random forest performed better than linear regression. The prediction accuracy for SVR, GB and NN for this research problem is not known. Therefore, the score that are assigned to these prediction methods are based on how these models perform in general based on [41], [42] and [43]. Linear regression has the lowest prediction accuracy, since it can only model linear relationships. Therefore, linear regression has a score of 1. The Neural Network has the highest score of 5, since it can model very complex relationships which increases the prediction accuracy. In general, Gradient Boosting is more accurate than

random forest and therefore get a score of 4 and 3 respectively. This is because Gradient Boosting build the decision trees sequentially instead of parallel. This means that the next decision tree is based on the mistake made by the previous trees instead of building the decision tree randomly as done by random forest. Support vector regression also has a relative high prediction accuracy and gets a score of 4, however this is limited to smaller data sets.

Hyperparameter Tuning
The scores for linear regression and random forest for the criterion hyperparameter tuning can be derived from the research of den Heijer. It was found that the linear regression model has no hyperparameters and that the training time is very quick with less than 3 seconds and therefore has the highest score of 5. The random forest model also had a fast training time around 10 seconds, but the hyperparameter *maximum depth of decision trees* had to be trained. Therefore, a score of 4 was given to the random forest prediction method. Neural network has the lowest score of 1, this is due to many hyperparameters (~10) and a relatively long training time. The long training time is due to the computational expensive training process where many weights and biases are optimized through many iterations. The SVR has mainly three hyperparameters: epsilon (thickness of a tube), C (penalty factor) and sigma (kernel function parameter[44]. However, the training time of the SVR increases quadratically with respect to a larger data set. For a large dataset, which is used in this research, the computational time for hyperparameter tuning may increase significantly. Due to the long training time, but little hyperparameters, the SVR gets a score of 2. Gradient Boosting has a relatively low training time compared to NN and SVR, but has many hyperparameters (~10)[45]. The number of hyperparameters make the hyperparameter tuning process of Gradient Boosting less attractive, and therefore gets a score of 2.

Interpretability
The scores that are assigned to the prediction methods for the criterion interpretability are based on [41], [42] and [43] just like the prediction accuracy. Linear regression is the method that is easiest to interpret and gets the highest score of 5. This is because the importance of each independent variable is expressed by the value of the coefficients. Neural networks are least interpretable and are also called 'black-boxes', where the decision making of the model cannot be derived and gets a score of 1. The random forest and Gradient Boosting methods can be interpreted to a certain extent and have a score of 3. Each individual decision tree can be interpreted well, but due to the many decision trees used for both methods, the interpretation is more challenging.

Large Dataset
In general, all machine learning methods in the method trade-off, except SVR, are able to handle large data sets. SVR are less appropriate for large data sets due to the quadratically increase of the training time with the increase of training points and therefore gets a score of 1. The neural network has the highest score of 5, because it handles large data sets very well. In general, it is able to improve its prediction accuracy with an increased data size, while the other prediction methods have reached that saturation point[46]. Therefore, LR, RF and GB have a lower score than the neural network which is 4.

PREDICTION METHOD TRIP-BASED DATA
The method trade-off that is performed in Table 3.4 is applicable to point-based data. However, learning and predicting from trip-based data requires an additional capability of the prediction method. The prediction method needs to be able to learn from combinations of multiple roads, between two GPS data points, while the model is finally used to predict the speeds on single roads. This can be seen as a sort of extrapolation. In subsection 2.3.1, it was explained that trip-based data cannot be used for random forest models. The random forest uses a bunch of decision trees which are bad at predicting different combinations of the independent variables than trained on (multiple road vs. single roads). Because Gradient Boosting also consists of decision trees, trip-based data is not suitable for this prediction method as well. As a result, linear regression, support vector regression and neural networks are left as suitable prediction methods for trip-based data. Because the neural network has the best total score in Table 3.4, the neural networks is also the best prediction method for trip-based data.

3.2.3. SENSITIVITY ANALYSIS METHOD TRADE-OFF
Additionally, a sensitivity analysis was done for the method trade-off shown in Table 3.4 to make sure that the outcome is robust. The sensitivity analysis will be done by changing the weights and scores in such a way that it disfavors the total score of the neural network compared to the other models. In total, four cases will be

analyzed that have the largest impact on the relative score between the neural network and the other models and are:

- **Case 1:** Decreasing the *weight* of the criterion 'prediction accuracy' by 1, since the neural network scores here the best.
- **Case 2:** Increasing the *weight* of the criterion 'hyperparameter tuning' by 1, since the neural network score here the worst along with Gradient Boosting.
- **Case 3:** Increasing the *weight* of the criterion 'interpretability' by 1, since the neural network score here the worst.
- **Case 4:** Decreasing the *score* of the neural network for the criterion 'prediction accuracy' by 1, since the neural network scores here the best.

The outcomes of the four cases can be found in Appendix D. The outcomes show that for each case the neural network has still the highest total score and is therefore a good choice as prediction method for this research.

3.3. NEURAL NETWORK

From the method trade-off in section 3.2, it was concluded that the neural network is the best prediction method for point- and trip-based data. Neural networks or also called artificial neural networks (ANN) are inspired by the human brain. The neural network consists of neurons and synapses. The synapses connect the neurons and conveys the input and output from one neuron to the other. Besides the hyperparameters that can be altered to tune the model, no other control can be exerted on the model. Basically, the network of neurons between the inputs and outputs is a black box. It is hard and usually impossible to trace back which input variable influences which output variable and to which extent. In this section the principles of the neural network are discussed as well as the neural network type that will be used for this research.

3.3.1. SINGLE PERCEPTRON

The most simple type of NN, with one input layer and one node in the output layer, is called a perceptron and is shown in Figure 3.2. The perceptron consists of n inputs x_1, x_2...x_n which are multiplied by weights w_1, w_2...w_n before fed into the node through the links, also called synapses. The node sums these inputs by Equation 3.3.1, after which a bias b is added. The bias can be compared to the constant in a linear function, $y = ax + b$, where b would represent the bias. The bias is unknown and should be learned by the model just like the weights. Inside the nodes, an activation function f can be found. An activation function is added to add non-linearity and to be able to model complex relationships. After the activation function has been applied, an output y is produced as shown in Equation 3.3.2.

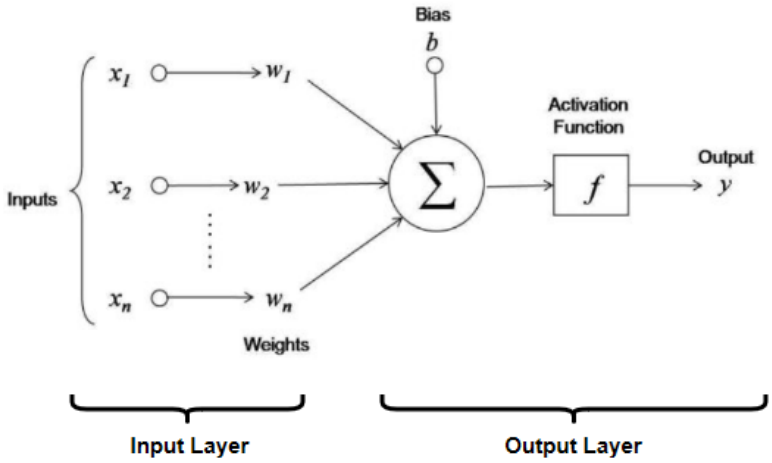

Figure 3.2: The most simple type of NN, perceptron, which consists of an input layer and an output layer with one node[47].

$$h = \sum_{i=1}^{N} w_i x_i = w_1 x_1 + w_2 x_2 + ... + w_n x_n \tag{3.3.1}$$

$$y = f\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{3.3.2}$$

3.3.2. FEED-FORWARD NEURAL NETWORK

The perceptron, as discussed above, is the simplest NN. However, it is mostly not an accurate predictor, since it does not support high complexity problems. By adding hidden layers and neurons between the input and output layer, a model can be developed with increased flexibility and complexity. In this way, the neural network can be applied to any non-linear problem and is therefore a powerful model. In subsection 3.1.3, four different types of neural networks were distinguished and explained, and are the following:

1. Feed-Forward Neural Network
2. Recurrent Neural Network
3. Long Short-Term Memory Network
4. State-Space Neural Network

The input data for this research is based on GPS data and is not sequential. This is because the order of the data, to train the model, is not important for the speed predictions. Therefore, the feed-forward neural network will be used for this research. The other three neural network types are only suitable for data with a sequential structure, where the next prediction depends on the previous ones. Below, a description of the feed-forward neural network is provided.

The information in a feed-forward neural network flows into one direction, from the input layer to the hidden layer(s) and finally to the output layer. The architecture of a feed-forward NN with an input layer, two hidden layers and output layer is shown in Figure 3.3. The two hidden layer can be extended to more hidden layers if a more complex model is desired. The number of neurons in the input layer is equal to the number of independent variables. The number of neurons in the output layer is equal to number of dependent variables (outputs) of the model. The number of neurons in the hidden layers differs per neural network and have to be determined by the user. The neurons in the neural network are connected to the neurons in the previous and next layer through links, where the output of a neuron is multiplied by the weight of the link.

The output of each neuron in the input layer is shown in Equation 3.3.3. This is equal to the (normalized) value of the independent variable x_j (I_j). Where j is the j_{th} neuron in layer i and i the layer number which is equal to 1 for the input layer.

$$a_j^i = x_j \tag{3.3.3}$$

The output of the neurons in the hidden layer and output layer can be calculated by Equation 3.3.4. Where σ^i is the activation function of each neuron in layer i, w_{jk}^i the weight from neuron k in layer *(i-1)* to neuron j in layer i, a_k^{i-1} the output of neuron k in layer *(i-1)* and b_j^i the bias of neuron j in layer i.

$$a_j^i = \sigma^i\left(\sum_k \left(w_{jk}^i \cdot a_k^{i-1}\right) + b_j^i\right) \tag{3.3.4}$$

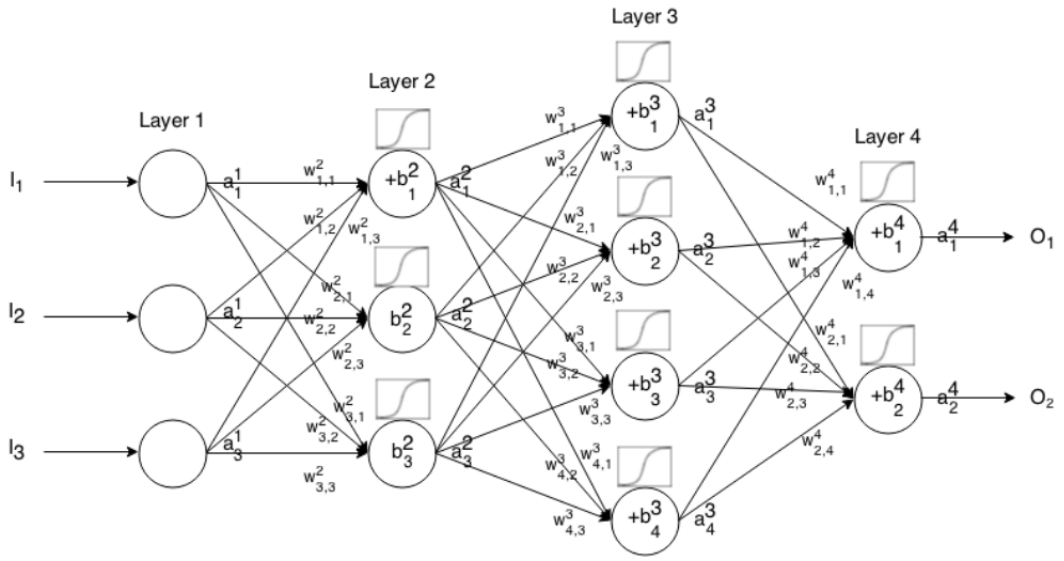

Figure 3.3: A feed-forward network with an input layer (Layer 1), hidden layers (Layer 2 & 3) and output layer (Layer 4)[48]

3.4. RESEARCH METHODOLOGY

In this section the methodology for this research is discussed. A visualization of the methodology is shown in Figure 3.4 through a flowchart. This methodology is based on den Heijer's approach and the neural network that will be used as prediction method. The methodology can be divided into four parts and will be discussed in the next three chapters. Below a short discussion of each part of the process is provided:

Preprocessing
In chapter 4, the collection and preprocessing of the data will be discussed. There are two GPS data sets that will be used from different customers of ORTEC. The first data set is from den Heijer which has already been collected and preprocessed. The second data set is collected and preprocessed during this research with a higher frequency of 2 minutes. Before using the raw collected GPS data, first some preprocessing steps have to be done. In this way, the data can be properly used to train the model. First, the data will be cleaned where all irrelevant GPS data points are removed. Then, the cleaned data points are matched to a road in the digital map using a map matching algorithm. After each GPS data point is matched to a road in the map, the features of the roads are extracted for each GPS point. This is done by looking up the corresponding road properties in the map data. The trip-based data needs an additional preprocessing step. This includes the calculation of the average road properties of all roads between two GPS points (trip). Lastly, the point- and trip-based data set will be split into a training (75%) and a test (25%) set. The training set will be used to the train the model and to find relationships between the independent (input) and dependent (output) variables. The test set is used to obtain the training and travel time prediction accuracy.

Model Design
In chapter 5, the model design of the neural network will be discussed. First the independent and dependent variables of the model will be defined. Subsequently, the neural network is implemented in Python, after which the hyperparameters can be chosen and tuned. The hyperparameters that will be tuned include the learning rate, number of neurons, number of hidden layers and mini-batch size of the neural network.

Training & Evaluation (Points and Trips)
In chapter 6, the model will be trained on either point- or trip-based data. After the model is trained, the speeds in the point-based data test set are predicted using the input (independent) variables of the speeds in the test set. By comparing these speeds with the actual speeds, the speed prediction accuracy can be calculated.

Prediction & Evaluation (Routes)
After the evaluation of the trained model, the model is used to predict the speed of each road in the map. After this, the map is updated. This is the process of replacing the speeds of the roads in the map by the new predicted speeds. Subsequently, the travel times of the travels in the test set can be predicted. This is done

by the routing software of ORTEC. It calculates the shortest path from the first to the last GPS point of the travel, after which the expected travel time is calculated. Lastly, the predicted travel times can be compared to the travel times of the travels in the test set. This results in a travel time prediction accuracy of the speed prediction model.

3.5. CONCLUSION

In this chapter, the following conclusions could be made:

- The travel time prediction methods that can be found in literature can be divided into model-based and data-driven methods. Model-based methods are based on a physical mechanism, while data-driven methods learn from data. It was found that no complete travel time prediction method in literature, except the one from den Heijer[1], can be used for this research problem. This is because these methods only focus on one road or route instead of a road network, for which the travel time is directly calculated. Also, the prediction methods have a prediction horizon, while a method without prediction horizon is needed for this research. This is because the speeds have to be predicted once, to be used throughout the year. Therefore, the methodology of den Heijer, explained in section 2.2, is the only method that can and will be used for this research.
- Despite the full travel time prediction methods in literature cannot be used, the used prediction methods can be adapted and used to improve the travel time prediction method of den Heijer. The prediction methods that can be used for this research problem are linear regression, random forest, support vector regression, gradient boosting and neural networks. These prediction methods are suitable, since they can predict a continuous output (speed) and are able to generalize speed predictions to an entire road network. After a method trade-off, it was found that the neural network is the best prediction method. The *feed-forward* neural network will be used for training and prediction, based on point- and trip-based data. Other types of neural networks are not suitable, since they can only be used for sequential input data, which will not be used in this research.
- In Figure 3.4, the methodology of this research is shown through a flowchart. The methodology is based on den Heijer's approach combined with the neural network prediction method. It can be divided into four parts: preprocessing, model design, training & evaluation and prediction & evaluation. These parts will be discussed in this order in chapter 4, chapter 5 and chapter 6.

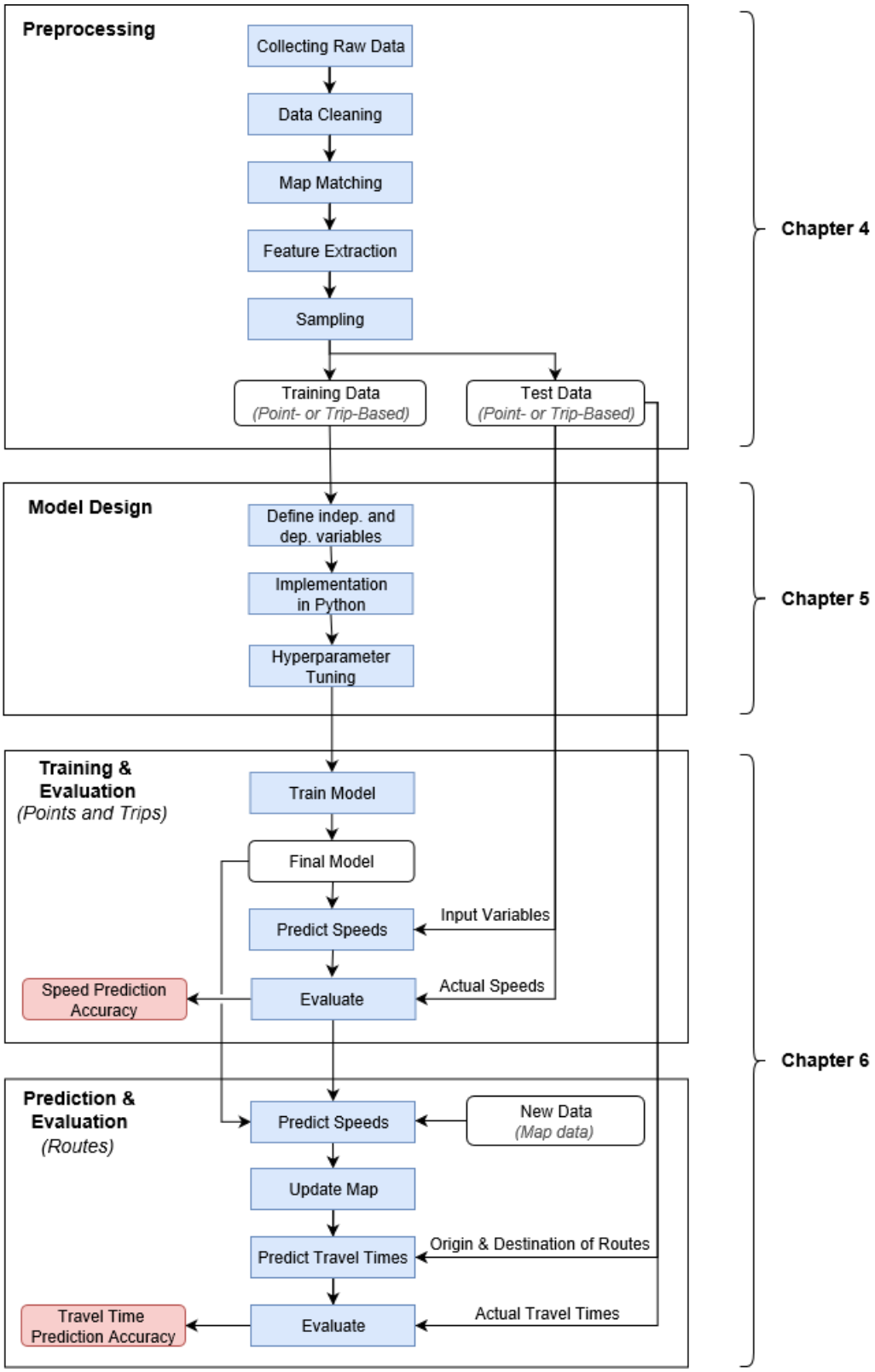

Figure 3.4: Flowchart of the methodology for this research, from preprocessing to evaluation of the travel times.

4

DATA

In this chapter, the collection and preprocessing of the data is discussed. The data is the input of the model and comes from two different sources:

1. **Map data:** This is the data that ORTEC has purchased from *HERE Technologies*. The map data contains information about the road network and many properties of each road, discussed in subsection 4.4.2. The map data is clean and well structured. This makes it easy to use.
2. **Customer GPS-data:** The GPS-data is collected from customers that operate with fleets of trucks. This data is raw and of relatively low quality. Much effort is required for cleaning and preprocessing to use this data.

In this chapter first a description of the GPS data is provided. Secondly, the cleaning steps and the map matching process of the GPS data is discussed. Where the map matching is needed to assign each GPS point to a road in the map. After this, the factors that influence the travel time are researched. In this way, only the road properties from the map data, which are important, are included in the model input. This reduces the computational time of the neural network as well as the chance of overfitting. At the end of the chapter the sampling of the data into a training and test data set is described.

4.1. DESCRIPTION OF USED GPS DATA

In this research, both the GPS data from den Heijer (5 min. frequency) and new collected GPS data (2 min. frequency) during this research will be used. For convenience, the preprocessed GPS data from den Heijer will be called 'old data' and the preprocessed GPS data collected during this research 'new data'. The new and old data are both used for this research because of the following reasons:

- **New data:** During this research, another customer of ORTEC was willing to share its GPS data. This GPS data has a frequency of 2 minutes which is higher than a frequency of 5 minutes from the old data. It is expected that the quality of the trip-based data will be improved with a higher GPS data frequency. As a consequence, the speed predictions of the neural network will improve. This may result in better travel time predictions such that the benchmarks are outperformed.
- **Old data:** den Heijer already collected and preprocessed GPS data with a frequency of 5 minutes which is ready to use. The data is obtained from a different customer than the new data. By using the old data as well, more insights can be obtained about the developed speed prediction models. Is the best speed prediction model for the new data the same as for the old data? Can a neural network, trained on trip-based data, already outperform the benchmarks with a GPS data frequency of 5 minutes?

A more detailed description of the GPS data that is used for the new and old data is provided below.

NEW GPS DATA

The new GPS data that is collected during this research is from a customer of ORTEC that is specialized in food services. The food services are distributed by the company itself with 165 trucks throughout the Netherlands. In total, 2,513,856 GPS data points were collected and are shown in Figure 4.1. The GPS data is mainly

recorded in the provinces: Groningen, Limburg, North Holland, South Holland, Utrecht and Zeeland. The majority of travels are driven between 4:00 and 17:00 and can be explained by the fact that the food has to be delivered before the time of either breakfast, lunch or dinner. The GPS data is collected throughout 2017 with an average frequency of 2 minutes. Further detailed information about the exact locations of the travels are kept confidential due to data protection laws. Besides the longitude and latitude, the GPS data also contains the following additional information about each recorded GPS point:

- Vehicle ID
- Driver name
- Trailer ID
- Activity
- Speed
- Heading
- Date and time
- Fuel level and fuel used
- Mileage (total distance of vehicle)
- Country

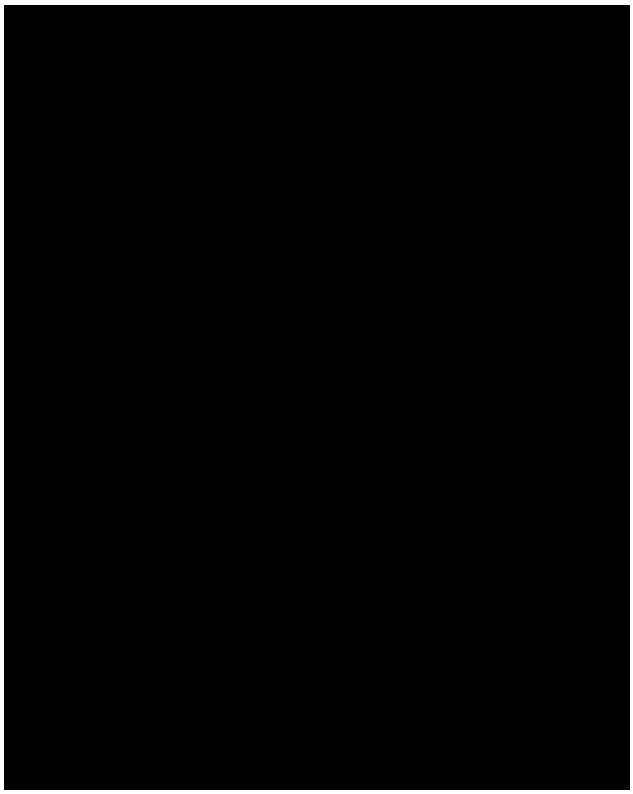

Figure 4.1: All collected GPS data points (2,513,856) from trucks in the Netherlands (new GPS data). This is point-based data.

OLD GPS DATA

The data that was used by den Heijer is GPS data is collected from a customer of ORTEC that is specialized in the storage and distribution of packaged chemicals. The packaged chemicals are delivered by the company throughout the Benelux with circa 60 trucks. The majority of the travels are driven between 3:00 and 17:00 and are tens of thousands of kilometers per day. The GPS data that is obtained is recorded between February and July 2018 with an average frequency of 5 minutes between two recorded GPS points. In total, 2,803,546 GPS data points were collected and are shown in Figure 4.2. It can be seen that the GPS points are more spread compared to the new GPS data in Figure 4.1. Further detailed information about the exact locations of the travels are kept confidential due to data protection laws. Besides the longitude and latitude, the GPS data also contains the following additional information about each recorded GPS point:

- Vehicle ID
- Speed
- Heading
- Date and time
- Mileage (total distance of vehicle)

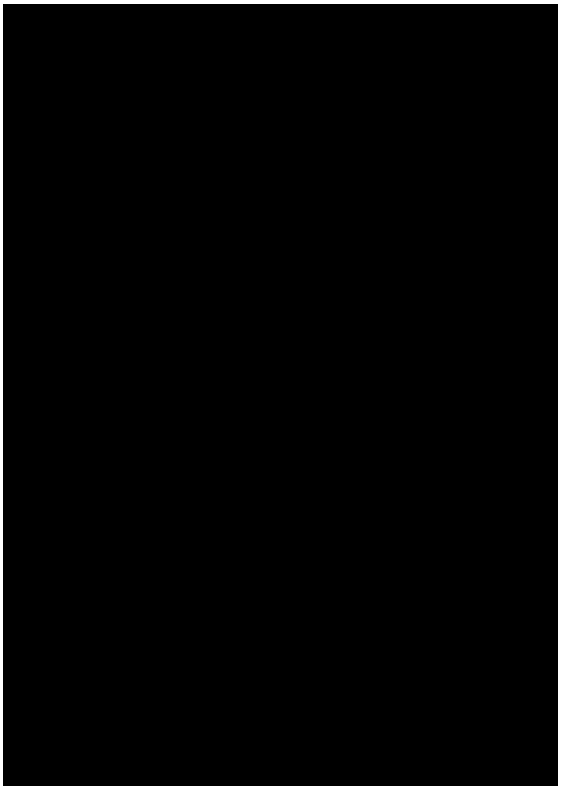

Figure 4.2: All collected GPS data points (2,803,546) from trucks in the Benelux (old GPS data). This is point-based data.

4.2. DATA CLEANING

After all raw GPS data is collected from the customer, the data needs to be cleaned so the model only learns from representative data. The data cleaning steps for the new and old GPS data sets are slightly different. This is due to different information that was included in the GPS data sets. However, the cleaning data steps of the new and old GPS data can be considered to be more or less similar. This should not have an impact on the final results. The data cleaning steps for the old GPS data were already done by den Heijer. The data cleaning steps for the new GPS data were done during this research.

NEW GPS DATA

In total, 2,513,856 GPS points were collected for the new data set. After data cleaning, the new data set was reduced to 1,436,985 points and 1,049,195 trips. The total number of remaining travels are 183,408, where 45,852 travels are added to the test set. The travels in the test set are used to evaluate the travel time prediction accuracy. The data cleaning steps of the new GPS data are described below:

1. Each collected GPS point contains information about the activity of the vehicle. These activities are: engine off, engine on, driving, standing still, refueling, contact key off, prepare for departure, contact key on, loading and unloading. Because each GPS point got assigned an activity, relevant GPS points for the learning model can be easily taken. The GPS points with an activity other than 'driving' are discarded from the data set.
2. All first and last GPS points from a travel that contain a speed equal to or below 5 km/h are filtered. It was found that the majority of the first and last GPS points of a travel, with a speed equal to or below 5 km/h, are on a private terrain. These data points are unrepresentative for learning.
3. All travels that include Belgium are filtered. When the travels in Belgium would be included, then also all roads in Belgium need to be predicted. This would increase the computational time significantly. By removing the travels in Belgium, only 0.17% of all travels are removed.
4. GPS points that could not be matched to a road in the map are filtered.

5. All GPS points and travels that contain a ferry 'road' are filtered. These travels are considered to be out of scope for this research.
6. After map matching, all GPS points with a minimum map matching error of 20 meters are filtered. This means that all points that are at least 20 meters from the closest road in the map are filtered. The minimum map matching error above 20 meters is mainly caused by trucks that are on private terrain, which is not included in the map. These GPS points are considered to be unreliable.
7. Travels that only contain 1 data point are filtered. This is because the training and test set of trips need at least two useful data points of each travel, which is also needed to evaluate travel times.
8. For the trip-based data, very short trips that are shorter than 15 seconds and 50 meters are filtered. These conditions are also used by den Heijer to remove inaccurate trips.

OLD GPS DATA

In total, 3,475,479 GPS points were collected by den Heijer. After applying all data cleaning steps, the old data was reduced to 550,268 points and 219,828 trips. The total number of remaining travels were 41,264, from which 10,316 were added to the test set. The travels in the test set are used to evaluate the travel time prediction accuracy. The data cleaning steps applied by den Heijer are as following:

1. Sometimes, measurements were recorded in quick succession. Perhaps, multiple events happen at one instant but cause several messages. To compensate for this, points are merged where the mileage has not changed. This way, stops (which normally consist of multiple points in sequence) are merged into one point.
2. Very long travels with over 10,000 km distance are removed.
3. All travels that are not driven by a truck are removed from the data set.
4. GPS points that could not be matched to a road in the map are filtered.
5. All GPS points and travels that contain a ferry 'road' are filtered. These travels are considered to be out of scope for this research.
6. After map matching, all GPS points with a minimum map matching error of 20 meters are filtered. This means that all points that are at least 20 meters from the closest road in the map are filtered. The minimum map matching error above 20 meters is mainly caused by trucks that are on private terrain, which is not included in the map. These GPS points are considered to be unreliable.
7. All travels that only contain 1 data point are filtered. This is because the training and test set of trips need at least two useful data points of each travel, which is also needed to evaluate travel times.
8. For the trip-based data, very short trips that are shorter than 15 seconds and 50 meters are filtered.

4.3. MAP MATCHING

A challenge that comes with the use of GPS data is map matching. Map matching is the process of matching GPS data, consisting of a latitude and longitude component, to locations in the digital map. In Figure 4.3, an example of this process is shown, where on the left image the GPS points are indicated by blue dots and on the right image the corresponding path is shown which is found by applying a map matching algorithm.

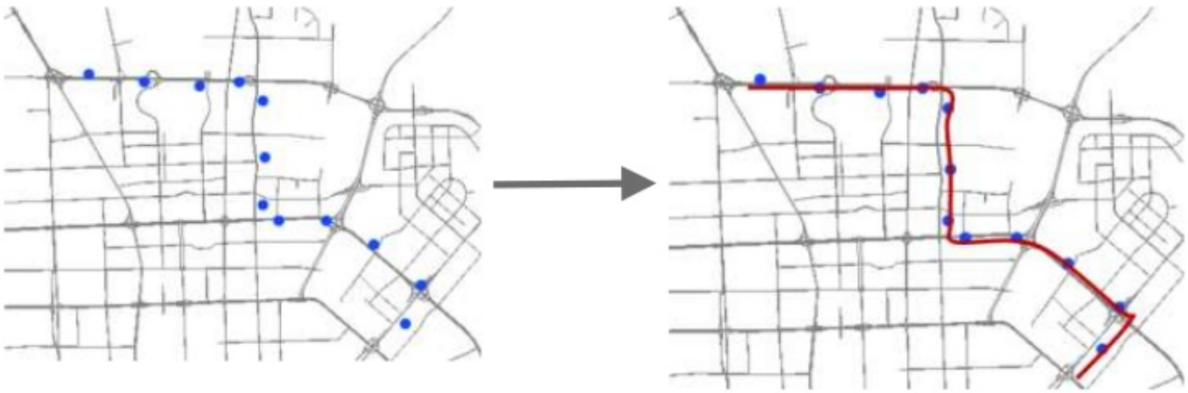

Figure 4.3: Illustration of matching GPS data to roads in a map, this process is called map matching[49].

4.3.1. MAP MATCHING ALGORITHMS IN LITERATURE

The inaccuracy of the GPS data makes the map matching process of GPS data to locations in the digital map challenging. The inaccuracy is caused by different reasons such as the satellite geometry, signal blockage, atmospheric conditions and receiver design features/quality[50]. For ideal circumstances, an accuracy between the 5 and 10 m is claimed[51]. Stanford University[52] did measurements in an open sky and in a city with a fixed GPS location. The mean for the measurements with an open sky was found to be 4.9 m, while the mean shifted to 16.8 m in the city. The reduction of the GPS accuracy in the city could be blamed to the signal blockages by the surrounded buildings.

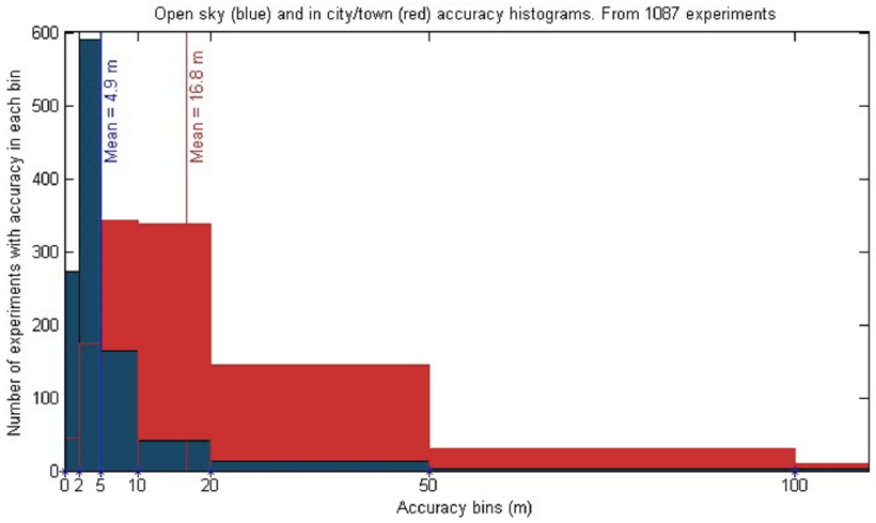

Figure 4.4: Accuracy distribution of fixed GPS measurements (1087 samples) in open sky and in city/town[52][1].

In literature, many algorithms can be found that improve the map matching accuracy. Quddus et al. [53] provides a literature review of available map matching algorithmsThese algorithms range from simple search techniques to more advanced techniques such as an Extended Kalman Filter, fuzzy logic and Belief Theory. The map matching algorithms were categorized into four categories: geometric, topological, probabilistic and advanced techniques. A brief explanation of these algorithms can be found below[53]:

- **Geometric:** Geometric map matching algorithms are simple and only consider the shape of the roads. The information about the road connections is not utilized leading to less accurate matches. The most common geometric map matching algorithm is a simple search algorithm called point-to-point matching. This algorithm is fast and easy to implement[54]. This algorithm matches the GPS location to the closest node or shape point of a road. Other geometric map matching algorithms use point-to-curve matching which performs better than point-to-point algorithms[54][55]. This algorithm calculates the

distances from the GPS location to each road, after which the closest road is matched. The drawbacks of the geometric map-matching approaches are the sensitivity to outliers and the instability for areas with a high road density.

- **Topological:** Topological map matching algorithms use the geometry of the roads as well as the connectivity and contiguity of the roads. These algorithms are relatively quick and easy to implement[56]. Greenfeld [57] introduced a weighted topological algorithm and uses the heading, proximity, link connectivity and turn restriction to assign each GPS point to the right road. Quddus et al. [58] proposed an algorithm that also included the speed.
- **Probabilistic:** Probabilistic map matching algorithms use an elliptical or rectangular confidence region around each GPS location. This region is based on the error variances of the GPS device[59]. Within this confidence region, the GPS location is matched to the most likely road based on the heading, connectivity and closeness criteria. Also other criteria such as speed, and distance to junctions can be used for further improvement of the map matching algorithm.
- **Advanced:** Advanced map matching algorithms are algorithms that use more refined approaches. Examples are: (Extended) Kalman Filter, Dempster-Shafer's mathematical theory of evidence and an interacting multiple model. In general, the performances of these advanced methods are better than the geometric, topological and probabilistic algorithms. However, these algorithms are more difficult to implement and require more computational time.

4.3.2. USED MAP MATCHING ALGORITHM

To save a significant amount of time on the development and implementation of a map matching algorithm, the online map matching service from *HERE* is used. This map matching service is called *Fleet Telematics Route Matching*. A sequence of GPS points of an unique travel are sent to the API and are matched to the most likely driven roads in the map. The most likely driven roads are determined by the longitude, latitude, timestamp, speed, heading and the other GPS points of the same travel. The map matching algorithm has two outputs: the minimum error and link IDs. The minimum error is the minimum distance from the GPS location to the nearest road in the map. The link IDs are the identification numbers of the matched roads in the map. The map matching service from *HERE* is both used for the old and new GPS data set.

It is not clear exactly which map matching algorithm has been used by *HERE*. From the required input and the documentation[60] it is expected that a topological algorithm is used. A topological algorithm uses, as explained in subsection 4.3.1, the geometry, connectivity and contiguity of the roads in the map and the longitude, latitude, speed, heading and time of the GPS points. A topological algorithm is relatively quick and simple to implement, while preserving a high accuracy. *HERE's* map matching algorithm also ignores GPS points that are too far from the path causing a big detour. GPS points that have '0' as longitude and latitude, meaning that the recorded GPS location is unknown, are also ignored.

4.4. FEATURE EXTRACTION

After matching the GPS data to locations in the digital map, features can be extracted. This includes the process of deriving values (features) that are informative to facilitate the learning and predicting process of the model. For this research, it is important to find features (independent variables) that are predictors for the speed and travel time. These features will be used to train the model and to predict the speed of each road in the map. In this section, first a literature review will be conducted to find out which features are relevant to include in the model input. After this, relevant features will be extracted from the map data which are the road engineering factors. Lastly, a description will be provided how the temporal factor will be included in the model to improve the travel time prediction accuracy even further.

4.4.1. LITERATURE REVIEW OF TRAVEL SPEED INDICATORS

In this section, travel speed indicators that are found in literature are discussed. Travel speed indicators are factors that influence the driven speed and as a result the travel time. MacAngus et al. [61] identified six factors that influence the operating speed and are: 1) vehicle classification, 2) temporal factors, 3) weather factors, 4) driver attitude and behaviour, 5) regulatory and enforcement environment and 6) road engineering factors. Sigakova [62] listed 12 important factors that influence the operating speed and are categorized into predefined, predictable and unpredictable classes. To combine the factors from both literature sources, the

factors are categorized into five classes. A description of these five classes is provided below:

1. **Vehicle classification** - truck, car, delivery van, motor, etc.
2. **Temporal factors** - season, day of week, time of day, etc.
3. **Weather factors** - precipitation, snow, wind, temperature, etc.
4. **Road engineering factors** - speed limit, lane width, speed bumps, bridge, tunnel, etc.
5. **Unpredictable factors** - driver attitude, incident, unscheduled road closure, etc.

VEHICLE CLASSIFICATION

The type of vehicle has a large influence on the operating speed. This differs due to the condition of the vehicle, the maneuverability, the shape of the vehicle and the vehicle speed management systems[63][64]. For larger vehicles such as trucks and buses, the maximum speed on highways is reduced due to environmental and safety reasons. In addition, the type of load such as hazardous and fragile materials, which needs special attention, affects the operating speed of the vehicle as well. Weight and height restrictions for large vehicles on specific location such as bridges, tunnels, mountainous passes and certain urban areas, also forces the driver to take a different route. This does not always directly influence the operating speed, but rather increases the travel time due to a diversion.

TEMPORAL FACTORS

Temporal factors such as part of the day, day of the week, seasons and holidays have a large impact on the vehicle operating speed [62]. The main reason for a reduction in operating speed is caused by the vehicle density on the road. The vehicle density is especially high during recurrent congestions that take place during morning and evening rush hours throughout the mid-week. Other reasons that influence the operating speed due to the temporal factors are driver performance, trip purpose and lighting conditions. Brilon & Ponzlet [65] found that darkness reduces the velocity by 5 km/h on German highways without a speed limit. The temporal factors can be used to find recurrent patterns in the driven speed. Non-recurrent congestions are harder to incorporate, where Google and Yandex use real-time GPS data from users to detect these events.

WEATHER FACTORS

There are different weather factors that may influence the vehicle operating speed. In the USA, 15% of the congestions are caused by bad weather conditions where rain accounts for 70%[66]. Other weather factors that influence the operating speed are among others: snow, wind, visibility, road surface condition, temperature[61]. **Rain** seems to influence the operating speed by the intensity and road surface condition. A speed reduction between 1 and 10 km/h, due to the precipitation intensity, is found for urban road and highways[67][68][69]. A speed reduction, due to wet surface conditions, is found between 9.5 and 12 km/h on highways[65][70].

Like rain, also **snow** influences the operating speed based on the intensity and road surface conditions. Depending on the snowfall intensity, a speed reduction between 3 and 50 km/h can be obtained for different types of roads[67][69]. When the road surface is covered by snow, then a speed reduction between 10 and 16 km/h was found on highways[70][71].

The **wind speed** has an impact on the driver's speed when it is above a certain threshold, which was found to be 16 km/h for highways[70]. When the wind speed exceeds 24 km/h, then it has a larger effect on the vehicle speed. However, this seems to have large variations and depends on the type of driver making the influence of wind on the driving speed hard to incorporate. A wind speed between 24 and 48 km/h reduces the average operating speed on highways by 11.7 km/h.

The **visibility**, caused by rain, snow, fog, dust or smoke, influences the operating above a certain threshold. Kyte et al.[71] found a speed reduction of 14 km/h when the visibility was less than 0.16 km. While Kyte et al.[70] found a speed reduction of 0.77 km/h for every 0.01 km below a visibility of 0.28 km on highways.

ROAD ENGINEERING FACTORS

Road engineering factors entail the characteristics that can be assigned to roads and are also called road attributes. Road attributes are fixed and are different for each road segment. There are many attributes that can

be assigned to roads from which the influence of some are described in literature. The road engineering factors that influence the operating speed and travel time that are found in literature are described below:

- **Road category:** The road category to which a road can be assigned to is an important factor, since other road engineering factors depend on this. Examples of road categories are highway, arterial road, side road and local road[62].
- **Speed limit:** The speed limit is strongly correlated to the road category[62]. For example, a highway allows a much higher speed than a local road. The speed limit highly influences the driving speed, since drivers want to be as fast as possible, while not exceeding the speed limit which might lead to penalties and undesired consequences[61].
- **Tunnels and bridges:** In tunnels, drivers are more alert and cautious due to the closed environment. Typically, this results in a speed reduction and drivers who try to stay away from the tunnel walls[72]. On bridges, the wind speed and oscillations may have an impact on the driving speed. The experienced discomfort results in a decrease of speed by the driver[73].
- **Mountain passes:** Mountainous regions can considerably slow down the vehicle speed, while the maximum allowed speed is higher. The main reasons for the reduction in speed is the limited sight distance, amount of turns, radius of turns, road gradient and limited road width[74].
- **Intersection:** Intersections highly affect the operating speed and the travel time, since they break up the road section and requires attention and time to perform a collision-free maneuver. There are many types of intersections and how they are controlled[75]. General types of intersections are among others roundabouts, T-intersection and cross-intersections. Each type of intersection is controlled by either a stop sign, give way sign, traffic signal or no sign or signal. Both the type of intersection and the way the intersection is controlled have a different influence on the travel time.
- **Country/region:** The country or region may considerably affect the driven speed. Reasons for this are among others: different speed limits, type of roads, infrastructure, population, traffic density, traffic rules, etc. All of these factors that differ per country may effect the difference in driving speed and thus the travel time per country.
- **Road gradient:** When going uphill, the driver's visibility is restricted, which leads to a speed reduction to be able to anticipate to uncertain situations. Besides the visibility, also the gravity plays a role in the operating speed which decreases the speed when going uphill and increases the speed when going downhill[76]. For both cases, the drivers seem not to compensate for these changes in speed compared to speed limit. The influence of the road gradient on heavy vehicle speeds in Germany is significant. A decrease of 40-60% is measured on highways for a positive gradient of 6% [68]. A gradient of 2% results in a decrease of 10-20% in operating speed. In urban areas, the gradient has not a significant impact on the average speed.
- **Speed humps:** Speed humps, which have a maximum length of 3.5 m and a height of about 0.15 m, decrease the driving speed by approximately 10 km/h[76]. Short humps up to 2 meters have a low impact on the speed, while humps with a length between 2 and 3 meters and a low driving speed, causes high discomfort and as a result a decreases in speed. Speed bumps are a more aggressive form of speed humps and are in general used on parking lots and private terrains. This is because of the substantial discomfort which may lead to damage to the vehicle suspension or loss of control at too high speeds.
- **Horizontal curve:** Different researches show that a decrease in curve radius decreases the operating speed due to extra effort required to stay on the road and reduction in visibility[76]. Ben-Bassat & Shinar [77] showed that for sharp (radius = 80 m) and shallow (radius = 380 m) turns the speed decreased from 120 km/h to 80 and 100 km/h respectively. Shallam & Ahmed [78] performed a study at the Shillong bypass (highway) for 10 different curves. From this study it was concluded that curves with a radius lower than 100 m has the largest impact on the speed. For low radius curves, the speed reduced from 80 km/h to 43-56 km/h.
- **Road width:** The driving speed seems to associate with the perceived usable road width by the driver[79]. A smaller perceived road width results in a lower driving speed. The perceived road width is influenced

by the lane width, number of lanes, shoulder width, presence of parked cars and vertical elements on the road side.

- **Road surface roughness:** The roughness of the road surface causes noise and vibration leading to discomfort of the driver and a reduction in speed[76]. For example, the driving speed on asphaltic concrete is higher than on brick roads due to the increased roughness. Therefore it is one of the most important factors that influences the driving speed.
- **Turn cost:** In general, the time needed to perform a left turn, right turn or going straight at an intersection is different. Performing a left turn takes most time due to the oncoming traffic that need to be crossed. Jensen & Larsen [4] show an example where a left turn takes considerably more time (39 sec) than a right turn (19 sec) or going straight (20 sec). This is not an average case for all intersections due to the influence of different factors. Examples are: presence of traffic signals, the timing of traffic signals, number of lanes, etc.

UNPREDICTABLE FACTORS

There are many unpredictable factors that influence the operating speed and thus the travel time of a vehicle. Because they are unpredictable, it is hard to impossible to take them into account for the prediction of the operating speed. One unpredictable factor has a larger influence than the other on the travel time, from which some of the most influential unpredictable factors are listed below[62]:

- **Driver attitude and behavior:** The driver attitude and behavior differs form one person to the other. They can be distinguished by drivers that stick to the speed limit and those that drive above the speed limit and are called the risk takers or socially deviant drivers[80][81]. Reasons for driving above the speed limit are: finding the speeds limit to low, selfishness, risk-taking, forgetting to pay attention to the speed limit and being afraid of being to late[80][82].
- **Non-recurrent congestions:** Where some congestions are predictable, for example due to rush hours throughout the mid-week, some are not. Reasons for these are: bad weather conditions, incidents, work-zones and other special events[66].
- **Road accidents:** Obviously, road accidents cannot be predicted beforehand. The results differ from small congestions to road closures, where the latter has a large influence on the travel time.
- **Unscheduled road closure:** Unscheduled closures may result in large travel time delays and are caused by severe accidents, pavement subsidence, fall of a bridge, an avalanche and other natural disasters[62].
- **Delay on customs/border control:** Delays on custom and border controls may vary a lot depending on the country and location. This type of delay is mostly applicable to routes that pass borders between an EU and a non-EU country[62].

4.4.2. FEATURE EXTRACTION FROM MAP DATA

In subsection 4.4.1, a literature study was conducted about travel speed indicators which influence the travel time. These influential factors can be categorized into five classes: vehicle classification, temporal, weather, road engineering and unpredictable factors. The influence of the following factors on the travel speed are considered to be out of scope for this research study:

- **Vehicle classification:** Since the collected GPS data is only from one type of truck, other vehicles such as cars and delivery vans are out of scope for this research. Therefore, the vehicle classification will not be taken into account as independent variable This results in a model that is only applicable to trucks.
- **Weather:** The weather is considered to be out of scope for this research due to a high additional workload. This is because no information about the weather is included in the GPS data. An external data source such as KNMI [83] needs to be consulted to link each GPS point to the weather at that moment in time.
- **Unpredictable factors:** Obviously, the unpredictable factors cannot be taken into account as independent variable, since these factors are unknown. However, it is important to keep in mind that these unpredictable factors cause a certain amount of inaccuracy which limits the improvement of the travel time predictions to a certain unknown accuracy.

The influential factors that will be included in the model are road engineering and temporal factors. The road engineering factors are included by using them as independent variable and can be obtained from the map data. This is explained below. The feature extraction of the temporal factor is explained in subsection 4.4.3. The temporal factor is researched at the end of this research after the best model is found using only road engineering factors. This is because the temporal factor cannot be included in the model as independent variable. This is explained in more detail in subsection 4.4.3.

ROAD ENGINEERING FACTORS

The road engineering factors can be obtained from the map data from *HERE*. These will be the independent variables of the speed prediction model. The majority of these factors are shown in Appendix B and used by den Heijer. In Table 4.1, an overview can be found of the road engineering factors that were found in literature. Also, road engineering factors are added that were not found in literature, but are expected to be appropriate predictors of the speed. In the second column, the map data that can be used, to include each road engineering factor, is provided. In the third column, it is indicated whether the factor can and will be applied to the model. A discussion of each road engineering is provided below including how and whether they will be used as model input.

Table 4.1: Overview of road engineering factors found in literature and other from *HERE*. It is indicated what map data of *HERE* can be used for the factors in the model and whether the factor will be applied.

Road Engineering Factors from Literature	***HERE* map data**	**Applied (y/n)**
Road Category	*Functional Class*	y
Speed Limit	*Speed Limit*	y
Tunnel	*Tunnel*	y
Bridge	*Bridge*	y
Mountain Passes	*Mountain Passes*	n
Intersection	*Traffic Signal*	y
Country/Region	*(Non) Urban*	y
Road Gradient	*Grade Category*	n
Speed bumps	*Speed Bumps*	y
Horizontal Curve	-	n
Road Width	*Road Width & Lane Category*	y
Road Surface Roughness	*Paved*	y
Turn Cost	-	n
Other Road Engineering Factors from *HERE*	***HERE* map data**	**Applied (y/n)**
Road Length	*Road Length*	y
Ramp	*Ramp*	y
Priority Road	*Priority Road*	y
Speed Pattern	*Speed Pattern Maximum, Minimum, Average & Monday 8:30*	y

- **Road Category:** *HERE* does not provide road categories in a form such as highway, local road, side road, arterial road, etc. However, *HERE* provides a functional class to each road between 1 and 5, which is also a categorization of the roads:
  - **Functional Class = 1:** roads allow for high volume, maximum speed traffic movement between and through major metropolitan areas.
  - **Functional Class = 2:** roads with very few, if any, speed changes that allow for high volume, high speed traffic movement.
  - **Functional Class = 3:** roads that intersect Functional Class = 2 roads and provide a high volume of traffic movement at a lower level of mobility than Functional Class = 2 roads.
  - **Functional Class = 4:** roads that provide for a high volume of traffic movement at moderate speeds between neighbourhoods. These roads connect with higher Functional Class roads to collect and distribute traffic between neighbourhoods.

  - **Functional Class = 5:** roads with volume and traffic movement below the level of any other Functional Class.

- **Speed Limit:** The speed limit indicates the maximum allowed driven speed on a road and is provided by *HERE* for each road and ranges between 5 and 130 km/h.
- **Tunnel/Bridge:** If a road in the map is a tunnel or a bridge then *HERE* assigns the attribute tunnel = yes or bridge = yes to the road.
- **Mountain Passes:** *HERE* indicates whether a road is a mountainous pass yes or no. However, since the research scope is the Netherlands, which does not have any mountain passes, this factor will be disregarded and not applied to the model.
- **Intersection:** *HERE* does not provide clear information about the type of intersections. The only attribute that can be used that contains useful information about the intersection is whether the road has a traffic light yes or no. If the road has a traffic light, then it is expected that the average speed on this road will be lower.
- **Country/Region:** The region can be taken into account by using the attribute whether a road is in an urban area yes or no. This might have a significant influence on the speed due to a higher vehicle density in urban areas.
- **Road Gradient:** The road gradient can be taken into account by the Grade Category provided by *HERE* and can be either up, level or down. However, due to the flatness of the Netherlands, Belgium and Luxembourg, this factor is not taken into account.
- **Speed Bumps:** *HERE* provides data whether a road has speed bumps yes or no.
- **Horizontal Curve:** The horizontal curve can be incorporated by adding an attribute to the road about whether the road is a horizontal curve yes or no. However, whether a road is a horizontal curve cannot be derived from the *HERE* map data and is therefore not taken into account.
- **Road Width:** The road width of the roads is provided by *HERE* and ranges between 2.50 and 3.50 m. Another attribute that is related to the road width is lane category and ranges between 1 and 3.
  - Lane category 1 contains roads with 1 lane
  - Lane category 2 contains roads with either 2 or 3 lanes
  - Lane category 3 contains roads with 4 or more lanes
- **Road Surface Roughness:** The road surface roughness can be described by whether the road is paved yes or no.
- **Turn Cost:** The turn cost is the time it takes to go left, right or straight at an intersection. Since only one speed can be assigned to an edge in the map, different speeds for going either left, right or straight cannot be assigned to the roads. Therefore, the turn cost cannot be incorporated in this research study.
- **Road Length:** Each road in the map has a road length assigned by *HERE* in meters. A longer road means less intersections which might increase the average driven speed and is therefore incorporated as independent variable.
- **Ramp:** *HERE* indicates whether a road is a ramp yes or no. This might give additional information about the driven speed, since vehicles accelerate or decelerate on ramps reducing the average driven speed.
- **Priority Road:** *HERE* indicates whether a road is a priority road yes or no. This might improve the accuracy of the speed prediction, since the vehicle does not need to stop to give priority to other vehicles which increases the average speed.
- **Speed Pattern:** *HERE* provides speed patterns of cars for each road which contain the minimum speed, maximum speed, average speed and the average speed on Monday 8:30. Den Heijer [1] researched whether the average speed of cars, obtained from *HERE* map data, could be used in the map to improve the travel time predictions. However, these speeds did not improve the travel time prediction accuracy

of trucks. Still, these speed patterns can be a good indicator for the model to predict the average truck speed on each road.

Point-Based and Trip-Based Data
There are different ways to learn from the GPS data and road engineering factors to improve the speed predictions. Den Heijer [1] used two types of training data, which are point- and trip-based data and are explained in subsection 2.3.1. The difference between these two types of data is that the point-based data only contains information about a few roads, where the GPS data is recorded, of each travel. The advantage of the trip-based data is that it contains information about all roads of each travel. Each trip-based data point is based on all roads between two consecutive GPS points. This needs an additional preprocessing step. This step includes the calculation of the average of the road properties (independent variables) of all roads that belong to each trip. This calculation step is explained in more detail in subsection 2.3.1.

In den Heijer's research[1], point-based data produced better speed predictions than trip-based data, such that a higher travel time prediction accuracy was obtained. However, trip-based data was only used as training data for linear regression models, since this data cannot be used for random forest models, as explained in subsection 2.2.2. Therefore, both point- and trip-based data will be used in this research, because another prediction method may perform better with trip-based data than point-based data.

4.4.3. FEATURE EXTRACTION TEMPORAL FACTOR
The temporal factors can be extracted from the time and date of the recorded GPS data. Unfortunately, the Highway Node Routing used by ORTEC is static and therefore only one speed value can be assigned to each road in the map. This means that the temporal factor cannot be included as independent variable for the model input. Therefore, a different method has to be used to research the influence of temporal factors on the travel time prediction accuracy. This can be done by splitting the data set into sub data sets, where each sub data set contains data that belongs to a certain time window such as rush hours (7:00 - 9:00 & 16:30 - 18:30), non-rush hours, day of the week, season, etc. For this research, it is decided to research the effect of dividing the data into rush and non-rush hours. This seems to be the most influencing temporal factor due to the many traffic jams at these moments of the day. An example, visualized through a flowchart, of this method with sub data sets 'rush hours' and 'non-rush hours' is shown in Figure 4.5. This is a simplification of Figure 3.4. The influence of the temporal factor is researched at the end of this research after the best model is found using only road engineering factors.

4.5. DATA SAMPLING
After the collection and preprocessing of the data sets, the data sets are randomly split into a training and a test data set. The training set will be used to train the model and the test set to evaluate the model on points/trips and complete travels. 75% of the full data set will be used as training set and 25% as test set, which is a common ratio used in many researches. Only complete travels, containing a sequence of GPS points, will be put in either the training or test data set. This is done to make sure that during the training process nothing is seen from the travels that will be evaluated on. For both the point- and trip-based data, the points and trips from the same travels are put in the training and test set. In this way, the models that are trained on point- and trip-based models are evaluated on the same travels.

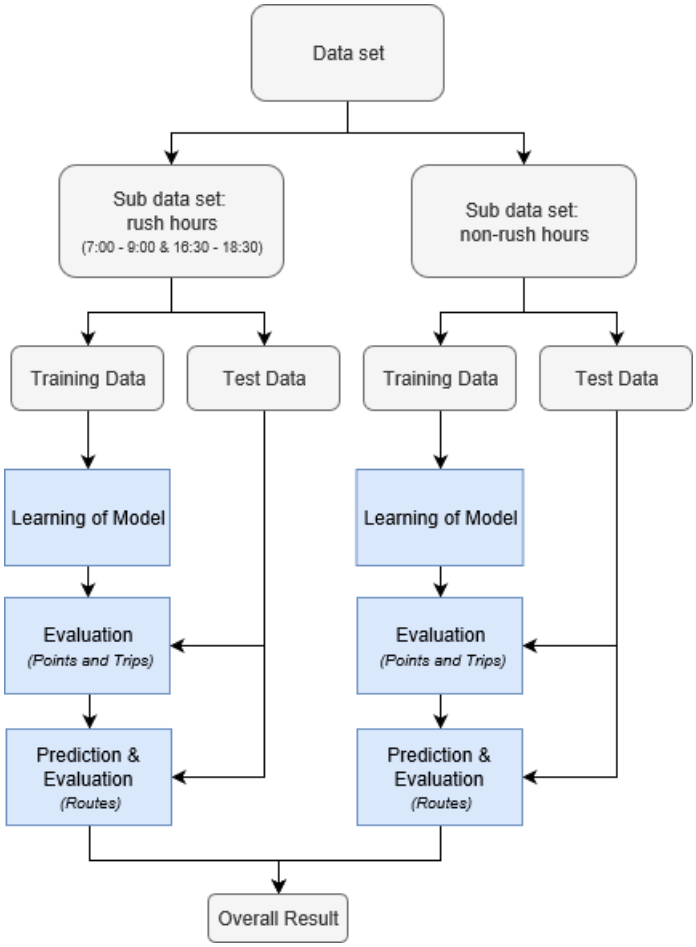

Figure 4.5: Example of how temporal factors can be researched with a static routing algorithm. This example shows a split in the data set for rush and non-rush hours.

4.6. CONCLUSION

In this chapter, the following conclusion could be made:

- The data that is used as model input, comes from map data and GPS data. The map data comes from map supplier *HERE* and contains information about the road network and many properties of each road. The GPS data comes from two different customers of ORTEC who operate with trucks. One customer operates in the Netherlands and has a data frequency of 2 minutes. The other customer operates in the Benelux with a data frequency of 5 minutes and is called. The 2 minute frequency data is called 'new data' and is collected and preprocessed during this research. The 5 minute frequency data is called 'old data' and is collected and preprocessed by den Heijer.
- The new data set is used due to a higher frequency than the old data set. It is expected that the quality of the trip-based data increases, since it can be more accurately estimated how the truck has driven between two GPS points. This may result in a higher travel time prediction accuracy. The old data is also used to get insight whether both the new and old data have the same best speed prediction model. Also, more knowledge can be obtained whether a neural network, trained on trip-based data, may already outperform the benchmarks with a data frequency of 5 minutes.
- After the collection of raw GPS data, the data needs to be cleaned so the model only learns from representative data. After this, the GPS data can be matched to locations in the digital map. This is called map matching. Different algorithms are available in literature to improve the map matching accuracy. These algorithms are categorized into: geometric, topological, probabilistic and advanced. To save a significant amount of time on the development and implementation of a map matching algorithm, the online map matching service from *HERE* is used.

- After each GPS point is matched to a road in the digital map, the factors (road properties) that can be used to predict the speeds, are extracted. First, a literature review was done to find out which factors influence the speed or travel time. These factors can be divided into five categories: type of vehicle, temporal, weather, road engineering and unpredictable factors. For this research, the weather and unpredictable factors are out of scope. The type of vehicle is trucks, since the GPS data is only obtained from customers with trucks. The road engineering factors can be obtained from the map data of *HERE*. These factors are used to train the model and to predict the speeds in the map. For the trip-based data, an additional computational step need to be performed. This is because the average of the road properties of the roads, that are between two GPS data points (trip), has to be calculated.
- Lastly, the data sets are randomly split into a training (75%) and test (25%) set. Complete travels are put in either the training or the test set, which includes all points and trips from the same travel. This is done to make sure that during the training process nothing is seen from the travels that will be evaluated on.

5

Model Design

In this chapter, the model design of the neural network will be discussed. First the objective of the model will be discussed, including the error function used to measure the travel time prediction accuracy. Secondly, the independent and dependent variables, which are the input and the output of the model, are discussed. After this, the model design choices are made. This includes the values of the hyperparameters such as loss function, number of neurons and hidden layers, optimizer, etc. After the neural network is implemented in Python, the hyperparameter process of the neural network is discussed.

5.1. Model Objective

The overall objective of the model is to predict the speed of each road in the digital map such that the travel time prediction accuracy of the current speed prediction models, developed by den Heijer, is outperformed for a given road network. This will be done by learning from points and trips obtained from GPS data, which contain valuable information about the driven speed, and the travel time, at many different locations in the road network.

Travel Time Prediction Accuracy

To compare the travel time prediction accuracy of the new and current speed prediction models, an error function is needed. The advantage of an error function is that the performance of the model can be summarized in just one value. This allows for an easy comparison between the models in a quantitative way. There are several error functions available which can be used to measure the performance. Armstrong & Collopy [84] proposed five primary criteria to find an appropriate error function and are the following:

- **Reliability:** The reliability of an error function indicates whether a repeated application of a procedure will produce similar results for different samples.
- **Construct validity:** Construct validity indicates in what extent the error function assesses the 'accuracy' of the model.
- **Outlier protection:** The influence of outliers, due to a much larger or smaller error, on the error function should be kept to a minimum. Outliers may occur due to a mistake in recording the data or a situation which is far from average.
- **Sensitivity:** It is important for calibration that the influence, even for a small change in parameters, on the model performance is understood.
- **Relationship to decisions:** To what extent can the error be used to support decision-making.

Other criteria from Armstrong & Collopy [84] are the **computational effort** and **understandability**. The computational effort is for all error functions not an issue due to the current computer capabilities. Also, the understandability of the common error functions is, except for squared error terms, relatively easy.

Another criterion that was not proposed by Armstrong & Collopy [84] is that the error function has to give relatively **equal penalties** for small and large values. This is important for this research study and can be

solved by measuring the error with respect to the actual value. For example, if you have a predicted travel time of 10 minutes and the actual travel time is 9 minutes, then the error should be the same as for a predicted travel time of 90 minutes with an actual travel time of 81 minutes. By dividing the error by the actual value, both predicted travel times have an error of 10%.

In Figure 5.1, an overview of different error functions is provided. The five primary criteria that are included are: reliability, construct validity, outlier protection, sensitivity and relationship to decisions. The ratings that are given to the error functions with respect to the criteria are poor, fair and good. Reliability and construct validity are based on empirical results, while the other criteria are based on subjective judgements. From the table, it can be concluded that no error function performs well for all criteria. The reliability and construct validity is fair and good for all error functions, except for the RMSE. Thus when choosing an error function, these two criteria do not have a significant impact on the decision.

Error measure	Reliability	Construct validity	Outlier protection	Sensitivity	Relationship to decisions
RMSE	poor	fair	poor	good	good
Percent Better	good	fair	good	poor	poor
MAPE	fair	good	poor	good	fair
MdAPE	fair	good	good	poor	fair
GMRAE	fair	good	fair	good	poor
MdRAE	fair	good	good	poor	poor

Figure 5.1: Ratings of error measures[84].

Most travel time prediction methods found in literature (section 3.1) use the mean absolute percentage error (MAPE). This error function is shown in Equation 5.1.1. Where n is the total number of travels, A_t the actual travel time and P_t the predicted travel time for travel t. This error function calculates the mean of all absolute comparisons between the actual and predicted travel time with respect to the actual travel time. However, this error function has mainly two drawbacks[85]. Firstly, the MAPE is sensitive for outliers, which means that some bad recorded data influence the MAPE negatively. Secondly, the MAPE uses a non-symmetric loss, which means that positive errors are more penalized than negative errors. Makridakis [86] showed an example where for the first case $A_t = 150$ and $P_t = 100$ and for the second case $A_t = 100$ and $P_t = 150$. This results in a MAPE of 33.33% and 50% for the first and second case respectively and shows the effect of non-symmetric loss. Also, by using a symmetric error function, large errors, when the actual value is close to 0, are avoided. With a symmetric error function, these problems can be mitigated.

$$\text{MAPE} = \frac{100\%}{n} \cdot \sum_{t=1}^{n} \left| \frac{A_t - P_t}{A_t} \right| \tag{5.1.1}$$

The sensitivity to outliers can be solved by taking the median instead of the mean of the absolute percentage errors. This results in the median absolute percentage error (MdAPE) function shown in Equation 5.1.2. In Figure 5.1 it can be seen that the 'outlier protection' improves from poor to good.

$$\text{MdAPE} = \text{median}\left(100\% \cdot \left| \frac{A_t - P_t}{A_t} \right|\right) \qquad t \in 1, \ldots, n \tag{5.1.2}$$

The second problem, non-symmetric loss, can be solved by using the symmetric median absolute percentage error (sMdAPE) function shown in Equation 5.1.3. The difference between A_t and P_t is now divided by the average of A_t and P_t. In this way, the error function becomes symmetric with an error that ranges between 0 and 200%. The sMdAPE function will be used as main indicator for the travel time prediction accuracy in this research study. This is the same error function as used by den Heijer, which also allows to make a comparison with den Heijer's models. For clarification in this report, the subscript TT is added to the sMdAPE to indicate that this error function is used to evaluate the travel times.

$$\text{sMdAPE}_{\text{TT}} = \text{median}\left(200\% \cdot \left| \frac{A_t - P_t}{A_t + P_t} \right|\right) \qquad t \in 1, \ldots, n \tag{5.1.3}$$

To see whether the travel time predictions are under- or overestimated, den Heijer [1] introduced a symmetric median percentage error (sMdPE) function (Equation 5.1.4). This function is similar to the sMdAPE, but it does not consider the absolute values. A negative sMdPE means that the travel time predictions are too fast, whereas a positive value means that the travel time predictions are too slow. In this way, the $\text{sMdAPE}_{\text{TT}}$ can be later improved by moving the sMdPE to 0%. Also for the sMdPE, the subscript *TT* is added to indicate that this error function is used to evaluate the travel times.

$$\text{sMdPE}_{\text{TT}} = \text{median}\left(200\% \cdot \frac{P_t - A_t}{A_t + P_t}\right) \qquad \text{t} \in 1, ..., n \tag{5.1.4}$$

The IQR sPE, inter-quartile range of the symmetric percentage error, is also included as error measure to get an idea how much the travel time prediction errors are spread. This error function is included, because a model can be accurate (low $\text{sMdAPE}_{\text{TT}}$), but may have errors that are widely spread (large IQR sPE_{TT}). The IQR is the difference between the upper and lower quartiles of the median and is shown in Figure 5.2. If for example the IQR sPE is 40%, then the difference between 25% of the errors below the sMdPE and 25% above the sMdPE is in a range of 40%. In Equation 5.1.5, the formula of the IQR sPE is shown, where the formula of the sPE_{TT} is shown in Equation 5.1.6. The subscript *TT* is added to indicate that this error function is evaluated on the travel times.

$$\text{IQR sPE}_{\text{TT}} = \text{Q3 sPE}_{\text{TT}} - \text{Q1 sPE}_{\text{TT}} \tag{5.1.5}$$

$$\text{sPE}_{\text{TT}_\text{t}} = 200\% \cdot \frac{P_t - A_t}{A_t + P_t} \qquad \text{t} \in 1, ..., n \tag{5.1.6}$$

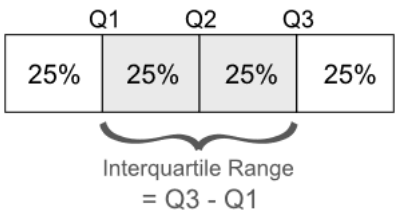

Figure 5.2: Visualization and formula of the inter quartile range (IQR).

5.2. INDEPENDENT AND DEPENDENT VARIABLES

The independent and dependent variables, which are the input and output of the neural network model, are shown in Table 5.1. In the second and third column, the range of the values for the point- and trip-based data is shown. As can be seen, most independent variables for the point-based data are Boolean variables, where 0 is 'False' and 1 is 'True'. For the trip-based data, these Boolean variables change to continuous variables ranging between 0 and 1, since it is the average of multiple roads. The dependent variable is either speed, logspeed or pace. The speed and logspeed will be used for the point-based data and the speed, logspeed and pace for the trip-based data. After training and prediction, the predicted logspeed and pace are converted back to speed. The number of independent variables used for the neural network are less than used by den Heijer. This is because only relevant independent variables are used in the neural network model based on literature. The road engineering factors, which are used by den Heijer as independent variables, are shown in Appendix B.

Table 5.1: Independent and dependent variables of the neural network model.

Independent Variables	**Range Points**	**Range Trips**
Functional Class 1	0 or 1	[0,1]
Functional Class 2	0 or 1	[0,1]
Functional Class 3	0 or 1	[0,1]
Functional Class 4	0 or 1	[0,1]
Speed Limit	[5,130]	[5,130]
Tunnel	0 or 1	[0,1]
Bridge	0 or 1	[0,1]
Traffic Signal	0 or 1	[0,1]
Urban	0 or 1	[0,1]
Speed Bumps	0 or 1	[0,1]
Road Width	> 0	> 0
Lane Category 1	0 or 1	[0,1]
Lane Category 2	0 or 1	[0,1]
Paved	0 or 1	[0,1]
Road Length	> 0	> 0
Ramp	0 or 1	[0,1]
Priority Road	0 or 1	[0,1]
Speed Pattern Max	[0,130]	[0,130]
Speed Pattern Min	[0,130]	[0,130]
Speed Pattern Avg	[0,130]	[0,130]
Speed Pattern Monday 8:30	[0,130]	[0,130]
Dependent Variables	**Range Points**	**Range Trips**
Speed	≥ 0	≥ 0
Logspeed	≥ 0	≥ 0
Pace	> 0	> 0

5.2.1. DUMMY VARIABLES

In Table 5.1, two categorical variables can be found, which are the Functional Class and Lane Category. To use these variables by the model, these variables have to be changed to numeric dummy variables, which are understood by the model. This means that the Lane Category is changed to dummy variables Lane Category 1, Lane Category 2 and Lane Category 3, which have a value between 0 and 1. For example, if a road belongs to Lane Category 1, then Lane Category 1 = 1, Lane Category 2 = 0 and Lane Category 3 = 0. To avoid a dummy variable trap, one dummy variable needs to be removed. For this model, Lane Category 3 is removed which is also excluded from the set of independent variables shown in Table 5.1. A dummy variable trap can be explained by Equation 5.2.1, which shows the multi-collinearity between the dummy variables. This example shows that Lane Category 1 can be predicted from the other two Lane Categories. The same is true for the Functional Class, where the Functional Class 5 is not included in the set of independent variables to avoid another dummy variable trap.

$$\text{Lane Category 1} = 1 - \text{Lane Category 2} - \text{Lane Category 3} \tag{5.2.1}$$

5.2.2. DEPENDENT VARIABLES

The main goal of this research is to improve the travel time predictions from point A to B. However, this can only be done by predicting the speed of each road in the map in such a way, that the travel time prediction accuracy will be improved. Because the driven speed is obtained from the GPS data, an obvious choice is to train the neural network on the dependent variable speed. However, den Heijer came up with three other dependent variables which can also be used to train the model. These have all their own benefits and have been explained in subsection 2.3.2. These dependent variables are logspeed, time and pace and are converted back to speed after training and prediction of the model. The choices for the dependent variables used in this research, for point-based and trip-based data, are explained below.

Dependent variables for point-based data

- **Speed:** An obvious choice is to train the neural network model on the dependent variable speed. This is because the speed of each road in the map needs to be improved to improve the travel time prediction accuracy. Also, the collected GPS data contains the driven speed. Therefore, the speed is a logical dependent variable to train the model on.
- **Logspeed:** Non-relative loss functions, such as the mean squared error (MSE) and mean absolute error (MAE), fit better to larger values. Two examples that describe the drawback of these loss functions are discussed in subsection 2.3.2. To avoid this problem, the logarithmic speed can be used, which moves the speeds to a relative space. Moving to the logarithmic speed gives the same relative error for low and high speeds. However, moving the speeds to a logarithmic space causes some problems when the speed is equal to 0 km/h. This is because log(0) is equal to minus infinity resulting in extremely large errors. To avoid this problem, all speeds are clipped to 1 km/h before conversion to logarithmic space.

  Eventually, training on the speed and logspeed have both their own benefits. Both will be tested to see which dependent variable is preferred and result in the best travel time predictions.

Dependent variables for trip-based data

- **Pace:** Besides the speed, another logical choice for the dependent variable is the time. This is because in the end, the travel time predictions have to be improved. Using time as dependent variable is possible for trip-based data, since the time between two consecutive GPS points (trip) can be derived from the GPS data. Since the distance and time between two consecutive GPS points contain multiple roads, the independent variables have to be weighted in some way. This can be done by multiplying each independent variable by the distance in meters that it occurs in the trip. This is shown in Equation 5.2.2, for 2 independent variables. However, using the time as dependent variable requires the model to extrapolate. This is because the independent variables that are trained on, have a (much) larger value than the independent variables that are used to predict the speed for single roads (Equation 5.2.3). This is because the independent variables are multiplied by 'meters driven', which is significantly less for single roads compared to trips.

$$\begin{aligned}[\text{trip time (multiple roads)}] = f(&\text{speed bumps} \cdot [\text{meters driven on speed bumps}], ..., \\ &\text{bridge} \cdot [\text{meters driven on bridge}])\end{aligned} \tag{5.2.2}$$

$$\begin{aligned}[\text{trip time (single road)}] = f(&\text{speed bumps} \cdot [\text{road length in meters}], ..., \\ &\text{bridge} \cdot [\text{road length in meters}])\end{aligned} \tag{5.2.3}$$

  An alternative way of training the model is to use 'fraction of meters' with respect to the full trip. By using 'fraction of meters', the range of values of the independent variables are the same for training and prediction. These values will range between 0 and 1 for training (trips) and is either 0 or 1 (Boolean) for prediction (single road). The conversion to 'fraction of meters' can be simply applied by dividing each independent and dependent variable by the 'total meters driven'. In Equation 5.2.4, the conversion is shown. This results in a dependent variable dimension of time/length (1/speed), which is called the average *pace*. An advantage of moving to the pace is that the data is normalized. This means that long trip times are not favored by the non-relative loss functions.

$$\begin{aligned}[\text{average pace}] = f(&\text{speed bumps} \cdot [\text{fraction of meters on speed bumps}], ..., \\ &\text{bridge} \cdot [\text{fraction of meters on bridge}])\end{aligned} \tag{5.2.4}$$

- **Speed:** The speed is, as with point-based data, an obvious choice as dependent variable. To derive the average speed from equation Equation 5.2.2, the independent variables have to be applied by the 'fraction of seconds' of the total trip time. However, these 'fraction of seconds' are not known due to the lack of information between two GPS points. Therefore, an alternative approximation will be used which is the 'fraction of meters'. This is shown in Equation 5.2.5.

$$[\text{average speed}] = f(\text{speed bumps} \cdot [\text{fraction of meters on speed bumps}], ..., \text{bridge} \cdot [\text{fraction of meters on bridge}]) \tag{5.2.5}$$

- **Logspeed:** Also, when using trip-based data, the model can be optimized by non-relative loss functions such as MSE and MAE. By optimizing the model with a non-relative loss function, larger actual values are favored. Therefore, also the logspeed will be used as dependent variable where the independent variables are multiplied by the 'fraction of meters'.

5.2.3. NEURAL NETWORK ARCHITECTURE

A visualization of the independent and dependent variables, which are summarized in Table 5.10, is shown in Figure 5.3. This is the architecture of the neural network. The number of neurons n and hidden layers m still need to be determined through hyperparameter tuning. The dependent variable or output is either speed, logspeed or pace.

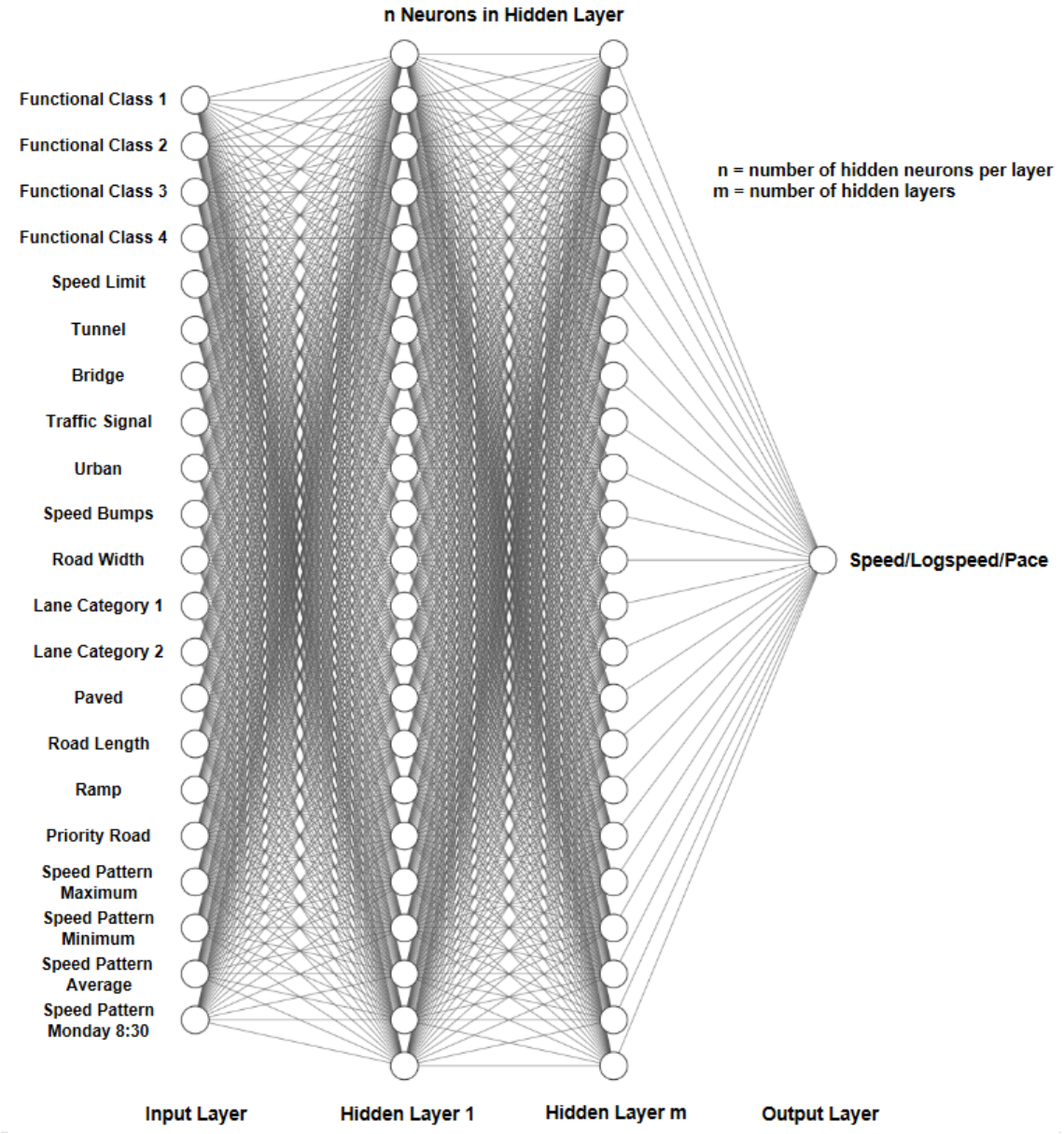

Figure 5.3: Neural Network architecture for speed, logspeed and pace prediction.

5.3. MODEL DESIGN CHOICES

For the neural network, several model design choices have to be made. These design choices are either a method or a hyperparameter. A hyperparameter is a parameter that is not learned by the model and needs to be chosen and/or tuned by the user. The model design choices that have been made for the neural network are described in the subsections below.

5.3.1. REGULARIZATION METHOD

Regularization is an additional method that can be applied to the optimization process of the neural network. A regularization method allows to find the most appropriate values for the weights and biases to get the best generalization error. The best generalization error is found when the model is neither under- or over-fitted. The most popular regularization techniques for the neural network are Early Stopping, L1 & L2 and Dropout and are discussed below:

- **Early stopping:** As the name suggest, the training process of the model is stopped before it has finished the full training process. If the full training process would be executed, then there is a higher chance of over-fitting and a higher generalization error at the test set. For early stopping, a validation dataset is used that represents the test dataset. It is used to stop the training process before the model is over-fitted. A visualization of this process is shown in Figure 5.4.

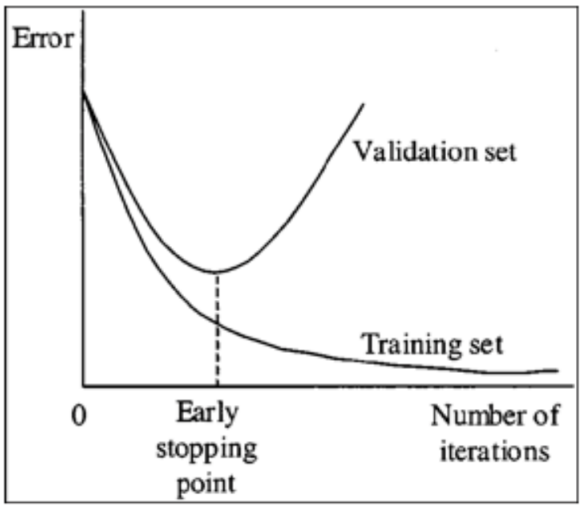

Figure 5.4: Early stopping[87].

- **L1 & L2:** The L1 and L2 regularization method are similar in a way that both methods lead to smaller weights. However, L2 regularization reduces the weight proportional to the value of the weight. This means that a larger weight value leads to a faster weight value reduction than a smaller weight value. In L1 regularization, the weight value is reduced independent of the value and is a fixed value for each iteration. Therefore, L1 regularization leads to more weight values of zero and results in a 'sparse' network[87]. The idea behind L1 and L2 regularization is that lower weight values lead to a simpler and more stable networks. As a result, an improved generalization of the model is obtained. In Equation 5.3.1 and Equation 5.3.2, both equations for L1 and L2 regularization are shown. L is the loss function and is usually the Cross Entropy loss. λ is the regularization parameter and indicates the importance of the regularization term. n is the size of training data set and w_i the weight on each connection.

$$L_1 = L + \frac{\lambda}{n} \sum_w |w_i| \tag{5.3.1}$$

$$L_2 = L + \frac{\lambda}{2n} \sum_w w_i^2 \tag{5.3.2}$$

- **Dropout:** The dropout method randomly ignores certain neurons in the NN at each iteration. This is applied throughout the entire training process and to the neurons present in the input and hidden layer of the NN. By doing this, neurons that are insignificant are identified and get assigned a lower weight than other ones or vice versa. A visualization of this process is shown in Figure 5.5. The neurons with a cross represent the ignored neurons during a certain iteration. The dropout regularization method is effective due to two reasons[87]:
  - Since random neurons are dropped during the training process, neurons cannot blindly rely on other neurons to get the best result for the model. This means that each neuron has to perform well for different combinations and therefore gets robust. This generalizes the NN and improves the generalization accuracy.
  - An technique that is often used in ML is 'bagging' and is used to improve the prediction accuracy by combining different ML algorithms. The dropout regularization technique uses the same principle by using different NN models. At the end, an average prediction model is obtained. This turns out to be an effective method[88].

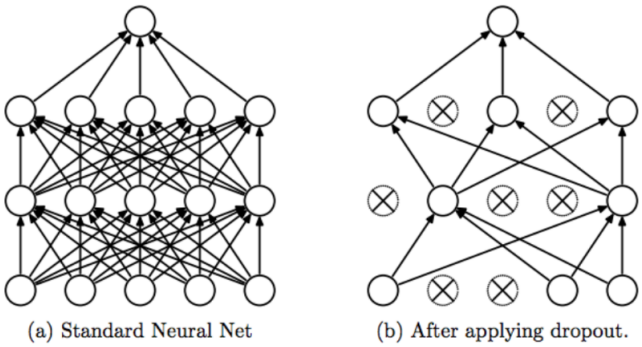

Figure 5.5: Dropout regularization[87].

For this research, Early Stopping will be used to avoid the neural network from under- and over-fitting. Early Stopping is chosen due to its effectiveness and simplicity[89]. Early Stopping uses both a training and validation set, where the validation set is used to measure the generalization of the model. If the model starts to degrade on the validation set, then the training process is stopped as visualized in Figure 5.4. Early stopping has one hyperparameter. This is the number of epochs (iterations) after which the model stops training when no improvement has been made on the validation set. The chosen number of epochs for the neural network in this research is 10. This number of epochs allows the model to get out of possible local minimum to find the global minimum. However, it also stops the training process to save unnecessary training time.

5.3.2. GRADIENT DESCENT METHOD

The optimization/training process of the NN is a time-consuming process. During the training process, the best values for w_i and b_i are found to minimize the loss function. The optimization process is done through a first order optimization algorithm, called 'gradient descent'. This algorithm uses the gradient of the loss function with respect to the weights and biases in the neural network. The gradient of each parameter with respect to the total error is sort of iterative and accumulative and therefore not easy to calculate. Therefore, back-propagation is first used to calculate these gradients before being used by the gradient descent method. Back-propagation propagates the total error from the output layer to the input layer through the connections and calculates the gradient for each weight and bias layer by layer.

The gradient descent method can be divided into three variants: batch gradient descent (BGD), stochastic gradient descent (SGD) and mini-batch gradient descent (MBGD). These methods are discussed below[90]:

- **Batch gradient descent:** The BGD updates the parameters based on the entire training data set. Therefore, this method requires a lot of computational time and is a problem for too large data sets due to memory issues. The formula for the gradient descent algorithm as well as the batch gradient descent is shown in Equation 5.3.3. Here, $\vec{\mathbf{x}}$ is the vector that contains all weights and biases of the model, η

is the learning rate, ∇ the gradient and L($\vec{\mathbf{x}}$) the loss function that should be minimized. The learning rate η, determines the step size of the update and therefore the accuracy and speed of the optimization process. In Figure 5.6, an example is shown of a big and small learning rate. A too big learning rate, as shown in the left image, causes the error to grow. While a too small learning rate requires much computational time to find the optimal solution.

$$\vec{\mathbf{x}} = \vec{\mathbf{x}} + \eta \cdot \nabla L(\vec{\mathbf{x}}) \tag{5.3.3}$$

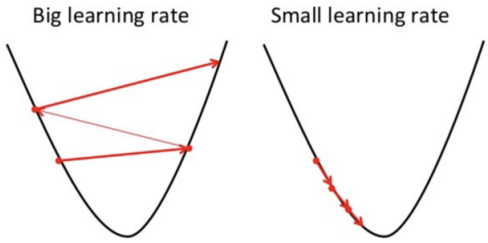

Figure 5.6: Example big and small learning rate[91].

- **Stochastic gradient descent:** The SGD updates the parameters after a single training example instead of a whole data set as done by BGD. This leads to a fluctuating error which means that two consecutive updates of the parameters may vary considerably. This is illustrated on the right in Figure 5.7. On the left the training error vs iterations of a standard gradient descent is shown. The convergence of this method might take some time due to the fluctuations, but can be mitigated by decreasing the learning rate. The stochastic behavior of the SGD allows to escape from a local minimum and to find the global minimum.

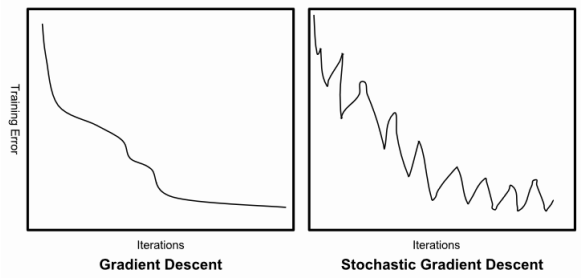

Figure 5.7: Comparison standard gradient descent and stochastic gradient descent[92].

- **Mini-batch gradient descent:** MBGD is a combination of the BGD and the SGD method. It updates all parameters after a batch of n data points instead of the whole data set or single training example. By using a batch, the convergence to the local or global minimum is more stable. The MBGD method is the most popular gradient descent method and almost always applied to NNs. A mini-batch size between 50 and 256 is mostly preferred, but depends on the application.

For this research the mini-batch gradient descent method will be used. This method allows a stable convergence to the local or global minimum compared to the stochastic method. Also, a reduced computational time is obtained compared to the batch gradient descent method. The hyperparameter that come along with this method is the mini-batch size. After each mini-batch, the weights and biases are updated in the neural network. Common mini-batch sizes that are used in literature are either 32, 64, 128, 256 or 512, which are all numbers of the power of two[93]. These mini-batch sizes are commonly chosen, because they fit well to the GPU or CPU memory requirements. In general, a smaller batch-size leads to a slow, but more accurate training process. A larger batch-size leads to faster convergence, but less accurate estimates. The right mini-batch size for the neural network will be found through hyperparameter tuning.

5.3.3. LOSS FUNCTION

To optimize the neural network, a loss function needs to be used. A loss function minimizes the error between the actual and predicted values to obtain the best prediction accuracy. To indicate that an error function is used as loss function, the subscript *LF* will be used. There are different loss functions available, the loss functions that will be used in this research are:

- **sMdAPE$_{LF}$**: In section 5.1, it was concluded that the sMdAPE$_{TT}$ will be used to evaluate the travel time predictions. Therefore, it would be logical to use the sMdAPE as loss function (sMdAPE$_{LF}$) to optimize the speed predictions. This loss function is shown in Equation 5.3.4, where A_t is the actual value of the dependent variable, P_t the predicted value of the dependent variable and n the number of observations. However, den Heijer[1] concluded that the speed and travel time prediction accuracy in sMdAPE$_{Speed}$ and sMdAPE$_{TT}$ are weakly correlated. A high speed prediction accuracy in sMdAPE$_{Speed}$, did not directly result in a high travel time prediction accuracy in sMdAPE$_{TT}$. In section 2.4, it has been explained why the speed and travel time predictions are not directly correlated. Therefore, also other common loss functions will be used to train the model, which are MSE, MAE, MAPE and sMAPE. One or multiple loss functions may optimize the speed predictions of the neural network, such that the sMdAPE$_{TT}$ will be improved with respect to den Heijer's best models.

$$\text{sMdAPE}_{LF} = \text{median}\left(200\% \cdot \left|\frac{A_t - P_t}{A_t + P_t}\right|\right) \qquad t \in 1, ..., n \tag{5.3.4}$$

- **MSE$_{LF}$:** The mean squared error (MSE) is a very common loss function and shown in Equation 5.3.5. The MSE$_{LF}$ is easy to implement and mostly used as default loss function. However, the MSE$_{LF}$ is a non-relative error function, which means that the model fits better to larger speeds as explained in subsection 2.3.2. Therefore, besides the speed, also the logspeed will be used as dependent variable which puts the speeds in a relative space and has been explained in section 5.2. Another property of the MSE$_{LF}$ is its sensitivity to outliers. Whether this is beneficial for the speed predictions is unknown and hard to estimate. Therefore, the MAE$_{LF}$ will also be used, which is less sensitive to outliers.

$$\text{MSE}_{LF} = \frac{1}{n} \cdot \sum_{t=1}^{n} (A_t - P_t)^2 \tag{5.3.5}$$

- **MAE$_{LF}$:** The mean absolute error (MAE) takes the absolute value instead of the square like the MSE$_{LF}$ and is shown in Equation 5.3.6. This means that the loss function gets less sensitive to outliers than the MSE$_{LF}$, which results in a more robust loss function. The MAE$_{LF}$ is like the MSE$_{LF}$ a non-relative error function. Therefore, also the logspeed will be used as dependent variable which puts the speeds in a relative space. Another solution is to use the relative loss function MAPE$_{LF}$.

$$\text{MAE}_{LF} = \frac{1}{n} \cdot \sum_{t=1}^{n} |A_t - P_t| \tag{5.3.6}$$

- **MAPE$_{LF}$:** The mean absolute percentage error (MAPE) is also a very common loss function due to its simplicity and easy interpretation, and is shown in Equation 5.3.7. It is easy to interpret, because it indicates how much percent the average prediction deviates from the actual value. This MAPE$_{LF}$ is a relative loss function, which means that the model fits well to large and small speeds. However, the disadvantage of the MAPE$_{LF}$ is that when speeds are equal to zero, the MAPE$_{LF}$ goes to infinite. Therefore, this loss function will only be used for the dependent variable pace (1/speed). Consequently, the sMAPE$_{LF}$ will also be used, which is able to handle speeds that are equal to 0 km/h.

$$\text{MAPE}_{LF} = \frac{100\%}{n} \cdot \sum_{t=1}^{n} \left|\frac{A_t - P_t}{A_t}\right| \tag{5.3.7}$$

- **sMAPE$_{LF}$:** The symmetric mean absolute percentage error (sMAPE) is a relative error function and is shown in Equation 5.3.8. The sMAPE$_{LF}$ is a symmetric loss function, which means that a prediction with a factor of 2 higher or lower, compared to the actual value, will have the same error. Whether this

is beneficial for the speed predictions is unknown and hard to estimate.

$$\text{sMAPE}_{\text{LF}} = \frac{200\%}{n} \cdot \sum_{t=1}^{n} \left| \frac{A_t - P_t}{A_t + P_t} \right| \tag{5.3.8}$$

5.3.4. NUMBER OF NEURONS AND HIDDEN LAYERS

Both the number of neurons and hidden layers are important for the neural network architecture and performance. Too few neurons and hidden layers result in under-fitting. A too large number of neurons and hidden layers result in over-fitting and a too large training time. Unfortunately, there is no exact formula that can be used to calculate the best number of neurons and hidden layers for the neural network. This differs from one data set to the other. Generally, to determine the number of neurons, the following rule-of-thumb methods are used[94]:

1. The number of hidden layer neurons are 2/3 (or 70% to 90%) of the size of the input layer. If this is insufficient, then the number of output layer neurons can be added later[95].
2. The number of hidden layer neurons should be less than twice of the number of neurons in the input layer[96].
3. The size of the hidden layer neurons is between the input layer size and the output layer size[97].

Usually, the same number of nodes are added to each hidden layer and will also be applied to the model in this research. Because there is no exact formula to determine the number of nodes, the three rule-of-thumb methods described above are used to determine the search space. This results in a search space for hidden neurons between one and twice the number of neurons in the input layer. For 21 independent variables, this result in a search space between 1 and 42 hidden neurons.

Like for hidden neurons, there is no exact rule to determine the number of hidden layers. Hagan et al. [98] claims that it is unusual to use more than two hidden layers. However, for difficult problems involving time-series and computer vision, more than two layers may be required. Additionally, Heaton [99] states that two hidden layers are rarely encountered. For the neural network it is decided to use two hidden layers, since from two hidden layers, functions with any kind of shape can be represented[99].

5.3.5. OPTIMIZER

To accelerate the optimization process of the neural network an additional algorithm can be applied to the MBGD algorithm. The most common optimizers are: momentum, Nesterov accelerated gradient, Adagrad, AdaDelta, RMSprop, Adam, AdaMax, Nadam and AMSGrad. Ruder [100] recommends Adam as optimizers to train a neural network and is a combination of the advantages of AdaGrad and RMSProp. In Figure 5.8, a comparison between different optimizers is shown for a multi-layer neural network[101]. As can be seen, Adam obtains the lowest training cost with less iterations than the other optimizers. Other advantages of Adam are[101]:

- Straightforward to implement
- Computationally efficient
- Little memory requirements
- Well suited for problems that are large in terms of data and/or parameters
- Hyper-parameters have intuitive interpretation and typically require little tuning

Due to the advantages and in general better performance than other optimizers, Adam is chosen as optimizer for the neural network. The values of the hyperparameters are set to good default settings which are[101]: α = 0.001, β_1 = 0.9, β_2 = 0.999 and $\epsilon = 10^{-8}$.

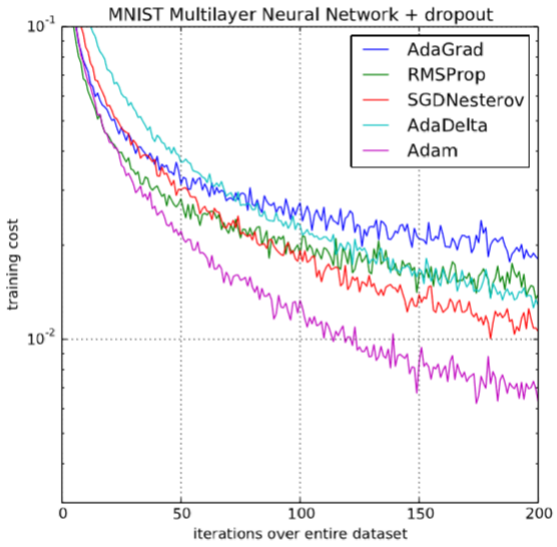

Figure 5.8: Training of multi-layer neural networks on MNIST images with AdaGrad, RMSProp, SGD Nesterov, AdaDelta and Adam as optimizers[101]

5.3.6. LEARNING RATE

The learning rate is known as one of the hyperparameters to tune. A small learning rate causes the network to optimize slowly, but carefully. A large learning rate causes the network to optimize quickly, but overshooting might also occur as shown in Figure 5.6. However, Adam is an adaptive learning rate optimization algorithm which means that the algorithm adjust the learning rate for each individual parameter during the training process. This is beneficial, since fast convergence can be made at the beginning with a large learning rate. While slow convergence can be made later with a small learning rate leading to accurate results. A good default setting for the learning rate of Adam is 0.001[101]. This will also be used in this research study.

5.3.7. ACTIVATION FUNCTION

Activation functions are part of the neurons in the hidden and output layers. The aim of an activation function is to decide in which extent the sum of the inputs and bias should be fed forward to the neurons in the next layer or output. In general 3 properties of activation functions are desired:

1. **Non-linearity:** The non-linearity property of an activation function is key to be able to solve non-linear problems.
2. **Continuously differentiable:** The first order derivative of the activation function has to be continuous to enable gradient-based optimization.
3. **Monotonic:** Monotonic means that the activation function accelerates the convergence of the NN to a precise model. The function is then either entirely non-increasing or non-decreasing.

A simple example of an activation function is a step function which has an output of either 0 or 1. If the output is '1', then the output of the neuron is activated and is sent to the next neurons. If the output is '0', then the output of the neuron is deactivated and not send to the neurons in the next layer. However, a binary activation function would limit the outcome to activated and deactivated. Intermediate results such as 0.3 (30%) or 0.8 (80%) cannot be applied which lowers the accuracy of the model. Popular activation functions along with their graphs, which are not binary, are shown in Figure 5.9. Below a short discussion of each function is provided[102]:

- **Sigmoid:** The Sigmoid function ranges from 0 to 1, is not zero centered and behaves exponentially. The Sigmoid function suffers from the vanishing gradient problem. This means that the end of the curve is almost horizontal and that the gradient is very low. This slows down or stops the gradient optimizer to learn further. The Sigmoid function is mostly used for the output layer where outputs between 0 and 1 are preferred.

- **Tanh:** Tanh or also called the hyperbolic tangent activation function has a range between -1 and 1 and is zero centered. Like the Sigmoid function, the end of the Tanh function is almost horizontal which might trouble the optimization of the NN.
- **ReLU:** The rectified linear unit activation function (ReLU) is the most popular activation function. This is because it is simple and requires relatively low computational effort. During the gradient optimizing process, the function does not saturate, but has fast convergence. However, if w>0 (weight) and x<0 (input), then ReLU(w*x) = 0 which vanishes the gradient and is called a dead ReLU problem.
- **Leaky ReLU:** The Leaky ReLU behaves in the same way as the ReLU function, but has improved performance, since it has not the dead ReLU problem.
- **Maxout:** Maxout has a linear property, never saturates or die and needs relatively more computational effort.
- **ELU:** ELU is an exponential version of the ReLU, but has no dead ReLU solution. Also, it requires more computational effort due to its exponential behavior.

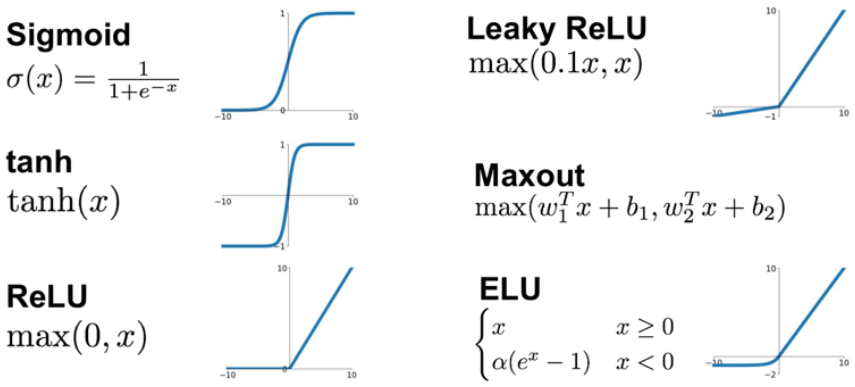

Figure 5.9: Examples of different activation functions and their graphs[87].

HIDDEN LAYER ACTIVATION FUNCTION

In Table 5.2, the available activation functions in *Python* can be found, including the range. The most common activation function to use in the hidden layer is the rectified linear unit (ReLU)[103]. However, the ReLU activation function has as disadvantage that if a neuron gets negative, then it can hardly recover. This is because the slope of the ReLU in the negative range is 0. It is possible to end up with many neurons in the network that are actually useless. To solve this problem, the Leaky ReLU, parametric rectified linear unit (PReLU) and exponential linear unit (ELU) were developed to fix this problem. For the neural network in this research, the exponential linear unit (ELU) function is chosen instead of the most common ReLU function. The ELU has a clear saturation plateau in the negative regime, which leads to a more robust and stable training process compared to Leaky ReLu and PReLU[103]. Other advantages of the ELU are fast convergence and accurate results, compared to other activation functions, which is based on experiments with different data sets[103].

Table 5.2: Available activation functions and range available in Keras.

Activation Function	Range
Linear	(-∞, ∞)
ReLU	(0, ∞)
ELU	(-α, ∞)
SELU	(-∞, ∞)
Softplus	(0, ∞)
Softmax	(0, 1)
Softsign	(-1, 1)
TanH	(-1, 1)
Sigmoid	(0, 1)
Hard Sigmoid	(0, 1)

OUTPUT LAYER ACTIVATION FUNCTION

The output layer activation function provides a final transformation of the features into output $\hat{y}_i$. For this research, the output of the activation function is the speed, which is in the range of (0,90]. According to Table 5.2, this leaves the Linear, ReLU, ELU, SELU and Softplus as potential options for the output activation function. This is because these functions can produce an output for the range (0,90]. In literature, the linear activation function is commonly used to predict a continuous output, since it is unbounded. For example, Wisitpongphan et al. [22] developed a feed-forward neural network, with a linear output activation function, to predict the travel speed on a highway. Later, the predicted travel speed was converted to the travel time, by dividing the highway length by the calculated travel speed. Another example is that Bilgili et al. [104] developed an feed-forward neural network, with a linear output activation function, to predict the wind speed. Both researches showed good results by using the linear activation function in the output layer. Therefore, in this research also the linear activation function will be used in the output layer which can be written as σ(x) = x.

5.3.8. MAXIMUM NUMBER OF EPOCHS

The maximum number of training epochs indicate after how many iterations the training process stops. This is not an important hyperparameter, since the Early Stopping method is used to stop the training process earlier, when the error starts to degrade on the validation set. Therefore, the maximum number of epochs is set to 200, which should be enough to let the Early Stopping method stop the training process, before reaching the maximum number of epochs. If 200 seems not to be sufficient, then the maximum number of epochs will be increased.

5.4. MATHEMATICAL FORMULATION NEURAL NETWORK

After the model design choices have been made, the mathematical formulation of the neural network can be written. First all parameters are defined and are shown below:

i	layer number, $i \in 1,2,...,I$
j	neuron number in layer i, $j \in 1,2,...,J_i$
k	neuron number in layer i-1, $k \in 1,2,...,J_{i-1}$
J_i	total number of neurons in layer i
I	total number of layers
a_j^i	output of neuron j in layer i
σ^i	activation function of all neurons in layer i
b_j^i	bias of neuron j in layer i
w_{jk}^i	weight from neuron k in layer i-1 to neuron j in layer i
n	number of observations
t	single observation out of all observations
y_t	actual value of observation t
x_j	input j (independent variable) in layer i=1

In section 5.3, it was discussed that five different loss functions will be used to train the neural network model. This means that the model has five different objectives based on the loss functions MSE_{LF}, MAPE_{LF}, MAE_{LF}, sMAPE_{LF} and $\text{sMdAPE}_{\text{LF}}$. These objectives are shown in Equation 5.4.1, Equation 5.4.2, Equation 5.4.3, Equation 5.4.4 and Equation 5.4.5. To minimize each objective, the Adam optimizer algorithm is used. In Equation 5.4.6, the mathematical formulation of the output of the neurons in the input layer is shown. In Equation 5.4.7, the mathematical formulation of the output of the other neurons is shown.

$$\text{Minimize} \quad C(j=1,i=I) = \frac{1}{n}\sum_{t=1}^{n}(y_t - (a_j^i)_t)^2 \qquad (\text{MSE}_{\text{LF}}) \qquad (5.4.1)$$

$$\text{Minimize} \quad C(j=1,i=I) = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{y_t - (a_j^i)_t}{y_t}\right| \qquad (\text{MAPE}_{\text{LF}}) \qquad (5.4.2)$$

$$\text{Minimize} \quad C(j=1,i=I) = \frac{1}{n}\sum_{t=1}^{n}|(y_t - (a_j^i)_t| \qquad (\text{MAE}_{\text{LF}}) \qquad (5.4.3)$$

$$\text{Minimize} \quad C(j=1,i=I) = \frac{200\%}{n}\sum_{t=1}^{n}\left|\frac{y_t - (a_j^i)_t}{y_t + (a_j^i)_t}\right| \qquad (\text{sMAPE}_{\text{LF}}) \qquad (5.4.4)$$

$$\text{Minimize} \quad C(j=1,i=I) = \text{median}\left(200\%\sum_{t=1}^{n}\left|\frac{y_t - (a_j^i)_t}{y_t + (a_j^i)_t}\right|\right) \qquad (\text{sMdAPE}_{\text{LF}}) \qquad (5.4.5)$$

Where:

$$a_j^i = x_j \qquad i \in 1, \forall j \in \{1,2,...,J_i\} \qquad (5.4.6)$$

$$a_j^i = \sigma^i\left(\sum_k \left(w_{jk}^i \cdot a_k^{i-1}\right) + b_j^i\right) \qquad \forall i \in \{2,3,...,I\}, \forall j \in \{1,2,...,J_i\} \qquad (5.4.7)$$

5.5. IMPLEMENTATION

5.5.1. SOFTWARE

For this research, ***Python version 3.6*** is used as programming language and is an open-source software. The reasons to take *Python* as programming language are the following:

1. *Python* has many open source frameworks, libraries and tools which can be used. This saves a lot of time and money.
2. ORTEC is familiar with python which makes it easy for them to use the files after the research.
3. The author is most familiar with *Python*, which favors the speed of programming and the research.

To build the neural network, many free packages are available which allows the user to build neural networks quickly and efficiently. It is decided to use ***Keras*** which is one of the leading APIs and is written in *Python*. *Keras* runs on top of ***Tensorflow*** to allow programming in *Python* and to make the usage of *Tensorflow* user friendly and modular. *Tensorflow* is an open source library created by Google for numerical computation and large-scale machine learning. Feature scaling and cross-validation will be done by ***scikit-learn*** which is a free machine learning library for *Python*.

The software that is used to calculate the shortest path and travel time from point A to B is called ***HNR*** and is developed by ORTEC. This software is based on the Highway Node Routing algorithm developed by Schultes and Sanders[2].

5.5.2. HARDWARE

The preprocessing, learning, prediction and evaluation steps are all performed on one computer with the following specifications:

Intel Core i7-8650U CPU 1.90GHz
16,0 GB RAM
Windows 10 OS

5.6. HYPERPARAMETER TUNING

All hyperparameters of the neural network, along with their values, are shown in Table 5.3. These values are determined in section 5.3 and based on literature references. As can be seen, most hyperparameters are already fixed, while the mini-batch size and number of neurons still have a range of values from which the best value needs to be chosen.

Table 5.3: Overview neural network hyperparameters and values based on literature references.

Hyperparameter	**Value**
Learning Rate	0.001
Loss Function	[MSE_{LF}, $MAPE_{LF}$, MAE_{LF}, $sMAPE_{LF}$, $sMdAPE_{LF}$]
Mini-Batch Size	[32, 64, 128, 256, 512]
Early Stopping Epochs	10
Number of Hidden Layers	2
Number of Neurons	[1 - 42]
Optimizer	Adam
Hidden Layer Activation Function	ELU
Output Layer Activation Function	Linear
Maximum Epochs	200

Because of the different loss functions, dependent variables and data types, different types of neural network models can be developed. In Table 5.4, an overview of the different models can be found, which are 21 models in total, indicated by a cross. As can be seen, the $MAPE_{LF}$ will not be tested for the speed and logspeed, since speeds equal to 0 km/h or log(1 km/h) causes the $MAPE_{LF}$ to rise to infinity. For each model, the best set of hyperparameters needs to be found. This will be done through a hyperparameter tuning process. To tune these hyperparameters, cross-validation will be applied. This is used to find the set of hyperparameters that generalizes the model such, that the best model performance is obtained.

Table 5.4: Overview of different neural network models that will be developed. These models differ in loss function, type of training data (point- and trip-based) and dependent variable (speed, pace and time).

Dependent Variable	Data Type	MSE_{LF}	$MAPE_{LF}$	MAE_{LF}	$sMAPE_{LF}$	$sMdAPE_{LF}$
Speed	points	x		x	x	x
Logspeed	points	x		x	x	x
Speed	trips	x		x	x	x
Logspeed	trips	x		x	x	x
Pace	trips	x	x	x	x	x

5.6.1. CROSS-VALIDATION METHOD

Cross-validation is a technique that can be used to assess the effectiveness of the built ML model on an unseen or independent data set. This is needed, because the model can have an excellent performance on the training set, but a bad performance on the test set. This means that the model is not well generalized. By using cross-validation, the correctness of the found patterns, bias and variance are found[105]. Cross-validation is also an useful method for hyperparameter tuning to choose the best combination of hyperparameters for the model. The hyperparameter tuning process, using cross-validation, is explained in the following subsection. Cross-validation can be applied to any ML model and is in general highly recommended. Cross-validation can be categorized into two classes:

- **Non-exhaustive methods:** These cross-validation methods do not split the data in all possible combinations for training and validation data sets. Non-exhaustive methods are: holdout, k-fold cross-validation and stratified k-fold cross validation.
- **Exhaustive methods:** These cross-validation methods are more time consuming and use every combination how the training and validation data sets can be split. An exhaustive methods is leave-p-out cross-validation.

A description of the four most common cross-validation methods are discussed below[105]:

- **Holdout method:** The holdout method is the simplest cross-validation method and reserves a part of the data set, validation data set, to assess the performance of the trained model. However, this method is vulnerable for high variances, since a different validation set with different data points may give different results. Therefore the holdout method is not preferred.
- **Leave-P-out cross-validation:** This method uses p data points from the training data for the validation set. When there are n data points, then the model is trained on n-p data points. Leave-p-out cross-validation is an exhaustive method, since it uses every possible combination with p data points, from the training set, to validate the model. For p=1, also called leave-one-out cross-validation, the method is least exhaustive. This is because the number of combinations is equal to n, which is the size of the training data set.
- **K-fold cross-validation:** The K-fold method uses instead of one validation set, k validation sets. The entire training data set is split into k subsets and at each iteration a different subset is used to validate the model. This is shown in Figure 5.10. This means that each training data point is used once for validation and k-1 times for training the model. In this way, the bias of the model is significantly reduced, since most data is used for fitting. Also, the variance is significantly reduced, since different validation sets are used. There is no general rule for the number of folds, but from experience, found in literature, k=5 or k=10 is generally preferred.

Figure 5.10: K-fold cross-validation with k=10[106].

- **Stratified k-fold cross-validation:** Stratified k-fold cross-validation is an advanced version of k-fold cross-validation. The difference is that this method equally divides the data into folds, based on the value of the dependent variable. This results in the same mean value of the dependent variable in each fold. This method is used for classification problems when the dependent variable is not equally distributed.

For this research, the k-fold cross-validation method will be used. This is method is chosen, since it has a reduced bias and variance, compared to the holdout and leave-p-out method. Also, the method is non-exhaustive, which means that it is not very time consuming such as the leave-p-out method. The k-fold is chosen over the stratified k-fold method, since stratified k-fold can only be applied to classification problems. In this research, the speed is predicted, which is a regression problem.

Number of Folds

The number of folds k that needs to be chosen for k-fold cross-validation is based on a bias-variance trade-off. A high bias oversimplifies the model, while a high variance generalizes the model too little. Values for $k = 5$ or $k = 10$ provide an error on the test fold that neither suffer from high bias nor variance [107]. A 5-fold cross-validation is chosen, since this requires about twice as less computational time than 10-fold cross-validation. This means that the training set will be split into 5 random sub data sets. During each iteration, one sub data set is used as validation set and the other four as training set.

5.6.2. HYPERPARAMETER TUNING PROCESS

Now the cross-validation method is chosen, the hyperparameters can be tuned. It is important to set the hyperparameters properly to obtain the best model performance. Hyperparameters are parameters that cannot be learned by the model itself and have to be set by the user. Cross-validation can be used to find out which combination of hyperparameters performs the best, where an example is shown in Figure 5.11. In this figure, 10-fold cross-validation is applied to n different hyperparameter combinations. The hyperparameter combination with the highest cross-validation accuracy is the best and is finally used to train the model with the full training set.

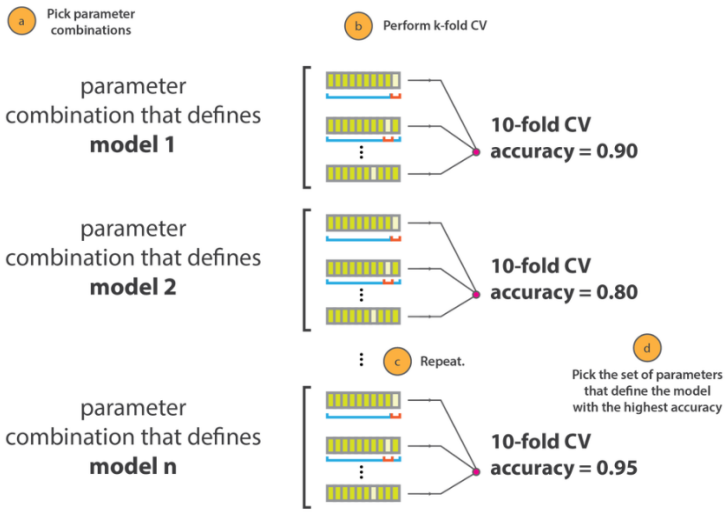

Figure 5.11: Example of the hyperparameter tuning process using 10-fold cross-validation[108].

In total, there are 42 different neural network models from which the hyperparameters need to be tuned. 21 models for the new data and 21 models for the old data (Table 5.4). There are different hyperparameter tuning methods that can be used to choose different hyperparameter combination at the first step in in Figure 5.11. The most common hyperparameter tuning methods are grid search, random search and Bayesian optimization and are discussed in Appendix E. Using either grid search, random search or Bayesian optimization for each model would take a significant amount of time and is not realistic within the time window of this research. Because it is expected that each model has more or less the same best hyperparameter combination, due to the same format of the dataset, the following approach will be used:

1. To save a tremendous amount of time on hyperparameter tuning, first the best combination of hyperparameters for one model will be found. This is done, instead of doing the hyperparameter tuning process for all 21 models for the new and old data.
2. After the best combination of hyperparameters for one model is found, a sensitivity analysis will done for all 21 models. This is done to find out whether the hyperparameter combination in the previous step is the best for all models. If this is not the case, hyperparameter tuning will be repeated for the models where the hyperparameter combination in the previous step does not provide the best result.

In section 5.3, the model design choices were made and are mostly based on literature references. In general, there is not one optimal model design that can be applied to the neural network for any problem. The best hyperparameter setting depends from one problem to the other. Instead of using a time consuming hyperparameter tuning algorithm such as grid and random search, the best hyperparameter combination will be found by analyzing the hyperparameters individually. The most important hyperparameters that have generally the largest influence on the prediction accuracy ([109], [110],[111]) and will be analyzed are:

1. Learning Rate
2. Number of Neurons
3. Number of Layers
4. Mini-Batch Size

Firstly, the hyperparameters of the Points-NN-Speed-MSE model will be tuned. This model is trained on point-based data with dependent variable speed and loss function MSE_{LF}. After the best hyperparameters of this model are found, a sensitivity analysis will be done for all models and is discussed in subsection 5.6.3.

1. Learning Rate

The learning rate is known as the most important hyperparameter. To make sure that a learning rate of 0.001 produces the best results, different learning rates between 0.0001 and 0.1 on a logarithmic scale are

researched. This approach is generally executed in literature such as [112] and [113]. This results in learning rates: 0.0001, 0.001, 0.01 and 0.1. Each learning rate is researched using 5-fold cross-validation. The other hyperparameter values are shown in Table 5.3. Where the loss function is MSE_{LF}, mini-batch size 128 and number of hidden neurons 21, which is the average of 1 and 42.

The results are shown in Table 5.5 and Figure 5.12, which show that a learning rate of 0.001 produces the best model accuracy. A learning rate of 0.01 and 0.1 are too high causing divergence of the optimization process. A learning rate of 0.0001 is too small causing very slow convergence. This may result in a optimization process that got stuck at a suboptimal solution. The lower training time of the learning rates 0.0001, 0.01 and 0.1, is due to the early stopping regularization method. The model stops training when the model does not improve after 10 epochs. These experiments show that a learning rate of 0.001 result in the best model accuracy. This learning rate is also recommended by the developer of the Adam optimizer. Therefore, a learning rate of 0.001 will be used for all other neural network models and further experiments.

Table 5.5: Model accuracy and 5-fold CV training time versus learning rate for the model Points-NN-Speed-MSE. The learning rate ranges between 0.0001 and 0.1 on a logarithmic scale, number of neurons = 21, number of hidden layers = 2 and the mini-batch size = 128. The best result is underlined.

Learning Rate	**Accuracy (MSE_{LF})**	**Time [seconds]**
0.0001	237.3	3383
0.001	<u>230.8</u>	5038
0.01	235.0	2140
0.1	249.4	653

Figure 5.12: Model accuracy (purple circle) and 5-fold CV training time (blue square) versus learning rate for the model Points-NN-Speed-MSE. The learning rate ranges between 0.0001 and 0.1 on a logarithmic scale, number of neurons = 21, number of hidden layers = 2 and the mini-batch size = 128.

2. Number of hidden neurons

To make sure that the best number of neurons is found, the search space will be between 1 and 512 neurons. Each model is trained using 5-fold cross-validation with the hyperparameters shown in Table 5.3. The loss function is MSE_{LF}, number of hidden layers is 2 and the mini-batch size is 128. The results are shown in Table 5.6 and Figure 5.13 and show that the model accuracy improves with the number of neurons. However, from 128 neurons the model accuracy is more or less constant. Also, it can be concluded that the training time

for 5-fold CV increases linearly from 64 neurons to 512 neurons. This analysis shows that the best number of neurons is 128, which is far outside the range (1-42) recommended in literature.

Table 5.6: Model accuracy and 5-fold CV training time versus number of hidden neurons for the model Points-NN-Speed-MSE. The number of hidden neurons ranges between 1 and 512, lr = 0.001, number of hidden layers = 2 and mini-batch size = 128. The best result is underlined.

Hidden Neurons	**Accuracy (MSE_{LF})**	**Time [seconds]**
1	271.4	1090
2	247.2	2959
4	238.1	2073
8	235.9	2989
16	232.9	4198
32	228.0	3340
64	224.3	3438
128	<u>222.4</u>	5243
256	222.6	8343
512	223.0	14461

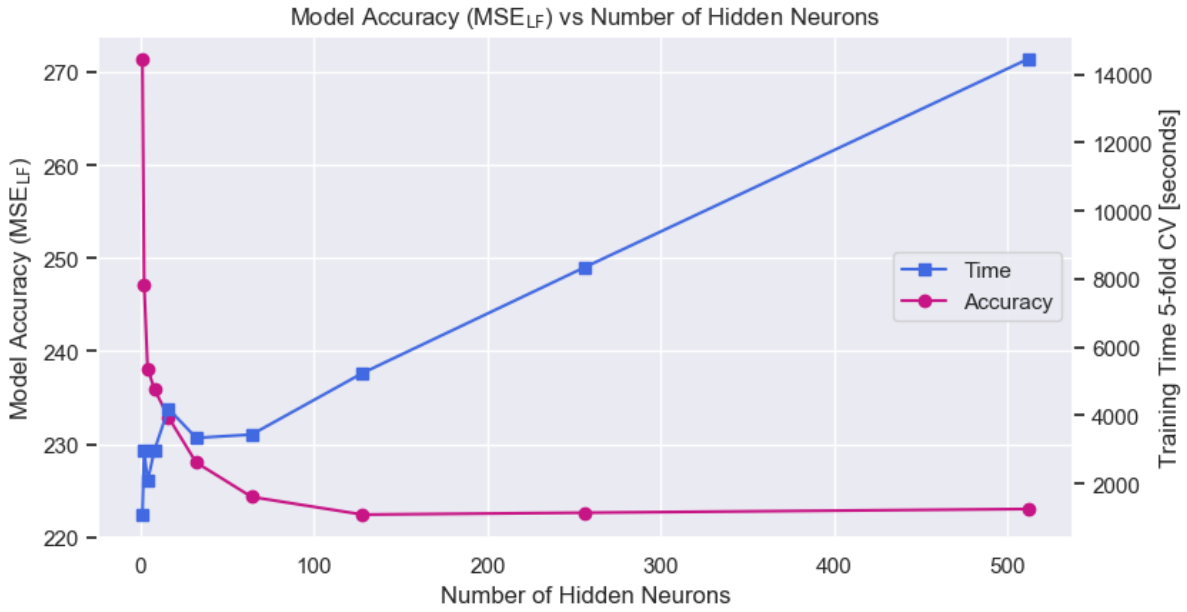

Figure 5.13: Model accuracy (purple circle) and 5-fold CV training time (blue square) versus number of hidden neurons for the model Points-NN-Speed-MSE. The number of hidden neurons ranges between 1 and 512, lr = 0.001, number of hidden layers = 2 and mini-batch size = 128.

3. Hidden Layers

Based on literature references, the number of hidden layers that was chosen for the neural network is 2. However, the number of hidden layers is known as an important hyperparameter and to make sure that 2 hidden layers are rightly chosen, the neural network will also be analyzed for more hidden layers. The number of hidden layers that are analyzed ranges between 1 and 8. Each model is trained using 5-fold cross-validation with the hyperparameters shown in Table 5.3. The loss function is MSE_{LF}, mini-batch size 128 and number of neurons 128, which has been determined in the previous step. The results are shown in Table 5.7 and Figure 5.14, and show that the prediction accuracy can be improved substantially, by increasing the number of layers. The best model accuracy in MSE_{LF} is obtained with 5 hidden layers. After this, the model becomes too complex which results in a lower model accuracy.

Table 5.7: Model accuracy and 5-fold CV training time versus number of hidden layers for the model Points-NN-Speed-MSE. Number of hidden layers ranges between 1 to 8, lr = 0.001, number of neurons = 128 and mini-batch size = 128. The best result is underlined.

Hidden Layers	**Accuracy (MSE_{LF})**	**Time [seconds]**
1	231.2	2414
2	223.3	3653
3	218.9	4841
4	217.9	4794
5	<u>216.6</u>	6612
6	221.9	3510
8	225.8	4179

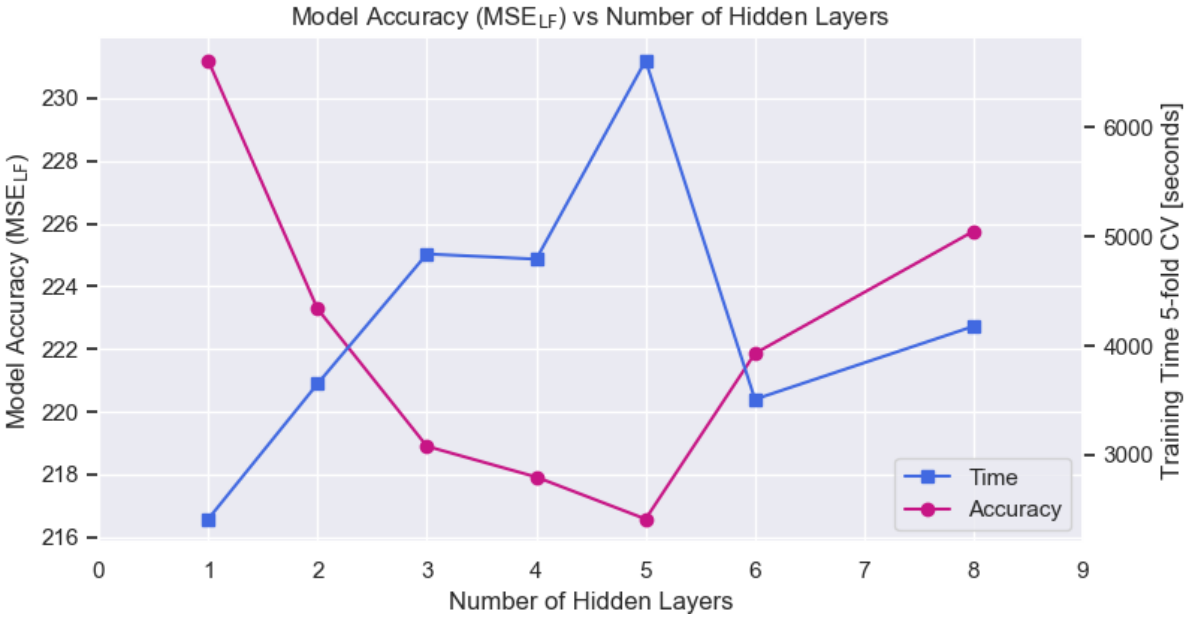

Figure 5.14: Model accuracy (purple circle) and 5-fold CV training time (blue square) versus number of hidden layers for the model Points-NN-Speed-MSE. Number of hidden layers ranges between 1 to 8, lr = 0.001, number of neurons = 128 and mini-batch size = 128.

4. Mini-Batch Size

The recommended mini-batch size in literature is 32, 64, 128, 256 or 512. These numbers are all to the power of 2, which fit well to the GPU or CPU memory requirements. To make sure that other mini-batch sizes are not overlooked, a mini-batch size of 16 and 1024 are also added to the hyperparameter tuning process. Each model is trained using 5-fold cross-validation with the hyperparameters shown in Table 5.3. The loss function is MSE_{LF}, the number of neurons 128 and the number of layers 5, which have been determined in the previous hyperparameter tuning steps. The results are shown in Table 5.8 and Figure 5.15. It can be seen that the best mini-batch size is 256. Smaller mini-batch sizes lead to a noisy training process and divergence. Too large mini-batch sizes lead to convergence at a non-optimal minimum. The training time increases exponentially with a smaller mini-batch size.

Table 5.8: Model accuracy and 5-fold CV training time versus mini-batch size and 5-fold CV training time for the model Points-NN-Speed-MSE. Mini-batch size ranges between 16 and 1028, lr = 0.001, number of neurons = 128 and number of hidden layers = 5. The best result is underlined

Mini-Batch Size	**Accuracy (MSE_{LF})**	**Time [seconds]**
16	231.2	12030
32	228.5	9060
64	220.3	6850
128	218.6	3907
256	<u>216.2</u>	2756
512	217.8	2137
1024	223.1	1430

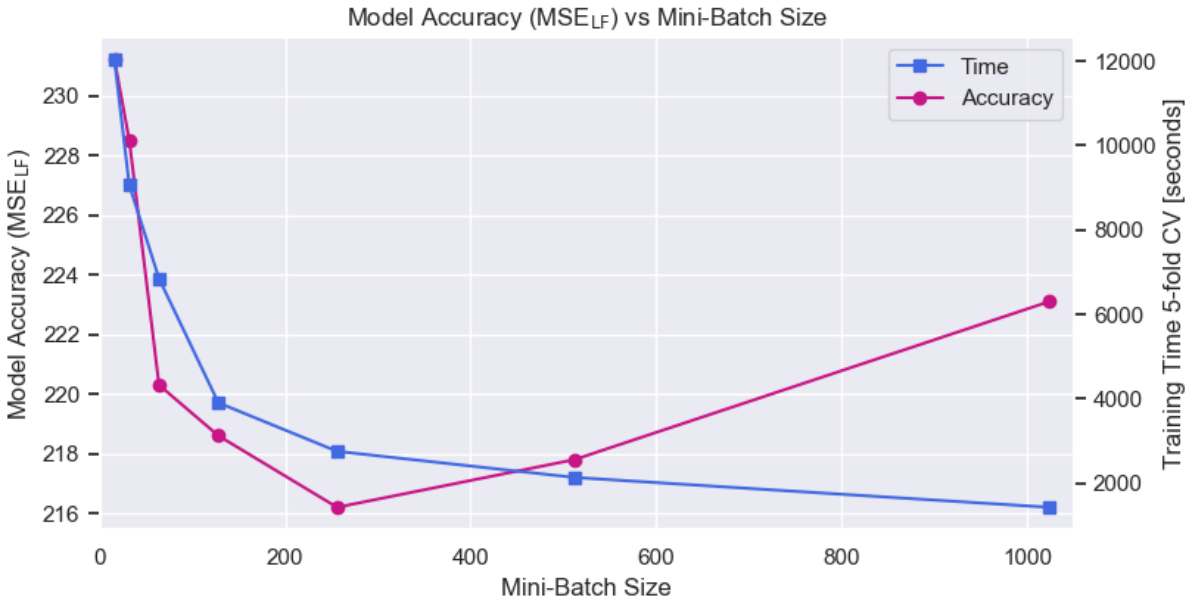

Figure 5.15: Model accuracy (purple circle) and 5-fold CV training time (blue square) versus mini-batch size for the Points-NN-Speed-MSE model. Mini-batch size ranges between 16 and 1028, lr = 0.001, number of neurons = 128 and number of hidden layers = 5.

5.6.3. SENSITIVITY ANALYSIS

To repeat the hyperparameter tuning process of the Points-NN-Speed-MSE model for all other 20 models, would take too much time. Therefore, it is decided to do a sensitivity analysis for the best hyperparameter combination found by the Points-NN-Speed-MSE model. The sensitivity analysis will be done for all models. The combinations of hyperparameters for the sensitivity analysis are shown in Table 5.9. The initial setting is the best combination found by the Points-NN-Speed-MSE model. For the other combinations, a lower or higher value is chosen for the number of hidden neurons, number of hidden layers and mini-batch size. The learning rate is fixed to 0.001, since this value is recommended by the developer of the Adam optimizer and showed the best model accuracy for the experiments shown in Figure 5.12. By fixing the learning rate, additional computational time is saved.

The results of the sensitivity analysis for the new data of all 21 models are shown in Table F.1, in Appendix F. In this table, the best model accuracy of the 7 different hyperparameter combinations, for each model, is underlined. From these results, it can be concluded that all models, except the ones with loss function $sMdAPE_{LF}$ and dependent variable pace, produce the best result with the initial setting. Remarkably, the models that are either trained on the dependent variable pace and/or loss function $sMdAPE_{LF}$, produce the best result for hyperparameter combination 2 (64 neurons and 5 hidden layers). This shows that these models require a less complex model to produce better results. Therefore, the hyperparameter tuning process was repeated for the models with dependent variable pace and loss function $sMdAPE_{LF}$. This is discussed in Appendix F. The results in Appendix F show that the best hyperparameter combination for the models with dependent variable pace is 32 neurons, 4 hidden layers and mini-batch size 256. The best hyperparameter combination

Table 5.9: Seven different hyperparameter combinations to research whether the initial setting produces the best model accuracy.

Combination	Learning Rate	Hidden Neurons	Hidden Layers	Mini-Batch Size
Initial Setting	0.001	128	5	256
1	0.001	256	5	256
2	0.001	64	5	256
3	0.001	128	6	256
4	0.001	128	4	256
5	0.001	128	5	128
6	0.001	128	5	512

for the models with loss function $sMdAPE_{LF}$ is 32 neurons, 7 hidden layers and mini-batch size 256.

Subsequently, a sensitivity analysis was done for the old data set, for all 21 neural network models, and is discussed in Appendix G. These results show that the best number of neurons and hidden layers is the same for the new and old data set. This shows that the smaller old data set, compared to the new data set, does not influence the best number of neurons, number of hidden layers and mini-batch size. In Table 5.10, an overview is shown of the final hyperparameter values that will be used to train the models in chapter 6. These sets of hyperparameter are divided into 3 categories: models without pace and $sMdAPE_{LF}$, models with pace and models with $sMdAPE_{LF}$.

Table 5.10: Overview of final neural network hyperparameters divided into three model categories: all models without dependent variable pace and $sMdAPE_{LF}$, models with dependent variable pace and models with $sMdAPE_{LF}$. This overview of hyperparameters applies to the new and old data set.

Hyperparameter	Models without pace and $sMdAPE_{LF}$	Models with pace	Models with $sMdAPE_{LF}$
Learning Rate	0.001	0.001	0.001
Mini-Batch Size	256	256	256
Early Stopping Epochs	10	10	10
Number of Hidden Layers	5	4	7
Number of Neurons	128	32	32
Optimizer	Adam	Adam	Adam
Hidden Layer Activation Function	ELU	ELU	ELU
Output Layer Activation Function	Linear	Linear	Linear
Maximum Epochs	200	200	200

5.7. CONCLUSION

The conclusions that could be made in this chapter are the following:

- The $sMdAPE_{TT}$ will be used to compare the travel time prediction accuracy of different speed prediction models. The $sMdAPE_{TT}$ has several properties that make this error function most suitable to calculate the travel time prediction accuracy. Examples are the insensitivity to outliers, due to the median, and equal penalization to positive and negative errors, due to its symmetry. In addition, den Heijer also used the $sMdAPE_{TT}$ to compare the travel time prediction accuracy of different models. This makes the comparison between den Heijer's best models and the new developed neural network models in this research easier.
- In total, 21 independent variables will be used as input of the neural network. These independent variables are all based on road engineering factors, which are available from the map data. In total, 3 different dependent variables will be used as output of the neural network. The dependent variable of the neural network is either speed, logspeed or pace. These dependent variables were also used by den Heijer, where each dependent variable has its own benefits.
- In total, 21 different neural network models will be developed for old and new data. This is because of the different dependent variables (speed, logspeed and pace), data types (point- and trip-based) and loss functions used. Five different loss functions will be used, where each loss function has its own

advantages. They will all be used, since it is unclear and hard to estimate, which loss function will predict the speeds such that the travel time prediction accuracy is improved.

- The hyperparameters that were tuned are the learning rate, number of hidden layers, number of neurons and mini-batch size. It was found that the best learning rate is 0.001 and is used for all models. The best number of neurons and hidden layers differs among the models. They depend on whether the dependent variable pace or loss function $sMdAPE_{LF}$ is used. Furthermore, it was found that a mini-batch size of 256, had the best results for all models. In Table 5.10, an overview of the best set of hyperparameters, for the new and old data set, can be found. These hyperparameters will be used to train the models in chapter 6.

6

EXPERIMENTS AND RESULTS

In this chapter, first the experiments that will be run will be discussed. Secondly, the speed prediction accuracy of the developed neural network models will be shown and discussed. After this, the travel time prediction accuracy of all models will be presented and discussed. Subsequently, the reason why the old and new data have different best speed prediction models will be analyzed. Lastly, the influence of the temporal factor will be researched by splitting the data set into a rush and non-rush hour data set.

6.1. EXPERIMENTS

The experiments will be run with new data, which is collected and preprocessed during this research, and the old data, from den Heijer[1]. Both data sets are used due to the following reasons:

- **New data:** The new data will be used, due to a higher data frequency than the old data. This data was collected and preprocessed during this research. The data has a frequency of 2 minutes and is obtained from trucks in the Netherlands. It is expected that the quality of the trip-based data will be improved with a higher GPS data frequency, since it is more certain how trucks have driven between two GPS points. As a consequence, it is expected that the speed predictions of the neural network will improve. This may result in better travel time predictions such that the benchmarks are outperformed.
- **Old data:** The old data, from den Heijer, has already been collected and preprocessed. Therefore, this data is ready to be used. The data has a frequency of 5 minutes and is obtained from trucks in the Benelux. By using this data, more insights can be obtained about the developed speed prediction models. Questions that may be answered are: Is the best speed prediction model for the new data the same as for the old data?, Can a neural network, trained on trip-based data, already outperform the benchmarks with a GPS data frequency of 5 minutes?

In Table 6.1, an overview of the most important properties of each data set is shown. The main difference between the two data sets is that the frequency of the new data is 2 minutes and for the old data 5 minutes. Other differences are:

- The minimum evaluated travel time for the new data set is 2 minutes, while for the old data set 10 minutes.
- The old point- and trip-based data set are smaller compared to the new point- and trip-based data set.
- The GPS points of the new data set are recorded in the Netherlands, while the GPS points of the old data set were recorded throughout the Benelux.

In Table 6.2, an overview is shown of the different neural network models that will be trained and used to predict the speeds in the map to improve the travel time prediction accuracy. These models are indicated by an 'x'. For convenience, an abbreviation will be used for these model in the format 'type of training data'-'prediction method'-'model output'-'loss function'. For example, Trips-NN-Pace-MAE means that it is a neural network, trained on trip-based data with output pace and loss function MAE_{LF}. The neural network models will be trained on the old and new data set, which means that there are in total 21*2=42 training processes.

Table 6.1: Summarized description of old and new data used for training and predicting speeds and ultimately travel time predictions.

Data Properties	New Data	Old Data
Average Recorded Frequency	2 min	5 min
Minimum Evaluated Travel Time	2 min	10 min
Training Points	1,083,237	410,137
Test Points	362,141	139,687
Training Trips	785,920	165,479
Test Trips	263.275	54,349
Test Travels	45,852	10,316
Country	Netherlands	Benelux
Vehicle Type	Truck	Truck

Due to the different dependent variables (speed, logspeed and pace), types of training data (point- and trip-based), loss functions (MSE_{LF}, $MAPE_{LF}$, MAE_{LF}, $sMAPE_{LF}$ and $sMdAPE_{LF}$) and data sets (old and new), quite a few experiments have to be run.

In den Heijer's research[1], point-based data performed better than trip-based data. However, by using a neural network model and a higher data frequency (2 min. vs 5 min.), a higher travel time prediction accuracy may be obtained using trip-based data. Therefore, both point- and trip-based data will be used in this research.

Table 6.2: Overview of all different neural network models that will be researched to find the best travel time predictor. The neural network models differ in dependent variable, type of data and loss function. In total, there are 21 different neural network models.

Dependent Variable	Data Type	MSE_{LF}	$MAPE_{LF}$	MAE_{LF}	$sMAPE_{LF}$	$sMdAPE_{LF}$
Speed	points	x		x	x	x
Logspeed	points	x		x	x	x
Speed	trips	x		x	x	x
Logspeed	trips	x		x	x	x
Pace	trips	x	x	x	x	x

The experiments that will be run for the models in Table 6.2, can be divided into two steps: **Training & Evaluation** and **Prediction & Evaluation**. These steps are shown in Figure 6.1 and are the last steps of the entire process shown in Figure 3.4. These last steps include the following:

- **Step 1:** The model is first trained on either point- or trip-based data. After this, the speed prediction accuracy is calculated of the trained model. This is done by comparing the predicted speeds of the trained model, based on the input variables, with the actual speeds in the test data set of the point-based data. The speed prediction accuracy of each neural network model will be calculated with the same error function as the used loss function. The speed prediction accuracy will be expressed in MSE_{Speed}, $MAPE_{Speed}$, MAE_{Speed}, $sMAPE_{Speed}$ and $sMdAPE_{Speed}$.
- **Step 2:** After the model is trained and evaluated on the speeds, the trained model can be used to predict the speed of each road in the map, based on its road properties (map data). In total, 3,304,031 speeds have to be predicted for the Netherlands and 5,169,618 for the Benelux. After all speeds of the roads in the map have been predicted, the map from either the Netherlands (new data) or the Benelux (old data) will be updated. The map update is the process of replacing the speeds of the roads in the map by the new predicted speeds. Subsequently, the travel times of the travels in the test set can be predicted. This is done by the routing software of ORTEC. It calculates the shortest path from the first to the last GPS point of the travel, after which the expected travel time is calculated. These predicted travel times are compared to the actual travel times in the test set. The travel time prediction accuracy will be expressed in $sMdAPE_{TT}$.

BENCHMARKS

The goal of this research is to improve the travel time prediction accuracy of den Heijer's models. Den Heijer's two best models will be used to compare the neural networks with and are the benchmarks. Den Heijer's two

best models are both based on random forest, with dependent variable speed and logspeed, and trained on point-based data. For convenience, the abbreviations Points-RF-Speed and Points-RF-Logspeed are used to indicate these models.

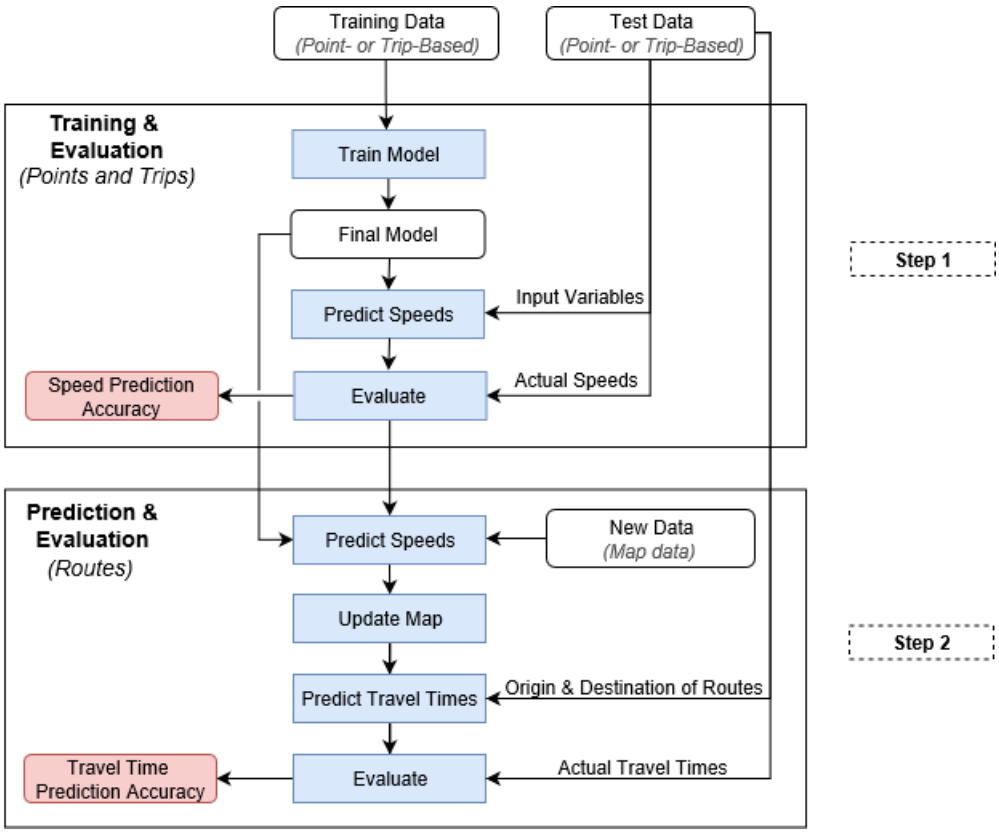

Figure 6.1: The last phase of the research. This includes the training of the models, calculation of the speed prediction accuracy, the prediction and update of the speeds in the map and finally the calculation of the travel time prediction accuracy.

6.2. RESULTS SPEED PREDICTION ACCURACY

Each of the 42 neural network models can be trained on either point- or trip-based data with the hyperparameters found in section 5.6. The training process is the optimization of the neural network model, such that the used loss function, MSE_{LF}, $MAPE_{LF}$, MAE_{LF}, $sMAPE_{LF}$ or $sMdAPE_{LF}$ is minimized. An example of the training process of the Points-NN-Speed-MSE model, trained on new data, is shown in Figure 6.2. This neural network model is trained on point-based data with output speed and loss function MSE_{LF}. In Figure 6.2, the training loss is the error between the predicted and the actual speeds in the training set. The validation loss is the error between the predicted and actual speed in the test data set (unseen data). The test set is not used to train the model, but for early stopping, to avoid over-fitting. If the validation loss has not improved after 10 iterations, then the training process is stopped. In Figure 6.2, it can be seen that the training loss, expressed in MSE_{LF}, keeps on improving until the training process is stopped at 89 epochs (iterations). From 60 epochs, the validation loss gets more or less constant and has reached its minimum at 79 epochs. The behavior of the training and validation loss in Figure 6.2 is as expected, based on the example shown Figure 5.4. The training process of the other models is more or less identical, where the validation loss gets constant and the training process is stopped after 10 epochs that the validation loss has reached its minimum.

The speed prediction accuracy of each neural network model for old and new data is shown in Table 6.3 and Table 6.4, respectively. The speed prediction accuracy is calculated with the same error function as the loss function, where the loss function is used to optimize the model. In addition, the speed prediction accuracy is calculated based on the speeds from the point-based in the test set. This means that the predicted logspeeds and paces are first converted back to speed, after which the speed prediction accuracy is calculated. In this way, the neural network models with the same loss functions can be compared. The speed prediction accuracy is either expressed in MSE_{Speed}, $MAPE_{Speed}$, MAE_{Speed}, $sMAPE_{Speed}$ or $sMdAPE_{Speed}$.

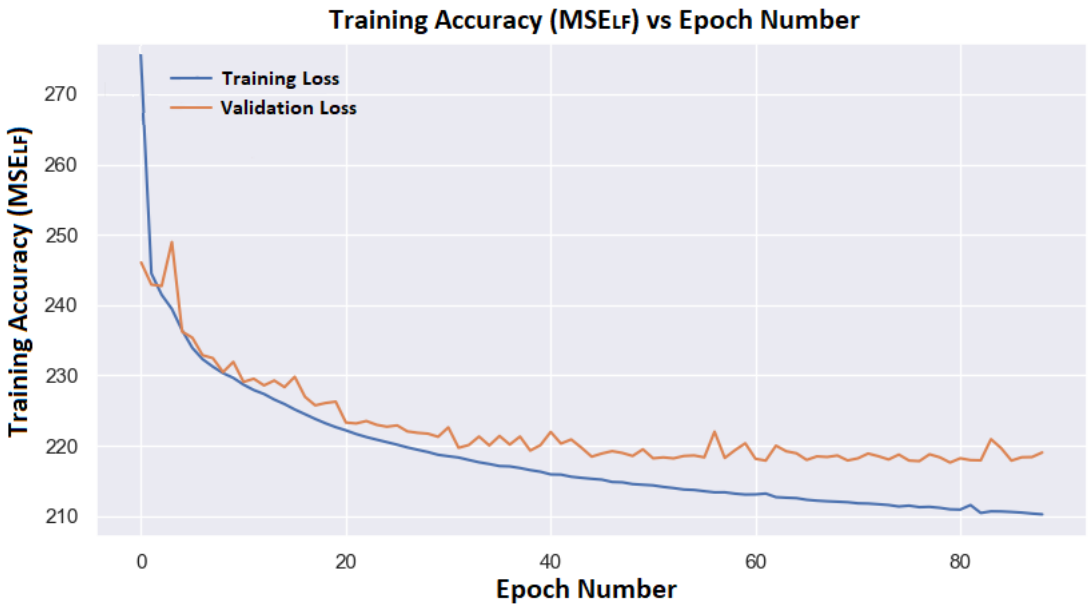

Figure 6.2: Training process of the Points-NN-Speed-MSE model with MSE_{LF} versus Epoch Number. Both the training and validation loss are shown, where the validation loss is used to avoid overfitting.

From Table 6.3 and Table 6.4, it can be concluded that the neural network models with dependent variable speed and point-based data, have the highest speed prediction accuracy with respect to the models with the same loss function. This is as expected, since the other models are trained with a different output (logspeed or pace) and data (trip-based), while the speed prediction accuracy is calculated for the speeds from point-based data in the test set.

Table 6.3: Overview of speed prediction accuracy for all 21 different neural network models for old data. The speed prediction accuracy is evaluated on the speeds from point-based data in the test set. The speed prediction accuracy is calculated with the same error function as the loss function used for each model.

		Speed Prediction Accuracy - Old data				
Dependent Variable	**Data Type**	**MSE_{Speed}**	**$MAPE_{Speed}$**	**MAE_{Speed}**	**$sMAPE_{Speed}$**	**$sMdAPE_{Speed}$**
Speed	points	255	-	9.88	32.5	7.61
Logspeed	points	330	-	10.0	32.6	8.12
Speed	trips	420	-	13.7	49.7	20.5
Logspeed	trips	480	-	13.6	48.9	21.1
Pace	trips	653	292	13.3	48.3	20.0

Table 6.4: Overview of the speed prediction accuracy for all 21 different neural network models for new data. The speed prediction accuracy is evaluated on the speeds from point-based data in the test set. The speed prediction accuracy is calculated with the same error function as the loss function used for each model.

		Speed Prediction Accuracy - New data				
Dependent Variable	**Data Type**	**MSE_{Speed}**	**$MAPE_{Speed}$**	**MAE_{Speed}**	**$sMAPE_{Speed}$**	**$sMdAPE_{Speed}$**
Speed	points	215	-	9.51	40.2	14.8
Logspeed	points	264	-	10.5	40.2	15.0
Speed	trips	391	-	14.7	43.7	18.0
Logspeed	trips	407	-	15.3	50.0	18.8
Pace	trips	669	331	16.6	49.7	21.7

6.3. RESULTS TRAVEL TIME PREDICTION ACCURACY

After the models are trained, the speed of each road in the map can be predicted with the trained neural network models. After this, the travel times can be predicted by finding the shortest path from the first to the last GPS point of each travel in the test set. Finally, the travel time prediction accuracy of each neural network model can be found by comparing the new predicted travel times, through new speed predictions, with the actual travel times in the test set. In section 5.1, it was discussed that the most suitable main indicator of the travel time prediction accuracy, which will be used, is the $sMdAPE_{TT}$. The reasons to use the $sMdAPE_{TT}$ as main indicator are:

1. To be able to compare the travel time prediction accuracy between all models, the travel time prediction accuracy has to be expressed with one and the same error function.
2. The $sMdAPE_{TT}$ has several properties that make this error function most suitable to calculate the travel time prediction accuracy. Examples are the insensitivity to outliers, due to the median, and equal penalization to positive and negative errors, due to its symmetry. In addition, den Heijer also used the $sMdAPE_{TT}$ to compare the travel time prediction accuracy of different models. This makes the comparison between den Heijer's best models and the new developed neural network models in this research easier.

In addition, the $sMdPE_{TT}$ and IQR sPE_{TT} will also be used to assess the travel time prediction accuracy. The $sMdPE_{TT}$ is an indicator whether the travel time predictions are under- or overestimated. The IQR sPE_{TT} indicates how much the errors are spread. A more detailed explanation of these error functions can be found in section 5.1.

6.3.1. COMPENSATION OF SYSTEMATIC BIAS

The results of the travel time prediction accuracy for all models are shown in Table H.1, in Appendix H. From these results, it can be concluded that the $sMdPE_{TT}$ for each model is not equal to 0%. This means that for a negative $sMdPE_{TT}$ the travel time predictions are underestimated (too fast), while a positive $sMdPE_{TT}$ means that the travel times are overestimated (too slow). This can be seen as a systematic bias where more than halve of the travel time predictions are under- or overestimated. There might be different reasons for the systematic bias:

- The shortest path, calculate by routing software from ORTEC, is not the same path as driven by the driver. The shortest path is calculated, based on the first and last GPS point of a travel.
- The prediction models optimize the speed of points and trips. This means that not the travel time, but instantaneous moments and short parts of the full travel are optimized.

From the research of den Heijer[1], it was concluded that the $sMdAPE_{TT}$ of each model improves after the $sMdPE_{TT}$ was moved to 0%. The $sMdPE_{TT}$ can be moved to 0% by multiplying the travel times or speeds assigned to each road in the map by a factor x. This multiplication is shown in Equation 6.3.1, where P_t is the predicted travel time or speed before compensation by x. $P_t^{'}$ is the predicted travel time or speed after compensation by x.

$$P_t^{'} = x \cdot P_t \tag{6.3.1}$$

Because the $sMdPE_{TT}$ is used to compensate the systematic bias of the travel time predictions, the value x will be based on the actual median and predicted median travel time. This results in Equation 6.3.2, where $P_{md}^{'}$ is the compensated median predicted travel time, P_{md} the median predicted travel time and A_{md} the actual median travel time.

$$P_{md}^{'} = x \cdot P_{md} = A_{md} \tag{6.3.2}$$

This results in a x that is equal to:

$$x = \frac{A_{md}}{P_{md}} \tag{6.3.3}$$

Starting from the sMdPE (Equation 6.3.4), x can be determined through a derivation that is shown from Equation 6.3.4 to Equation 6.3.10:

$$\text{sMdPE} = \text{median}\left(200\% \cdot \frac{P_t - A_t}{A_t + P_t}\right) \qquad t \in 1, ..., n \tag{6.3.4}$$

$$\text{sMdPE} = 200\% \cdot \frac{P_{md} - A_{md}}{A_{md} + P_{md}} \tag{6.3.5}$$

$$\frac{\text{sMdPE} \cdot (A_{md} + P_{md})}{200\%} = P_{md} - A_{md} \tag{6.3.6}$$

$$\frac{\text{sMdPE} \cdot A_{md}}{200\%} + A_{md} = P_{md} - \frac{\text{sMdPE} \cdot P_{md}}{200\%} \tag{6.3.7}$$

$$A_{md}\left(1 + \frac{\text{sMdPE}}{200\%}\right) = P_{md}\left(1 - \frac{\text{sMdPE}}{200\%}\right) \tag{6.3.8}$$

$$\frac{A_{md}}{P_{md}} = \frac{1 - \frac{\text{sMdPE}}{200\%}}{1 + \frac{\text{sMdPE}}{200\%}} \tag{6.3.9}$$

$$x = \frac{200\% - \text{sMdPE}}{200\% + \text{sMdPE}} \tag{6.3.10}$$

An example of the compensation of the systematic bias, sMdPE_{TT}, is shown in Figure 6.3. In this figure, the travel time prediction errors, in sPE_{TT}, before and after compensation of the sMdPE_{TT}, are shown for the Points-RF-Logspeed model. Before compensation of the sMdPE_{TT}, the sMdAPE_{TT} is equal to 28.9%. After compensation, the sMdAPE_{TT} improved significantly to 14.3%.

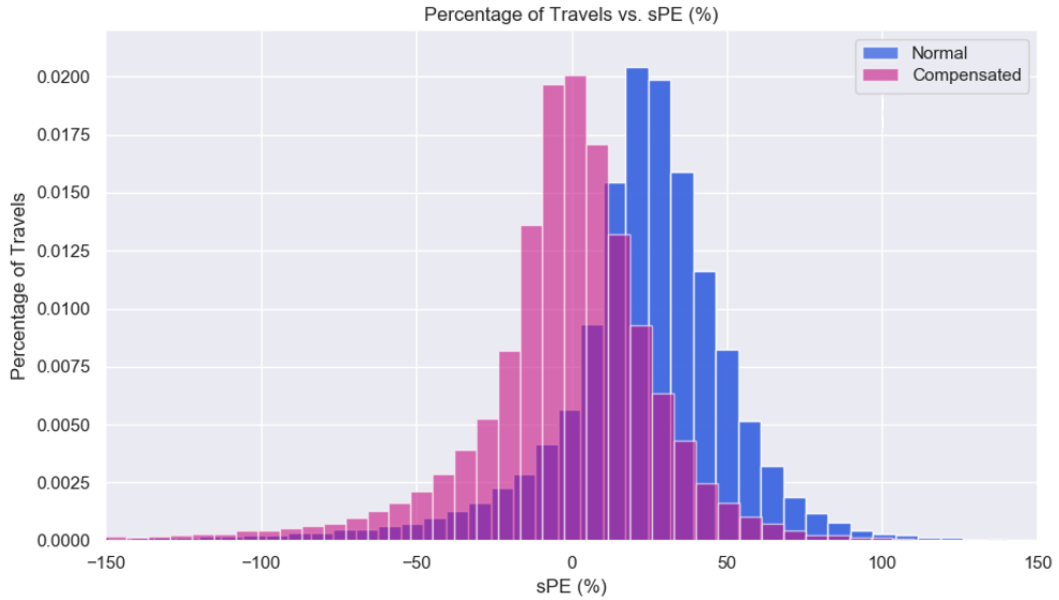

Figure 6.3: Distribution of the travel time prediction error in sPE_{TT} of the Points-RF-Logspeed model. The results are for new data before (normal) and after moving sMdPE_{TT} from 26.0 % to 0%.

6.3.2. RESULTS

The results of the compensated travel time predictions for all speed prediction models are shown in Table H.2, in Appendix H. In Table 6.5 and Table 6.6, an overview is shown of the speed prediction models with the highest travel time prediction accuracy for point- and trip-based data and for old and new data. The speed prediction models with the highest travel time prediction accuracy for old and new data are highlighted in purple. From Table 6.5 and Table 6.6, the following can be concluded:

- Both the Points-RF-Logspeed and Points-NN-Logspeed-MSE models have the highest travel time prediction accuracy for old data with $sMdAPE_{TT}$ 13.8%. The Points-RF-Speed model has the highest travel time prediction accuracy for new data with $sMdAPE_{TT}$ 12.4%. From these results, it can be concluded that all neural network models, for both the new and old data set, do not improve the travel time prediction accuracy of den Heijer's best models. This means that none of the neural networks is able to predict the speeds in such a way that the travel time prediction accuracy is improved.
- For old data, the best trip-based neural network model did not outperform the best point-based neural network model. For new data, the best trip-based neural network model, slightly outperformed the best point-based neural network model with $sMdAPE_{TT}$ 12.9% versus 13.0%. It seems that a lower $sMdAPE_{TT}$ can be obtained by neural networks, trained on trip-based data, with a higher data frequency (2 min. vs 5 min.). However, the best trip-based neural network model differ between old and new data. Also, from the results of all models in Table H.2, it can be concluded that some trip-based neural network models have a low $sMdAPE_{TT}$ for old data, but a high $sMdAPE_{TT}$ for new data or vice versa. It is unclear whether this is due to a different data frequency or different type of data set. More research need to be done to find the influence of the data frequency on the travel time prediction accuracy.
- The results show that the best speed prediction model, with respect to $sMdAPE_{TT}$, differs between old and new data. This means that there is not one best speed prediction model that can be used for different data sets. Also, Table H.1 and Table H.2 show inconsistent differences in $sMdAPE_{TT}$ between the same speed prediction models for old and new data. The cause seems to be the consistently higher $sMdPE_{TT}$, before compensation of the systematic bias, for new data. This means that all speed prediction models of the new data set, predict the travel times consistently slower, compared to the models of the old data set. Possible reasons will be analyzed in section 6.4.

Table 6.5: Overview of den Heijer's best models and the best neural network models for old and new data in $sMdAPE_{TT}$. The best model(s) for old and new data is/are highlighted.

	Training Data	Den Heijer's Best Model in $sMdAPE_{TT}$ (%)	Best Neural Network in $sMdAPE_{TT}$ (%)
Old Data	Point-Based	**13.8**	**13.8**
	Trip-Based	15.4	14.5
New Data	Point-Based	**12.4**	13.0
	Trip-Based	-	12.9

Table 6.6: Overview of den Heijer's best models and the best neural network models for old and new data. The best model(s) for old and new data is/are highlighted. The best models for old data are Points-RF-Logspeed and Points-NN-Logspeed-MSE, the best model for new data is Points-RF-Speed.

	Training Data	Den Heijer's Best Model	Best Neural Network
Old Data	Point-Based	**Points-RF-Logspeed**	**Points-NN-Logspeed-MSE**
	Trip-Based	Trips-LR-Logspeed	Trips-NN-Logspeed-MSE
New Data	Point-Based	**Points-RF-Speed**	Points-NN-Speed-MSE
	Trip-Based	-	Trips-NN-Pace-sMdAPE

6.3.3. TRAVEL TIME PREDICTION ACCURACY VERSUS SPEED PREDICTION ACCURACY

In this subsection, possible relationships between the speed and travel time prediction accuracy will be researched. Den Heijer concluded that the speed prediction accuracy, in $sMdAPE_{Speed}$, has no correlation with the travel time prediction accuracy in $sMdAPE_{TT}$. Because of the different loss functions that are used, it can be researched whether the speed prediction accuracy, expressed with a different error function than $sMdAPE_{Speed}$, is related to $sMdAPE_{TT}$.

The speed prediction accuracy in MSE_{Speed} showed the best relationship with the travel time prediction accuracy $sMdAPE_{TT}$. This is shown in Figure 6.4 and Figure 6.5, which include the neural network models with loss function MSE_{LF}. Also, the random forest models (benchmarks) are included, since the random forest models also use the MSE to optimize the model. The MSE_{Speed} is based on how well the speeds in the test set, from the point-based data, are predicted. The relationships of the other error functions MAE_{Speed}, $sMAPE_{Speed}$ and $sMdAPE_{Speed}$ with respect to $sMdAPE_{TT}$ can be found in Figure H.1, Figure H.2 and Figure H.3. $MAPE_{Speed}$ was not included, since only one neural network model for old and new data used the loss function MAPE ($MAPE_{LF}$). That are too few models to find possible relationships.

From the graphs in Figure 6.4 and Figure 6.5, and Figure H.1, Figure H.2 and Figure H.3, the following can be concluded:

- The clearest relationships between the speed and travel time prediction accuracy can be found between MSE_{Speed} and $sMdAPE_{TT}$, which seems to be linearly related. However, for old data, this seems not to be true for 200-300 MSE_{Speed} and for new data after 400 MSE_{Speed}. For new data, the model with the lowest MSE_{Speed} has the lowest $sMdAPE_{TT}$, while for old data the models with the third and fourth lowest MSE_{Speed} has the lowest $sMdAPE_{TT}$. Therefore, minimizing MSE_{LF} does not directly mean for all data sets, that the lowest $sMdAPE_{TT}$ will be obtained. However, the MSE_{Speed} can be used as a rough indicator for the $sMdAPE_{TT}$.
- The speed prediction models with the lowest MSE_{Speed} are Points-RF-Speed and Points-NN-Speed-MSE and shown in Figure 6.4 and Figure 6.5. This is as expected, because both models are trained on the output speed and point-based data, where the MSE_{Speed} is evaluated on. Remarkably, from Figure 6.4 and Figure 6.5, it can be concluded that the Points-RF-Speed model has a lower MSE_{Speed} than Points-NN-Speed-MSE. This is not as expected, since the neural networks are generally known as models with a high complexity, which increases the prediction accuracy. Since the hyperparameters of the neural network models are chosen carefully and tuned, it is not expected that other hyperparameter settings will further improve the MSE_{Speed} of the Points-NN-Speed-MSE model. Apparently, the relationships between the independent and dependent variables are not such, that a more complex model than the random forest, find better relationships. In addition, den Heijer found that also linear regression, which is a less complex model than random forest, does not find better relationships between the independent and dependent variables[1]. This means that the complexity of the relationships between the independent and dependent variables are best found by the random forest model.
- The relationships between $sMdAPE_{Speed}$ and $sMdAPE_{TT}$ in Figure H.3 confirm that these are weakly correlated, as concluded by den Heijer[1]. The results are contradictory, since the $sMdAPE_{TT}$ generally improves with a worse $sMdAPE_{Speed}$. Furthermore, the relationships between MAE_{Speed} and $sMAPE_{Speed}$ with respect to $sMdAPE_{TT}$ seem to be quite random. A lower MAE_{Speed} or $sMAPE_{Speed}$ does not necessarily result in a lower $sMdAPE_{TT}$. Therefore, these speed prediction accuracies should not be used as indicator for the travel time prediction accuracy.

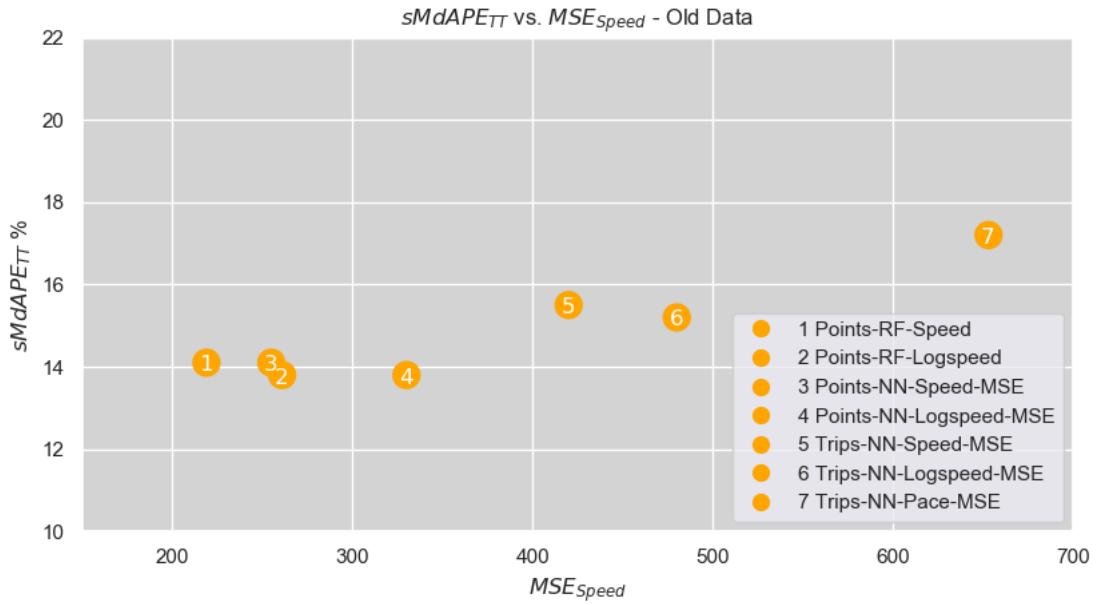

Figure 6.4: Travel time prediction accuracy in $\text{sMdAPE}_{\text{TT}}$ versus speed predictions in $\text{MSE}_{\text{Speed}}$ for old data. Only the models with loss function MSE_{LF} are included in the graph.

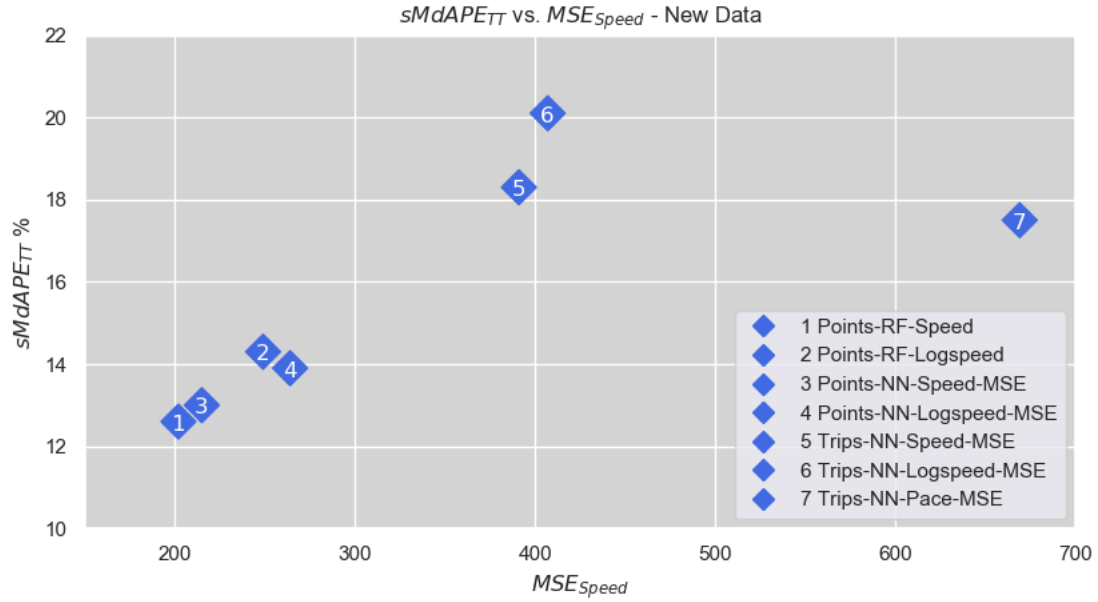

Figure 6.5: Travel time prediction accuracy in $\text{sMdAPE}_{\text{TT}}$ versus speed predictions in $\text{MSE}_{\text{Speed}}$ of new data. Only the models with loss function MSE_{LF} are included in the graph.

6.4. ANALYSIS RESULTS

In this section, it will be analyzed why the old and new data have different best speed prediction models with respect to $sMdAPE_{TT}$. From the results in Table H.1, it can be concluded that the $sMdPE_{TT}$ (systematic bias) for the new data, is consistently higher, compared to the same models of the old data. This means that the travel times of the new data set are consistently predicted slower, compared to the old data set. As a consequence, the $sMdAPE_{TT}$ differs between old and new data for the same speed prediction model. The reasons that may cause these differences and will be researched are:

1. Different starting times of travels throughout the day
2. The minimum travel time that is evaluated on
3. Number of travels that are evaluated on
4. Difference in country
5. Different size of training set

The Points-RF-Logspeed model will be used to analyze the consistently slower travel time predictions of the new data models, compared to the old data models. This model shows one of the largest differences in $sMdPE_{TT}$ between old and new data with $sMdPE_{TT}$ 4.2% and 26.0%, respectively. In Figure 6.6, the distribution of the travel time prediction accuracy in sPE_{TT} (%) is shown of the Points-RF-Logspeed model. As can be seen, the travel time predictions of the new data are predicted much slower (higher sPE_{TT}) than the old data.

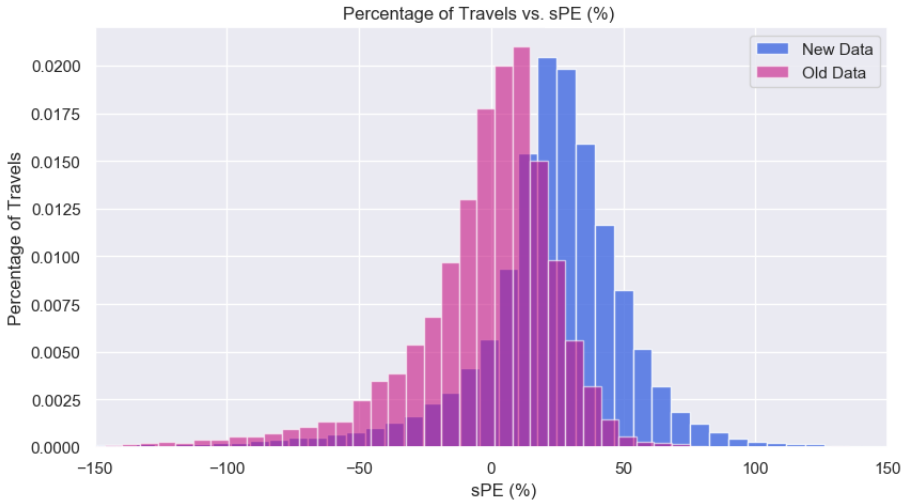

Figure 6.6: Distribution of the travel time prediction accuracy in sPE_{TT} of the Points-RF-Logspeed model for new and old data before compensation with $sMdPE_{TT}$. The $sMdPE_{TT}$ for old and new data is 4.2% and 26.0%, respectively.

1. A first reason for the large difference in $sMdPE_{TT}$ between the old and new data set, might be due to more travels during rush hours (7:00-9:00 & 16:30-18:30) in the old data set. Travels that are driven during rush hours may be predicted too fast leading too a lower $sMdPE_{TT}$. In Figure 6.7, a distribution of the starting time of each travel is shown. As can be seen, both data sets follow the same distribution and there is no significant difference in percentages of travels during rush. Therefore, the difference in working hours can be excluded as reason for the large differences in $sMdPE_{TT}$ between the new and old data set.

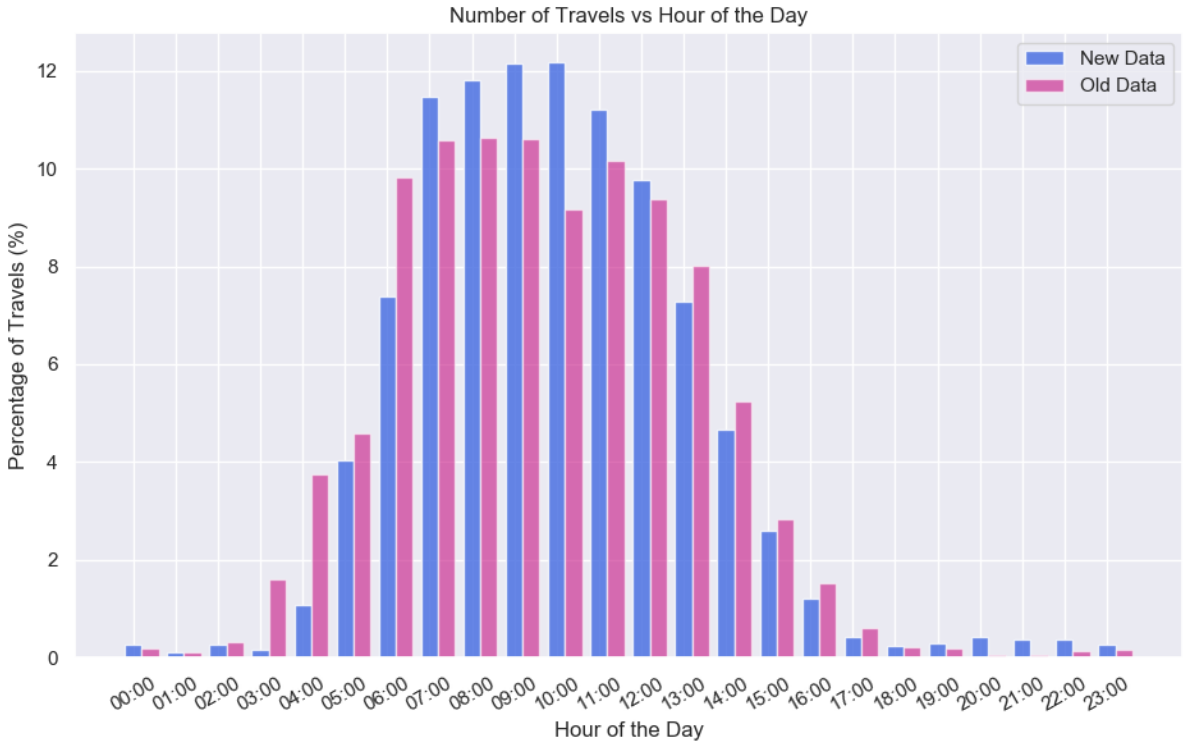

Figure 6.7: Distribution of the starting time of the travels throughout the day for the new and old data set.

2. A second reason for the large differences in $sMdPE_{TT}$ might be due to the minimum actual travel time that is evaluated on. The distribution of the actual driven travel times are shown in Figure 6.8. The minimum actual travel time for the new data is 2 minutes and for the old data 10 minutes. The influence of the minimum travel time on the accuracy is analyzed by removing travels that are shorter than 10 minutes from the new data. The results are shown in Table 6.7. It can be concluded that the $sMdPE_{TT}$ and $sMdAPE_{TT}$ improve with 5.5% and 5%, if only travel times larger than 10 minutes are evaluated, instead of 2 minutes. However, this improvement is still too small to explain the large difference in $sMdPE_{TT}$ between the old and new data set of 4.2% and 26.0% respectively.

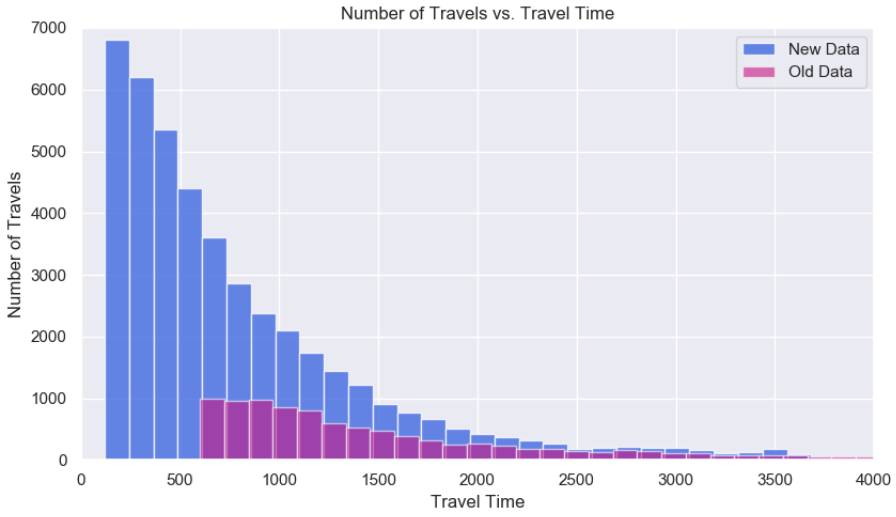

Figure 6.8: Distribution of the travel starting time throughout the day for the new data set.

Table 6.7: Influence minimum travel times on travel time prediction accuracy before compensation with sMdPE.

Model	New Data (All travel times)			New Data (travel time ≥ 10 min)		
	sMdAPE$_{TT}$ (%)	**sMdPE$_{TT}$ (%)**	**IQR sPE$_{TT}$ (%)**	**sMdAPE$_{TT}$ (%)**	**sMdPE$_{TT}$ (%)**	**IQR sPE$_{TT}$ (%)**
Benchmark Points-RF-Logspeed	28.9	26.0	28.1	23.9	20.5	20.7

3. A third reason might the number of travels that are evaluated on. To analyze the influence of the number of travels on the prediction accuracy, the number of travels of the new data set is reduced from 45,852 to 10,316. This is equal to the number of travels in the old data set. From the results in Table 6.8, it can be concluded that the number of travels used for evaluation, does not explain the large difference in sMdPE$_{TT}$ between the old and new data set of 4.2% and 26.0% respectively.

Table 6.8: Influence number of travels on accuracy before compensation with sMdPE.

Model	New Data (Evaluated on 45,852 travels)			New Data (Evaluated on 10,316 travels)		
	sMdAPE$_{TT}$ (%)	**sMdPE$_{TT}$ (%)**	**IQR sPE$_{TT}$ (%)**	**sMdAPE$_{TT}$ (%)**	**sMdPE$_{TT}$ (%)**	**IQR sPE$_{TT}$ (%)**
Benchmark Points-RF-Logspeed	28.9	26.0	28.1	27.5	24.7	27.9

4. A fourth reason for the large differences in sMdPE$_{TT}$, might be the country where the travels are driven (Netherlands vs. Benelux). This is analyzed by only evaluating the travels of the old data set that are driven in the Netherlands. In Table 6.9, the travel time prediction accuracy of the travel times in the Netherlands and the Benelux of the old data set is shown. It can be concluded that the sMdPE$_{TT}$ and sMdAPE$_{TT}$ improve from 4.2% and 14.8% to -1.7% and 13.5%. This shows that the sMdPE$_{TT}$ and sMdAPE$_{TT}$ do not come closer to the 26.0% and 28.9% of the new data set. Therefore, the country where the travels are driven is not the cause for a large difference in sMdPE$_{TT}$ and thus the difference in best speed prediction models of the old and new data.

Table 6.9: Influence country on travel time prediction accuracy before compensation with sMdPE.

Model	Old Data (Benelux)			Old Data (Netherlands)		
	sMdAPE$_{TT}$ (%)	**sMdPE$_{TT}$ (%)**	**IQR sPE$_{TT}$ (%)**	**sMdAPE$_{TT}$ (%)**	**sMdPE$_{TT}$ (%)**	**IQR sPE$_{TT}$ (%)**
Benchmark Points-RF-Logspeed	14.8	4.2	28.5	13.5	-1.7	28.2

5. A last reason might be that the large differences in sMdPE$_{TT}$ are caused by an smaller training set. This is analyzed by reducing the point-based data set, of the new data, from 1,083,237 to 410,137. Which is the size of the point-based data set of the old data. The results are shown in Table 6.10. It can be seen that the sMdPE$_{TT}$ and sMdAPE$_{TT}$ slightly improve if the model is trained on a smaller point-based data set. This result shows that the large differences in sMdPE$_{TT}$, between the new and old data set, are not caused by the smaller data sets used by the old data.

Table 6.10: Influence point-based data size on travel time predictions new data, before compensation with sMdPE.

Model	New Data (1,083,237 points)			New Data (410,137 points)		
	sMdAPE$_{TT}$ (%)	**sMdPE$_{TT}$ (%)**	**IQR sPE$_{TT}$ (%)**	**sMdAPE$_{TT}$ (%)**	**sMdPE$_{TT}$ (%)**	**IQR sPE$_{TT}$ (%)**
Benchmark Points-RF-Logspeed	28.9	26.0	28.1	27.5	24.5	27.3

Finally, by analyzing the old and new data, none of the five reason explained why all sMdPE$_{TT}$ for the new data are consistently higher than the old data. Therefore, it also does not explain why the old and new data have a different best speed prediction model. An explanation that is left over, is that the trucks of the new data drive faster than the trucks of the old data, which transport chemical packages. Apparently, the speed prediction models do not learn and predict the speeds for fast and slow trucks such that the same travel time prediction accuracy is obtained. This means that the best speed prediction model for fast trucks may be different compared to slow trucks.

6.5. RESULTS TEMPORAL FACTOR

The temporal factor could not be included in the neural network model as independent variable, since this would require the digital map to assign more speeds to one road for different moments in time. Therefore, the temporal factor will be researched by splitting the data set into rush (7:00-9:00 & 16:30-18:30) and non-rush hour data sets and training two different neural network models. This is shown through a flow chart in Figure 4.5. If more than halve of the travel is driven during rush hours, then all data of this travel will be put in the rush hours data set. If not, then the travel will be added to the non-rush hours data set. The rush and non-rush hours data sets will be both split into a training and test set. Only complete travels will be put into the training and test sets. This is done to make sure that during the training process, nothing is seen from the travels in the test set that will be evaluated on.

The number of points and trips in the rush and non-rush hours data sets, for point- and trip-based data, is shown in Table 6.11. From this table it can be concluded that about $1/5^{th}$ of the travels are driven during rush hours and $4/5^{th}$ outside the rush hours.

Table 6.11: Number of data points in rush and non-rush hour data sets for point- and trip-based data.

Model	Point-Based Data	Trip-Based Data
Rush and non-rush hours (total)	1,445,378	1,049,195
Rush hours	313,721	219,535
Non-rush hours	1,131,655	829,660

Three different models, Points-RF-Speed, Points-NN-Speed-MSE and Trips-NN-Speed-MSE, are retrained to research the influence of the temporal factor. In Table 6.12, the results of the speed prediction accuracy in MSE_{Speed} and travel time prediction accuracy in $sMdAPE_{TT}$ are shown. The speed prediction accuracy is expressed in MSE_{Speed}, since all models use loss function MSE_{LF} to optimize the speed prediction model. The best value for each model, in each column, is highlighted. It can be concluded that the speed and travel time prediction accuracy for all three models, which are trained on the non-rush hour dataset, is the best. This is as expected, because most traffic jams are excluded from the non-rush hour data set, reducing the variance in speeds. The speed prediction accuracy on the rush hour dataset, is the worst for all three models. This is also as expected, because of a high variance in speeds due to more vehicles on the roads.

Finally, the results of the models, which are trained on the rush and non-rush hour training set, can be added together to be able to asses the influence of the temporal factor. It can be seen in Table 6.12 that the MSE_{Speed} improves when rush and non-rush hours are taken into account (normal versus rush and non-rush hour). The $sMdAPE_{TT}$ of the Points-NN-Speed-MSE and Trips-NN-Speed-MSE model improves with 0.6% and 3.0%, respectively. The $sMdAPE_{TT}$ of the Points-RF-Speed model does not improve, however the IQR sPE_{TT} improves from 28.0% to 27.3% for this model. This means that the errors are less widely spread. As a conclusion, the speed and travel time prediction accuracy can be improved by taking rush and non-rush hours into account. However, the influence of the temporal factor decreases for models with an relatively high speed and travel time prediction accuracy.

Table 6.12: Speed and travel time prediction accuracy of Points-RF-Speed, Points-NN-Speed-MSE and Trips-NN-Speed-MSE model in MSE_{Speed}, $sMdAPE_{TT}$, $sMdPE_{TT}$ and IQR sPE_{TT}.

	New Data			
Model	**MSE_{Speed}**	**$sMdAPE_{TT}$ (%)**	**$sMdPE_{TT}$ (%)**	**IQR sPE_{TT} (%)**
Points-RF-Speed (normal)	202	12.6	0.0	28.0
Points-RF-Speed (only rush hour)	257	16.0	0.0	33.7
Points-RF-Speed (only non-rush hour)	**175**	**11.6**	0.0	**25.5**
Points-RF-Speed (rush and non-rush hour)	193	12.6	0.0	27.3
Points-NN-Speed-MSE (normal)	215	13.0	0.0	28.6
Points-NN-Speed-MSE (only rush hour)	267	15.7	0.0	30.8
Points-NN-Speed-MSE (only non-rush hour)	**176**	**11.4**	0.0	**27.4**
Points-NN-Speed-MSE (rush and non-rush hour)	195	12.4	0.0	28.4
Trips-NN-Speed-MSE (normal)	392	18.3	0.0	37.2
Trips-NN-Speed-MSE (only rush hour)	452	20.7	0.0	39.9
Trips-NN-Speed-MSE (only non-rush hour)	**356**	**14.0**	0.0	**33.5**
Trips-NN-Speed-MSE (rush and non-rush hour)	370	15.3	0.0	35.3

6.6. CONCLUSION

In this chapter, the following conclusions could be made:

- In total, 42 neural network models were trained. 21 models were trained on the old data and 21 models on new data. The models differ in type of training data, loss function and dependent variable. The neural network models were first trained on point- and trip-based data. Then, the models were evaluated on the speeds in the test set with the same error function that was used as loss function. This resulted in a speed prediction accuracy, where the models that were trained on point-based data and output speed had the highest speed prediction accuracy.
- After the speed of each road in the map was predicted, the travel times of the travels in the test set could be predicted. This was done by finding the shortest path from the first to the last GPS point of a travel. It was found that the Points-RF-Logspeed model and Points-NN-Logspeed-MSE model have the best travel time prediction accuracy for the old data with $sMdAPE_{TT}$ 13.8%. The Points-RF-Speed model has the best $sMdAPE_{TT}$ for new data with 12.4%. Remarkably, none of the neural network models did improve the travel time prediction accuracy $sMdAPE_{TT}$ of the models of den Heijer (benchmarks). This is not as expected, since neural networks are known as models with high prediction accuracies. This means that the developed neural network models are not able to predict the speeds in such a way, that the travel time prediction accuracy is improved.
- An attempt was made to find relationships between the speed and travel time prediction accuracy. The speed prediction accuracy of each neural network model was expressed in the same error function as the loss function used, to optimize the neural network model. It was found that the MSE_{Speed} is mostly related to $sMdAPE_{TT}$. In general, a lower MSE_{Speed} results in a lower $sMdAPE_{TT}$. For new data, the speed prediction model with the lowest MSE_{Speed} has the lowest $sMdAPE_{TT}$. However, this is not true for the old data. Therefore, the relationship between MSE_{Speed} and $sMdAPE_{TT}$ is not fully related. However, MSE_{Speed} can be used as rough indicator for $sMdAPE_{TT}$.
- It was expected that by increasing the data frequency from 5 to 2 minutes, a higher travel time prediction accuracy could be achieved with respect to the benchmarks. However, from the results it was unclear whether a higher data frequency improves the $sMdAPE_{TT}$. More research need to be done to find the influence of the data frequency on the travel time prediction accuracy.
- The best speed prediction model, with respect to $sMdAPE_{TT}$, differs between the old and new data. This means that there is not one best speed prediction model that can be used for different data sets. The cause seems to be the consistently higher $sMdPE_{TT}$, before compensation of the systematic bias, for new data. This means that all speed prediction models of the new data set, predict the travel times consistently slower, compared to the models of the old data set. The most likely reason is that the speed prediction models behave differently for different data sets, due to different type of trucks (food versus chemical packages) and routes from the customer.
- By taking rush and non-rush hours into account, a better speed and travel time prediction accuracy can be obtained. The $sMdAPE_{TT}$ of the Points-NN-Speed-MSE and Trips-NN-Speed-MSE model improved with 0.6% and 3.0%, respectively. The $sMdAPE_{TT}$ of the Points-RF-Speed model did not improve, however the IQR sPE_{TT} improved from 28.0% to 27.3%. This means that the errors are less widely spread. It can be concluded that the influence of the temporal factor decreases for models with an relatively high speed and travel time prediction accuracy. By taking into account more temporal factors (season, day of the week, holidays) and weather factors (rain, snow, wind speed), it is expected that even better travel time prediction accuracies can be obtained.

7

CONCLUSION & RECOMMENDATIONS

In this chapter, the conclusion and recommendations are discussed. Conclusions of the research will be made by answering each sub research question, after which the main research question will be answered. The recommendations are split into two sections. First, the recommendations for ORTEC will be discussed and secondly, the recommendations for further research.

7.1. CONCLUSION

The answers to the sub research questions, which are essential to answer the main research question, are the following:

1) How are travel times currently estimated at ORTEC and what are the shortcomings?
To calculate the travel time at ORTEC, first the shortest path has to be found. From the shortest path, the travel time can be derived. This is the sum of the length, divided by the speed, of each road that is included in the shortest path. Therefore, it is important that the predicted speed for each road in the road network is accurate to obtain accurate travel time predictions. Currently, each road in the map belongs to one of the 20 road types, where each road type has a speed that is assigned by the customer. The shortcomings of this current method are that the classification of the 20 road types is quite rough, and that the assigned speeds to each road type is based on experience of the customer. Both shortcomings lead to a reduction in travel time prediction accuracy.

Recently, den Heijer [1] developed a new method at ORTEC, based on GPS data, to improve the speed predictions in the map. Based on this new method, eight different speed prediction models were developed that were either based on linear regression or random forest. One model, which was based on random forest, improved the accuracy of the current method (20 road types) significantly with $\text{sMdAPE}_{\text{TT}}$ 4.6%. The developed models by den Heijer, predict the speed for each road in the map based on found relationships. These relationships were found between the driven speed, obtained from the GPS data, and the properties of the corresponding road, such as speed limit, speed bumps, lane width, etc. The shortcoming of the linear regression model is that only linear relationships can be modelled, which limits the prediction accuracy. The shortcoming of the random forest model is that it is not able to learn from trip-based data. A more accurate model than linear regression, which is able to learn from trip-based data, may lead to a higher travel time prediction accuracy.

2) Which related literature and researches are available, including available travel time prediction methods and influential travel time factors?
The travel time prediction methods in literature, except the method from den Heijer, focus on improving the travel time predictions for one route or road. This research study focuses on improving travel time predictions for an entire road network. Therefore, the travel time prediction method, developed by den Heijer, is only applicable to this research. However, to improve the method of den Heijer, prediction methods that are used by the travel time prediction methods in literature, can be adapted to this research problem. The prediction methods that can be used are linear regression, random forest, support vector regression, gradient

boosting and neural networks.

The influential factors of the travel time, which were found in literature, can be divided into five categories. These categories are: vehicle classification, temporal factors, weather factors, road engineering factors and unpredictable factors. For this research, the vehicle classification is fixed to trucks and the weather and unpredictable factors are considered to be out of scope. The temporal and road engineering factors were used to improve the speed predictions. Furthermore, the most suitable error function that was found to measure the travel time prediction accuracy, is $sMdAPE_{TT}$. This is the symmetric median absolute percentage error. In addition, the $sMdPE_{TT}$ and IQR sPE_{TT} were used as indicator whether the predictions are too fast or too slow and how widely the errors are spread, respectively.

3) Which prediction method is most suitable to predict the speeds?
A method trade-off was performed to choose the most suitable prediction method for the speeds in this research. The criteria that were used for the method trade-off are: 'prediction accuracy', 'hyperparameter tuning', 'interpretability' and 'large data set'. After performing the method trade-off and a sensitivity analysis, it was found that the neural network is the most suitable prediction method, for both point- and trip-based data.

4) How can a prediction method be developed for this problem?
The neural network can be used as prediction method by finding relationships between the independent variables (speed limit, region, lane width, etc.) and the dependent variable. In total, 21 independent variables were used to find relationships between the road engineering factors and speeds in the data sets. This is the training process. After the model is trained, the speed of each road in the map can be predicted individually. In total, 21 different neural network models were developed. These models differ in type of training data (point- or trip-based data), loss function (MSE_{LF}, $MAPE_{LF}$, MAE_{LF}, $sMAPE_{LF}$ and $sMdAPE_{LF}$) and dependent variable (speed, logspeed and pace (1/speed)). During hyperparameter tuning, it was found that the best number of neurons and hidden layers differ for each model. This depends on whether the dependent variable pace or loss function $sMdAPE_{LF}$ was used.

5) How does the new model compare to the current models?
In total, 21 neural network models were developed and trained on the old and new data set. The old data set is collected and preprocessed by den Heijer with a frequency of 5 minutes. The new data set is collected and preprocessed during this research with a frequency of 2 minutes. Den Heijer's two best models, which are based on random forest, were used as benchmarks. From the results, it was concluded that none of the neural networks had an improved travel time prediction accuracy, compared to the two random forest models. This is true for both old and new data. This means that the neural network is not able to learn and predict the speeds in such a way, that the travel time prediction accuracy is improved.

Main Research Question: Can a new speed prediction model be developed for trucks that outperforms the travel time prediction accuracy of the current speed prediction models for a given road network?
Based on the results that were obtained in this research, a new speed prediction model that outperforms the current speed prediction models, with respect to $sMdAPE_{TT}$, could not be developed. This is true for both the old and new data set. For both old and new data, one of the two random forest models, developed by den Heijer, had the best performance with $sMdAPE_{TT}$ 13.8% and 12.4%. However, it can be neither concluded that the neural network does not provide better travel time predictions for other data sets. Also, it cannot be concluded that neural networks, which are trained on trip-based data, will not outperform the random forest models with a higher data frequency than 2 minutes.

7.2. RECOMMENDATIONS FOR ORTEC

From this research, the following recommendations for ORTEC were found:

- From the results in this research, it can be concluded that none of the neural network models outperforms the $sMdAPE_{TT}$ of the random forest models of den Heijer. These random forest models are Points-RF-Speed and Points-RF-Logspeed. However, the best random forest model differs between the old and new data. If the best travel time prediction accuracy wants to be achieved for each customer, then it is recommended to train both random forest models and pick the best model subsequently.
- In this research, the temporal factor was researched by splitting the data set into rush and non-rush hours. It was found that the improvement in travel time prediction accuracy, of the best speed predic-

tion model for the new data, is not significant. Currently, ORTEC is considering to move from a static to a dynamic routing algorithm, where more speeds can be assigned to one road. Therefore, ORTEC should do more research about the influence of other temporal factors (holidays, day of week, etc.), to know whether the investment is worth it.

- More GPS data from different customers should be researched whether the random forest models are consistently the best models. Also, it should be researched whether GPS data from one customer, can be used for another customer with the same type of truck. This would save a lot of time to collect and clean GPS data for each customer individually.

7.3. FURTHER RESEARCH

After this research, the following suggestions were found to further improve the travel time prediction accuracy:

- After this research, it cannot be concluded whether a higher data frequency of 2 minutes, compared to 5 minutes, did improve the travel time prediction accuracy of neural network models that were trained on trip-based data. The difference in data sets, due to different vehicles and routes, seemed to have a too large impact on the results to draw conclusions about the influence of the GPS data frequency on the results. Therefore, it is recommended to research different data frequencies for the same data set.
- None of the neural network models that was trained on trip-based data, with a frequency of 2 and 5 minutes, had improved the travel time prediction accuracy of the benchmarks. Therefore, it is recommended to research data frequencies that are higher than 2 minutes, such as 15 or 30 seconds. A higher frequency may produce better results and outperforms the benchmarks. This is because a higher frequency gives more certainty about the driven path between two consecutive GPS points. This improves the quality of the training data. Also, a higher frequency means that the (average) independent variables of a trip, are based on less roads. This makes it easier for the model to extract information for the speed predictions of single roads.
- By including more influential factors of the travel time, higher travel time prediction accuracies might be obtained. Factors that can still be researched are other temporal factors such as day of the week, season, holidays, hour of the day, etc. But also weather factors such as rain, snow, wind speed, etc. can be included to improve the travel time predictions. However, including weather factors is quite a difficult job, which also requires access to historical weather data.
- It would be interesting to research the minimum amount of data required, until the random forest and neural network are saturated. If too little data is used, the model has not found all possible relationships. However, if too much data is used, the training time of a neural network can increase significantly.
- The models developed during the previous research by den Heijer and this research, are based on ML methods. These methods find relationships between the road properties and the speed. However, the final goal is to improve the travel time predictions. Therefore, more research can be done, to develop a new method that is able to learn and predict the speeds in such a way, that it is directly related to the travel time.

A

Academic Paper

Improving the travel time prediction accuracy for trucks based on historical GPS data

R.A. Weerheim, Dr. ir. X. Jiang, Prof. dr. R.R. Negenborn

Technical University Delft
R.A.Weerheim@student.tudelft.nl

***Abstract*— Route planning is an important part for companies that transport goods between different locations. To optimize the route planning, it is important that the travel time predictions come close to reality. A too tight schedule would lead to pressure to the driver, late fees and reputation damage, but a too loose schedule would result in wasted capacity. To predict the travel time, first a speed has to be assigned to each road in the digital map, after which the shortest path can be calculated. This research proposes a speed prediction model based on neural networks to predict the speed of each road such that the travel time prediction accuracy is improved. The neural network speed prediction model will be compared to the current speed prediction model, which is based on random forest. The speed prediction model first learns from preprocessed GPS data that is obtained from two different companies that operate with trucks. After relationships have been found between the driven speed and road properties (e.g. speed limit, road width, tunnel, etc.), the speed prediction model predicts a speed for each road in the digital map. Subsequently, the shortest path can be calculated from the first to the last GPS point of different routes. Then the predicted travel times from the shortest paths were compared to the real driven travel times. After comparison, it turned out that for both GPS data sets, the neural network speed prediction model did not outperform the travel time prediction accuracy in $\text{sMdAPE}_{\text{TT}}$ of the random forest speed prediction model.**

I. Introduction

With the growing population and the demands for products and goods, the transportation by trucks is increasing[1]. To cope with this increase, an accurate route planning is key to do the transportation between different locations efficiently and to save transportation cost as a result. To optimize the route planning, it is important that the travel time predictions come close to reality. A too tight schedule would lead to pressure to the driver, late fees and reputation damage, but a too loose schedule would result in wasted capacity. More addresses could be visited or less vehicles could be used.

This research is conducted at *ORTEC*, where the travel time prediction between point A and B is based on the sum of the length, divided by the speed, of each road that is included in the shortest path. Since the length of each road is fixed, the speed that is assigned to each road in the road network has to be improved to improve the travel time prediction accuracy. Currently, each road in the map belongs to one of the 20 road types, where each road type has a speed that is assigned by the customer. These speeds are based on the experience of the implementation consultant (customer). However, dividing the roads in the map into 20 road types, is quite a rough road classification. Also, the speeds that are assigned to each road type is based on the experience from the customer. This reduces the accuracy of the travel time predictions.

An easy and obvious option to improve the travel time prediction would be to use available routing software such as *Google maps*, *Bing maps*, etc., which can provide accurate travel time predictions. However, these routing softwares cannot be used by *ORTEC*, because they do not support many-to-many queries (shortest paths between all sources s and targets t), which is needed to solve the vehicle routing problem (VRP). Also, the computational time of these routing softwares is high and undesired[2], since the customers have a limited amount of time for the planning and scheduling process. Lastly, these routing softwares do only route calculations for cars, while route calculations for trucks are desired due to the majority of customers that operate with trucks.

At *ORTEC*, a first attempt was made to improve the travel time predictions based on historical GPS data and machine learning (ML) algorithms by den Heijer[3]. GPS data is used, since it contains information about the driven speed of the vehicles at many roads and plenty of historical GPS data can be obtained from the customer. Eight different models were developed by den Heijer and were either based on linear regression (LR) or random forest (RF). The developed models predict the speed for each road in the digital map based on found relationships. These relationships were found between the driven speed, obtained from the GPS data, and the properties of the corresponding road, such as speed limit, speed bumps, lane width, etc. An illustration of this process can be found in Figure 1.

However, the shortcoming of the linear regression models is that only linear relationships can be modelled. This limits the prediction accuracy when it learns from point- or trip-based data. The shortcoming of the random forest model is that it is not able to learn from trip-based data. A more accurate model than linear regression, which is able to learn from trip-based data, may lead to a higher travel time prediction accuracy.

This paper presents a new speed prediction model which will answer the following main research question: *Can a new*

speed prediction model be developed for trucks that outperforms the travel time prediction accuracy of the current speed prediction models for a given road network?

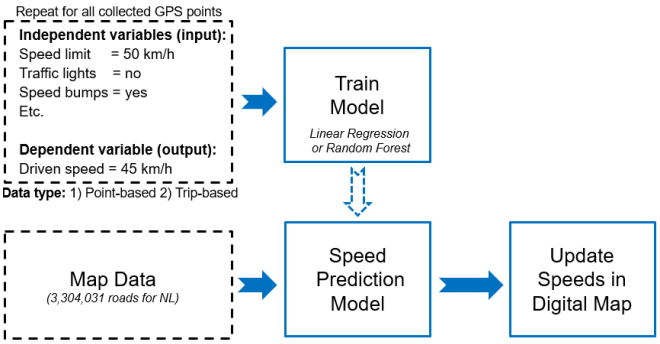

Fig. 1: Illustration how the speed prediction model is developed and used to predict a new speed for each road in the digital map.

II. Prediction Method

Besides the research of den Heijer[3] at *ORTEC*, also other travel time prediction methods were reviewed in literature. Bai et al. [4] provides an extensive review of different travel time prediction methods. However, these methods only focus on one route or road instead of an entire road network. Because this review of Bai et al. is incomplete, additional travel time prediction methods were investigated which seemed to be applicable for a road network and thus this research[5][6][7]. However, these methods focuses only on a couple of individual roads in a road network and are therefore also not suitable for this research. Therefore, it is decided to improve the travel time prediction method, developed by den Heijer[3], for this research problem based on its shortcomings.

The method of den Heijer can be improved by using a different prediction method, than linear regression or random forest, for the speed prediction model. Other suitable prediction methods are: support vector regression (SVR), Gradient Boosting (GB) and neural networks (NN). These prediction methods are suitable, since they can predict a continuous output (speed) and are able to generalize speeds from a data set to an entire road network. In Table I, an method trade-off can be found that helps to select the most appropriate prediction method. The neural network has the highest total score and will therefore be used for the speed prediction model for point- and trip-based data. Since, the input data of the speed prediction model is not sequential, the feed-forward neural network will be used. Other types of neural networks, such as the recurrent or long short-term memory neural network are only suitable for sequential input data and therefore not applicable to this research.

TABLE I: Trade-off between multiple prediction methods based on different criteria.

Criteria	Weight	LR	RF	SVR	GB	NN
Prediction Accuracy	5	1	3	4	4	5
Hyperparameter Tuning	2	5	4	2	2	1
Interpretability	1	5	3	2	3	1
Large Dataset	5	4	4	1	4	5
Total		40	46	31	47	**53**

III. Available Data

The data that is available for this research are map data and GPS data:

1) **Map data:** This is the data that *ORTEC* has purchased from *HERE Technologies*. The map data contains information about the road network and many properties of each road such as speed limit, road width/length, tunnel, etc. The map data is clean and well structured.
2) **Customer GPS data:** The GPS data is collected from customers of *ORTEC* that operate with fleets of trucks. This data is raw and of relatively low quality. Much effort is required for cleaning and preprocessing to be able to use this data.

To use the neural network as speed prediction model, a training and test set are required. The training set is used to find relationships between the independent (input) and dependent (output) variables. The test set is used to evaluate the model. To obtain the training and test set, first a few preprocessing steps have to be followed as shown in Figure 2.

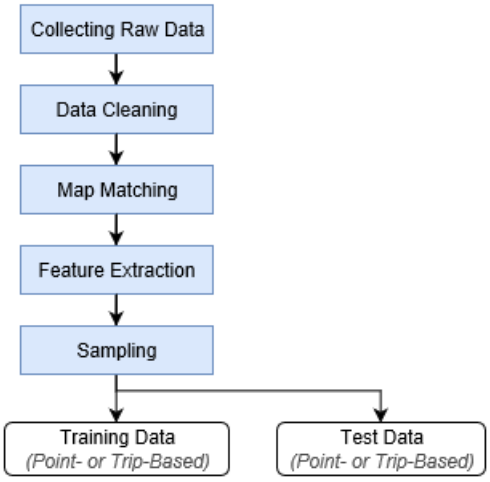

Fig. 2: Data preprocessing steps.

Den Heijer [3] used both point- and trip-based data to train the model. In den Heijer's research, point-based data performed better than trip-based data to train the speed prediction model. However, by using a neural network model and a higher data frequency (2 min. vs 5 min.), a higher travel time prediction accuracy may be obtained using trip-based data. Therefore, both point- and trip-based data will be used in this research. The differences between point- and trip-based data are:

- **Point-based data:** This type of data contains the road properties and the corresponding travel speed of each recorded GPS point. The road properties are the independent variables (model input) and the travel speed is the dependent variable (model output). This type of data is also shown in Figure 1.
- **Trip-based data:** The advantage of trip-based data is that it contains information about all roads of each travel. This is not the case for point-based data, where the information is based on a few roads where the GPS points are recorded. Each trip-based data point is based on all roads between two consecutive GPS points. This needs an additional preprocessing step. This step includes the calculation of the average of the road properties (independent variables) of all roads that belong to each trip.

A. Collecting Raw Data

For this research, two different GPS data sets are used. One GPS data set is collected and preprocessed during this research and is called 'new data' for convenience. This GPS data set comes from a customer that operates with trucks in the Netherlands and has a data frequency of 2 minutes. The other GPS data set is collected and preprocessed during the research of den Heijer[3], and is called 'old data'. This GPS data set comes from a customer that operates with trucks in the Benelux and has a data frequency of 5 minutes. Both GPS data sets are used because of the following reasons:

- **New data:** This GPS data has a frequency of 2 minutes, which is higher than a frequency of 5 minutes from the old data. It is expected that the quality of the trip-based data will improve with a higher GPS data frequency. As a consequence, the speed predictions of the neural network can be improved, which may result in better travel time predictions. In this way, the travel time predictions of the random forest speed prediction models may be outperformed.
- **Old data:** Den Heijer already collected and preprocessed GPS data with a frequency of 5 minutes which is ready to use. The data is obtained from a different customer than the new data. By using the old data as well, more insights can be obtained about the developed speed prediction models. Is the best speed prediction model for the new data the same as for the old data? Can a neural network, trained on trip-based data, already outperform the random forest speed prediction models with a GPS data frequency of 5 minutes?

B. Data Cleaning

After all raw GPS data is collected from the customer, the data needs to be cleaned so the model only learns from representative data. In total, eight data cleaning steps were applied to both GPS data sets. Examples of data cleaning steps are: 1) All GPS points that are not recorded during a travel are discarded 2) Travels that only contain one GPS point are removed 3) Travels that contain a ferry are removed and considered to be out of scope.

C. Map Matching

Map matching is the process of matching GPS data, consisting of a latitude and longitude component, to locations in the digital map. To save a significant amount of time on the development and implementation of a map matching algorithm, the online map matching service from *HERE* is used. This map matching service is called *Fleet Telematics Route Matching*. A sequence of GPS points of an unique travel are sent to the API and are matched to the most likely driven roads in the map. The most likely driven roads are determined by the longitude, latitude, timestamp, speed, heading and the other GPS points of the same travel.

D. Feature Extraction

After matching the GPS data to locations in the digital map, features (independent variables) can be extracted. These independent variables will be used to train the model and to predict the speed of each road in the map. For this research, road engineering factors (road properties) are used and are available from the map data. In Table II, an overview can be found of the used independent variables. Also the range of the values of the independent variables are shown for point- and trip-based data, which are either Boolean, continuous or discrete.

TABLE II: Independent variables of the neural network model, including the range of values for point- and trip-based data.

Independent Variables	Range Points	Range Trips
Functional Class 1	0 or 1	[0,1]
Functional Class 2	0 or 1	[0,1]
Functional Class 3	0 or 1	[0,1]
Functional Class 4	0 or 1	[0,1]
Speed Limit	[5,130]	[5,130]
Tunnel	0 or 1	[0,1]
Bridge	0 or 1	[0,1]
Traffic Signal	0 or 1	[0,1]
Urban	0 or 1	[0,1]
Speed Bumps	0 or 1	[0,1]
Road Width	> 0	> 0
Lane Category 1	0 or 1	[0,1]
Lane Category 2	0 or 1	[0,1]
Paved	0 or 1	[0,1]
Road Length	> 0	> 0
Ramp	0 or 1	[0,1]
Priority Road	0 or 1	[0,1]
Speed Pattern Max	[0,130]	[0,130]
Speed Pattern Min	[0,130]	[0,130]
Speed Pattern Avg	[0,130]	[0,130]
Speed Pattern Monday 8:30	[0,130]	[0,130]

E. Sampling

After the collection and preprocessing of the data sets, the data sets are randomly split into a training and a test data set. The training set will be used to train the model and the test set to evaluate the model on points/trips and complete travels. 75% of the full data set will be used as training set and 25% as test set.

IV. Model Design

The overall objective of the model is to predict the speed of each road in the digital map such that the travel time prediction accuracy of the current speed prediction models, developed by den Heijer, is outperformed for a given road network. This will be done by learning from points and trips obtained from GPS data, which contain valuable information about the driven speed, and the travel time, at many different locations in the road network.

A. Travel Time Prediction Accuracy

The travel time prediction accuracy will be expressed in $\text{sMdAPE}_{\text{TT}}$ and is shown in Equation 1. Where n is the total number of travels, A_t the actual travel time and P_t the predicted travel time for travel t. The $\text{sMdAPE}_{\text{TT}}$ has several properties that make this error function most suitable to calculate the travel time prediction accuracy. Examples are the insensitivity to outliers, due to the median, and equal penalization to positive and negative errors, due to its symmetry. In addition, the sMdPE_{TT} and IQR sPE_{TT} will also be used to assess the travel time prediction accuracy. The sMdPE_{TT} is an indicator whether the travel time predictions are under- or overestimated. The IQR sPE_{TT} indicates how much the errors are spread.

$$\text{sMdAPE}_{\text{TT}} = \text{median}\left(200\% \cdot \left|\frac{A_t \text{ - } P_t}{A_t \text{ + } P_t}\right|\right) \quad \text{t} \in 1, ..., n \tag{1}$$

$$\text{sMdPE}_{\text{TT}} = \text{median}\left(200\% \cdot \frac{P_t \text{ - } A_t}{A_t \text{ + } P_t}\right) \quad \text{t} \in 1, ..., n \tag{2}$$

$$\text{IQR sPE}_{\text{TT}} = \text{Q3 sPE}_{\text{TT}} - \text{Q1 sPE}_{\text{TT}} \tag{3}$$

$$\text{sPE}_{\text{TT}_\text{t}} = 200\% \cdot \frac{P_t \text{ - } A_t}{A_t \text{ + } P_t} \qquad \text{t} \in 1, ..., n \tag{4}$$

B. Neural Network Models

In Table III, an overview can be found of 21 different neural network models that will be used for the old and new data to predict the speeds. They differ in dependent variable, type of training data and loss function.

TABLE III: Overview of all 21 different neural network speed prediction models that will be researched to find the best travel time prediction accuracy.

Dependent Variable	Data Type	MSE_{LF}	MAPE_{LF}	MAE_{LF}	sMAPE_{LF}	$\text{sMdAPE}_{\text{LF}}$
Speed	points	x		x	x	x
Logspeed	points	x		x	x	x
Speed	trips	x		x	x	x
Logspeed	trips	x		x	x	x
Pace	trips	x	x	x	x	x

- **Dependent variable:** The dependent variables that will be used as model output are speed, logspeed and pace (1/speed). Speed is an obvious choice, since the speed is obtained from the GPS data and the speed of each road in the road network has to be improved. However, non-relative loss functions, such as the mean squared error (MSE) and mean absolute error (MAE), fit better to larger values. Therefore, the logspeed (logarithmic speed) is also used to move the speeds to a relative space. This gives the same relative error for low and high speeds. Furthermore, for trip-based data, the time could also be used as dependent variable, since the time between two GPS points is known. However, this requires the neural network to extrapolate. A solution is to divide the (in)dependent variables by the distance between two GPS points, which results in the dependent variable pace. In this way the data is normalized and does not need to extrapolate.
- **Data type:** In den Heijers research[3], point-based data performed better than trip-based data. However, by using a neural network model and a higher data frequency (2 min. vs 5 min.), a higher travel time prediction accuracy may be obtained using trip-based data. Therefore, both point- and trip-based data will be used in this research.
- **Loss function:** It would be logical to use the sMdAPE as loss function ($\text{sMdAPE}_{\text{LF}}$) to optimize the speed predictions, since $\text{sMdAPE}_{\text{TT}}$ is used for the travel time prediction accuracy. However, den Heijer[3] concluded that the speed prediction accuracy in $\text{sMdAPE}_{\text{Speed}}$ is not related to $\text{sMdAPE}_{\text{TT}}$. Therefore, also other common loss functions will be used, which may be more related to $\text{sMdAPE}_{\text{TT}}$. These are MSE_{LF}, MAE_{LF}, MAPE_{LF} and sMAPE_{LF}. These loss functions are shown from Equation 5 to Equation 8, where A_t is the actual value of the dependent variable, P_t the predicted value of the dependent variable and n the number of observations. This may result in an improved $\text{sMdAPE}_{\text{TT}}$, compared to the random forest speed prediction models of den Heijer.

$$\text{MSE}_{\text{LF}} = \frac{1}{n} \cdot \sum_{t=1}^{n} (A_t \text{ - } P_t)^2 \tag{5}$$

$$\text{MAE}_{\text{LF}} = \frac{1}{n} \cdot \sum_{t=1}^{n} |A_t - P_t| \tag{6}$$

$$\text{MAPE}_{\text{LF}} = \frac{100\%}{n} \cdot \sum_{t=1}^{n} \left|\frac{A_t - P_t}{A_t}\right| \tag{7}$$

$$\text{sMAPE}_{\text{LF}} = \frac{200\%}{n} \cdot \sum_{t=1}^{n} \left|\frac{A_t - P_t}{A_t + P_t}\right| \tag{8}$$

C. Neural Network Architecture

A visualization of the neural network architecture, including the independent and dependent variables, is shown in Figure 3. The number of neurons n and hidden layers m have to be determined through hyperparameter tuning. The independent variables are the same as shown in Table II and the dependent variable is either speed, logspeed or pace.

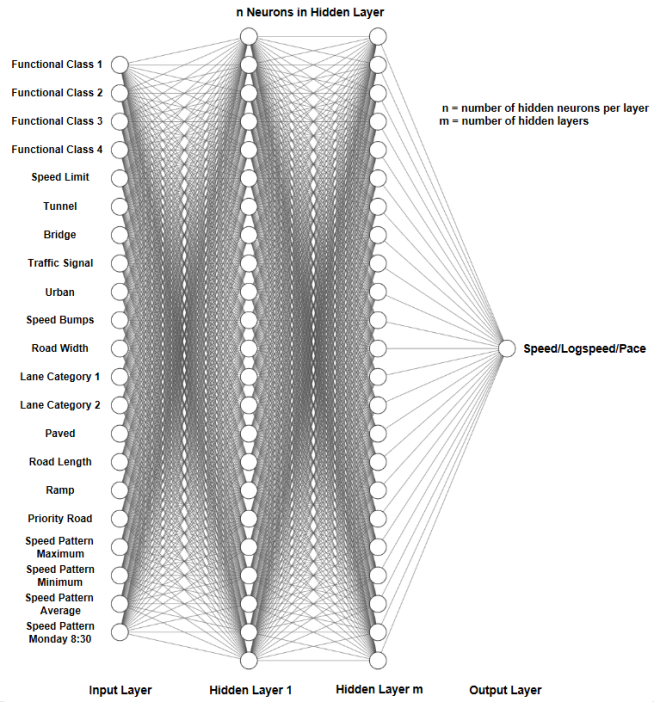

Fig. 3: Neural Network architecture for speed, logspeed and pace prediction.

D. Hyperparameter Tuning

Hyperparameters are parameters that have to be chosen by the user. In Table IV, the hyperparameters along with their values of the neural network models can be found. Hyperparameter tuning was done for the learning rate, mini-batch size, number of hidden layers and number of neurons. These are the most important hyperparameters that have generally the largest influence on the prediction accuracy[8][9][10]. Hyperparameter tuning was done by using 5-fold cross-validation. It was found that all models without dependent variable pace and $\text{sMdAPE}_{\text{LF}}$ had the best prediction accuracy for 5 hidden layers and 128 hidden neurons. For the models with dependent variable pace the best number of hidden layers is 4 and hidden neurons 32. For the models with $\text{sMdAPE}_{\text{LF}}$, the best number of hidden layers is 7 and hidden neurons 32.

TABLE IV: Overview neural network hyperparameters and values. All hyperparameter values, except the number of hidden layers and neurons, are the same for all 21 neural network models.

Hyperparameter	**Value**
Learning Rate	0.001
Mini-Batch Size	256
Early Stopping Epochs	10
Number of Hidden Layers	5, 4 or 7
Number of Neurons	128, 32 or 32
Optimizer	Adam
Hidden Layer Activation Function	ELU
Output Layer Activation Function	Linear
Maximum Epochs	200

V. Experiments

The experiments will be run with new data, which is collected and preprocessed during this research, and the old data, from den Heijer[3]. An overview of the new and old data is shown in Table V. The main difference between the two data sets is that the frequency of the new data is 2 minutes and for the old data 5 minutes. Since there are 21 different neural network models, as shown in Table III, for new and old data, 2*21=42 experiments have to be run.

TABLE V: Summarized description of new and old data that is used for training and predicting speeds and ultimately improving travel time predictions.

Data Properties	**New Data**	**Old Data**
Average Recorded Frequency	2 min	5 min
Minimum Evaluated Travel Time	2 min	10 min
Training Points	1,083,237	410,137
Test Points	362,141	139,687
Training Trips	785,920	165,479
Test Trips	263.275	54,349
Test Travels	45,852	10,316
Country	Netherlands	Benelux
Vehicle Type	Truck	Truck

The experiments that will be run consists of two steps: *Training & Evaluation* and *Prediction & Evaluation*. These steps are shown in Figure 4.

- **Step 1:** The model is first trained on either point- or trip-based data. After this, the speed prediction accuracy is calculated of the trained model. This is done by comparing the predicted speeds of the trained model, based on the input variables, with the actual speeds in the test data set of the point-based data. The speed prediction accuracy of each neural network model will be calculated with the same error function as the used loss function. The speed prediction accuracy will be expressed in $\text{MSE}_{\text{Speed}}$, $\text{MAPE}_{\text{Speed}}$, $\text{MAE}_{\text{Speed}}$, $\text{sMAPE}_{\text{Speed}}$ and $\text{sMdAPE}_{\text{Speed}}$.

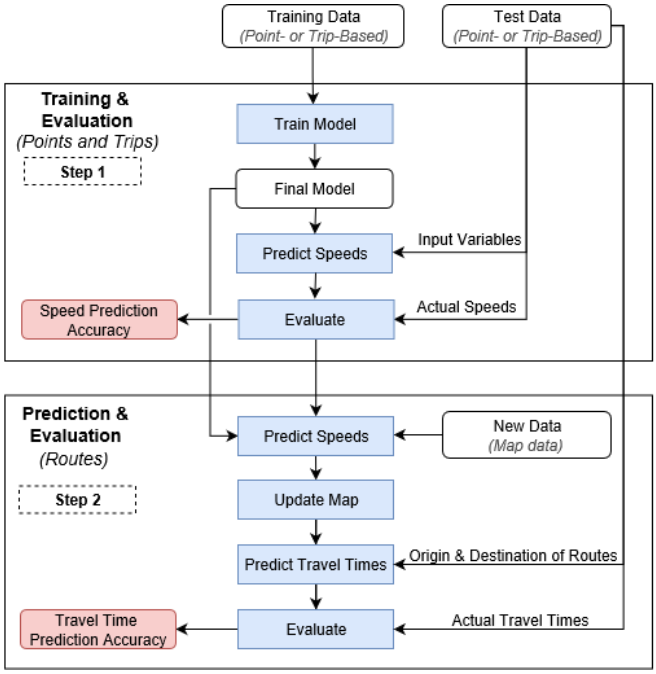

Fig. 4: The two last steps of the research, which contains the experiments. This includes the training of the models, calculation of the speed prediction accuracy, the prediction and update of the speeds in the map and finally the calculation of the travel time prediction accuracy.

- **Step 2:** After the model is trained and evaluated on the speeds, the trained model can be used to predict the speed of each road in the map, based on its road properties (map data). In total, 3,304,031 speeds have to be predicted for the Netherlands and 5,169,618 for the Benelux. After all speeds of the roads in the map have been predicted, the map from either the Netherlands (new data) or the Benelux (old data) will be updated. The map update is the process of replacing the speeds of the roads in the map by the new predicted speeds. Subsequently, the travel times of the travels in the test set can be predicted. This is done by the routing software of *ORTEC* based on an improved Dijkstra algorithm called Highway Node Routing[11]. It calculates the shortest path from the first to the last GPS point of the travel, from which the expected travel time is calculated. These predicted travel times are compared to the actual travel times in the test set. The travel time prediction accuracy will be expressed in $\text{sMdAPE}_{\text{TT}}$.

VI. Results

A. Results Speed Prediction Accuracy

After the training process of each neural network speed prediction model, the speed prediction accuracy is calculated. The speed prediction accuracy for old and new data can be found in Table VI and Table VII, respectively. It can be concluded that the neural network models with dependent variable speed and point-based data, have the highest speed prediction accuracy with respect to the models with the same loss function.

TABLE VI: Speed prediction accuracy for all 21 neural network models for old data.

Dependent Variable	Data Type	Speed Prediction Accuracy - Old data				
		$\text{MSE}_{\text{Speed}}$	$\text{MAPE}_{\text{Speed}}$	$\text{MAE}_{\text{Speed}}$	$\text{sMAPE}_{\text{Speed}}$	$\text{sMdAPE}_{\text{Speed}}$
Speed	points	255	-	9.88	32.5	7.61
Logspeed	points	330	-	10.0	32.6	8.12
Speed	trips	420	-	13.7	49.7	20.5
Logspeed	trips	480	-	13.6	48.9	21.1
Pace	trips	653	292	13.3	48.3	20.0

TABLE VII: Speed prediction accuracy for all 21 neural network models for new data.

Dependent Variable	Data Type	Speed Prediction Accuracy - New data				
		$\text{MSE}_{\text{Speed}}$	$\text{MAPE}_{\text{Speed}}$	$\text{MAE}_{\text{Speed}}$	$\text{sMAPE}_{\text{Speed}}$	$\text{sMdAPE}_{\text{Speed}}$
Speed	points	215	-	9.51	40.2	14.8
Logspeed	points	264	-	10.5	40.2	15.0
Speed	trips	391	-	14.7	43.7	18.0
Logspeed	trips	407	-	15.3	50.0	18.8
Pace	trips	669	331	16.6	49.7	21.7

B. Compensation Systematic Bias

From the research of den Heijer[3], it was concluded that the $\text{sMdAPE}_{\text{TT}}$ of each model improves after the sMdPE_{TT} was moved to 0%. If the sMdPE_{TT} is negative, then the travel time predictions are underestimated (too fast), while a positive sMdPE_{TT} means that the travel times are overestimated (too slow). This can be seen as a systematic bias where more than halve of the travel time predictions are under- or overestimated. Reasons for the systematic bias might be:

1) The shortest path, calculate by routing software from *ORTEC*, is not the same path as driven by the driver.
2) The prediction models optimize the speed of points and trips. This means that not the travel time, but instantaneous moments and short parts of the full travel are optimized.

The sMdPE_{TT} can be moved to 0% by multiplying the travel times or speeds assigned to each road in the map, by a factor x. This multiplication is shown in Equation 9, where P_t is the predicted travel time or speed before compensation by x. $P_t^{'}$ is the predicted travel time or speed after compensation by x. The equation of x is shown in Equation 10.

$$P_t^{'} = x \cdot P_t \tag{9}$$

$$x = \frac{200\% - \text{sMdPE}}{200\% + \text{sMdPE}} \tag{10}$$

C. Results Travel Time Prediction Accuracy

In Table VIII and Table IX, the speed prediction models with the highest travel time prediction accuracy (lowest $sMdAPE_{TT}$) are shown. These results are after compensation of the systematic bias with factor x. For convenience, an abbreviation is used for the models in the format 'type of training data'-'prediction method'-'model output'-'loss function'. For example, Trips-NN-Logspeed-MSE means that it is a neural network, trained on trip-based data with output logspeed and loss function MSE_{LF}. The best speed prediction model for old data is Points-RF-Logspeed and Points-NN-Logspeed-MSE with $sMdAPE_{TT}$ 13.8%. For new data this is Points-RF-Speed with $sMdAPE_{TT}$ 12.4%. This means that the best speed prediction model differs between the data sets and that the random forest models of den Heijer are not outperformed by the neural network models with respect to $sMdAPE_{TT}$. Furthermore, the neural networks trained on trip-based data, with a higher frequency of 2 min., instead of 5 min., does also not outperform the random forest speed prediction model that is trained on point-based data.

TABLE VIII: Overview of den Heijer's best models and the best neural network models for old and new data in $sMdAPE_{TT}$. The best model for old and new data is highlighted.

	Training Data	Den Heijer's Best Model in $sMdAPE_{TT}$ (%)	Best Neural Network in $sMdAPE_{TT}$ (%)
Old Data	Point-Based	**13.8**	**13.8**
	Trip-Based	15.4	14.5
New Data	Point-Based	**12.4**	13.0
	Trip-Based	-	12.9

TABLE IX: Overview of den Heijer's best models and the best neural network models for old and new data. The best model for old and new data is highlighted.

	Training Data	Den Heijer's Best Model	Best Neural Network
Old Data	Point-Based	**Points-RF-Logspeed**	**Points-NN-Logspeed-MSE**
	Trip-Based	Trips-LR-Logspeed	Trips-NN-Logspeed-MSE
New Data	Point-Based	**Points-RF-Speed**	Points-NN-Speed-MSE
	Trip-Based	-	Trips-NN-Pace-sMdAPE

D. Travel Time versus Speed Prediction Accuracy

Den Heijer[3] concluded that the travel time prediction accuracy in $sMdAPE_{TT}$ is not related to the speed prediction accuracy in $sMdAPE_{Speed}$. Because different loss functions were used, it could be researched whether MAE_{Speed}, MSE_{Speed} or $sMAPE_{Speed}$ is related to $sMdAPE_{TT}$. It was found that MSE_{Speed} is mostly related to $sMdAPE_{TT}$. This is shown in Figure 5 and Figure 6 for old and new data. For new data, the model with the lowest MSE_{Speed} has the lowest $sMdAPE_{TT}$, while for old data the models with the third and fourth lowest MSE_{Speed} has the lowest $sMdAPE_{TT}$. Therefore, minimizing MSE_{LF} does not directly mean for all data sets, that the lowest $sMdAPE_{TT}$ will be obtained. However, the MSE_{Speed} can be used as a rough indicator for the $sMdAPE_{TT}$.

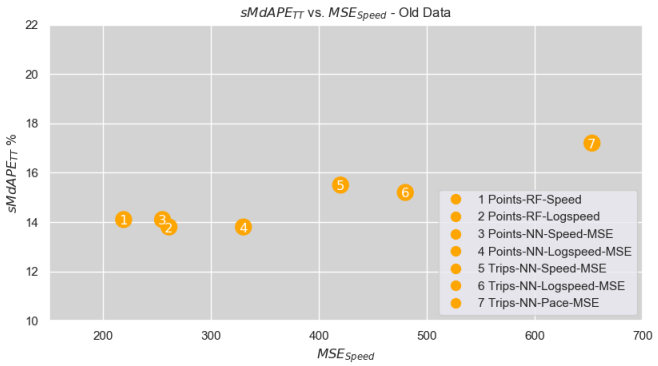

Fig. 5: Travel time prediction accuracy in $sMdAPE_{TT}$ versus speed prediction accuracy in MSE_{Speed} for old data. Only the models with loss function MSE_{LF} are included in the graph.

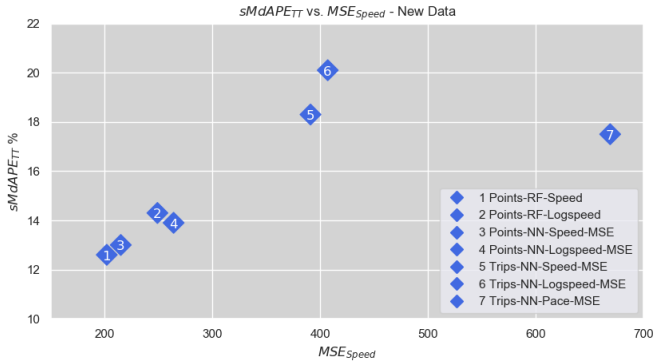

Fig. 6: Travel time prediction accuracy in $sMdAPE_{TT}$ versus speed prediction accuracy in MSE_{Speed} for new data. Only the models with loss function MSE_{LF} are included in the graph.

E. Influence Temporal Factor

The temporal factor could not be included in the neural network model as independent variable, since this would require the digital map to assign more speeds to one road for different moments in time. This is not possible with the routing software of *ORTEC*. Therefore, the temporal factor was researched by splitting the data set into rush (7:00-9:00 & 16:30-18:30) and non-rush hour data sets. Subsequently, two different neural network models were trained. The temporal factor was researched for the Points-RF-Speed, Points-NN-Speed-MSE and Trips-NN-Speed-MSE model. The improvement of the speed and travel time prediction accuracy can be found in Figure 7, after moving the $sMdPE_{TT}$ to 0%. The $sMdAPE_{TT}$ of the Points-NN-Speed-MSE and Trips-NN-Speed-MSE model improves with 0.6% and 3.0%, respectively. The $sMdAPE_{TT}$ of the Points-RF-Speed model does not improve, however the IQR sPE_{TT} improves from 28.0% to 27.3% for this model. This means that the errors are less widely spread.

Model	New Data		
	MSE_{Speed}	$sMdAPE_{TT}$ (%)	IQR sPE_{TT} (%)
Points-RF-Speed	202 → 193	12.4 → 12.4	28.0 → 27.3
Points-NN-Speed-MSE	215 → 195	13.0 → 12.4	28.6 → 28.4
Trips-NN-Speed-MSE	392 → 370	18.3 → 15.3	37.2 → 35.3

Fig. 7: Improvement of MSE_{Speed}, $sMdAPE_{TT}$ and IQR sPE_{TT}, when including rush and non-rush hours in the speed predictions.

VII. Conclusion & Recommendations

A. Conclusion

The main research question of this paper was: *Can a new speed prediction model be developed for trucks that outperforms the travel time prediction accuracy of the current speed prediction models for a given road network?* Based on the results that were obtained in this research, a new speed prediction model that outperforms the current speed prediction models, with respect to $sMdAPE_{TT}$, could not be developed. This is true for both the old and new data set. For both old and new data, one of the two random forest models, developed by den Heijer, had the best performance with $sMdAPE_{TT}$ 13.8% and 12.4%. However, it can be neither concluded that the neural network does not provide better travel time predictions for other data sets. Also, it cannot be concluded that neural networks, which are trained on trip-based data, will not outperform the random forest models with a higher data frequency than 2 minutes.

B. Further Research

Several recommendations for further research can be made based on this research. 1) From this research it is unclear what the influence is of a higher data frequency of 2 min. instead of 5 min. This is because the GPS data is obtained from two different customers. It would be interesting to research multiple data frequencies of the same data set to investigate whether a higher frequency improves the travel time prediction accuracy. 2) The neural network, trained on trip-based data, did not outperform the random forest models. Therefore, it is recommended to research data frequencies higher than 2 minutes, which increases the quality of the trip-based data and possibly the travel time prediction accuracy of the neural network speed prediction model. 3) By including more influential factors of the travel time, higher travel time prediction accuracies might be obtained. This includes weather factor and more temporal factors, but also road engineering factors that were not included in the available map data. 4) More research can be done, to develop a new method that is able to predict the speeds in such a way, that it is directly related to the travel time. In this way, the highest travel time prediction accuracy can be obtained. Due to the road network, which consists of many roads, and the lack of research for this subject in literature, this is a challenging problem.

References

[1] Glaeser, K. P., and Mr A. Ritzinger. "Comparison of the Performance of Heavy Vehicles Results of the OECD Study:Moving Freight with Better Trucks." *Procedia-Social and Behavioral Sciences* 48 (2012): 106-120.

[2] Wang, Fahui, and Yanqing Xu. "Estimating OD travel time matrix by Google Maps API: implementation, advantages, and implications." *Annals of GIS* 17.4 (2011): 199-209.

[3] B. den Heijer, "Improving Travel Time Predictions for Trucks with Historical GPS-Data", Masters thesis, University of Amsterdam, 2018.

[4] Bai, Mengting, et al. "Travel-Time Prediction Methods: A Review." International Conference on Smart Computing and Communication. Springer, Cham, 2018.

[5] Jensen, A. F., and T. V. Larsen. "Travel-time estimation in road networks using GPS data." Unpublished 8.8 (2014).

[6] Anderson, Jessica, and Michael Bell. "Travel time estimation in urban road networks." Proceedings of Conference on Intelligent Transportation Systems. IEEE, 1997.

[7] Asif, Muhammad Tayyab, et al. "Unsupervised learning based performance analysis of n-support vector regression for speed prediction of a large road network." 2012 15th International IEEE Conference on Intelligent Transportation Systems. IEEE, 2012.

[8] Greff, Klaus, et al. "LSTM: A search space odyssey." *IEEE transactions on neural networks and learning systems* 28.10 (2016): 2222-2232.

[9] Smith, Leslie N. "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay." *arXiv preprint arXiv:1803.09820* (2018).

[10] Klein, Aaron, et al. "Towards reproducible neural architecture and hyperparameter search." (2018).

[11] Schultes, Dominik, and Peter Sanders. "Dynamic highway-node routing." *International Workshop on Experimental and Efficient Algorithms*. Springer, Berlin, Heidelberg, 2007.

B

Road Engineering Factors used by den Heijer

Table B.1: Overview of road engineering factors that are used by den Heijer for linear regression and random forest models.

Road Engineering Factor	**HERE map data**
Road Category	*Functional Class, Speed Category*
Speed Limit	*Speed Limit*
Tunnel	*Tunnel*
Bridge	*Bridge*
Intersection	*Intersection Category*
Traffic Signal	*Traffic Signal, Traffic signal/km,*
Country	*The Netherlands, Belgium, Luxemburg*
Region	*(Non) Urban*
Speed Bumps	*Speed Bumps*
Horizontal Curve	*Sharp Turns*
Road Width	*Lane Category, WidthCm*
Road Surface Roughness	*Paved*
Road Length	*Road Length*
Ramp	*Ramp*
Priority road	*Priority Road*
Frontage road	*Frontage*
One way road	*One Direction*
Only four wheel driven vehicles	*Only 4WD*
Advisory Speed	*Advisory Speed*
Transition of amount of lanes	*Transition*
Narrowing of the road	*Narrowing*
Controlled Access Road	*Controlled Access Road*
Speed pattern	*Speed Pattern Max, Range, Average, Min, Mon830*

C

Description Machine Learning Methods

Machine Learning (ML) is a domain of artificial intelligence (AI) and can be used to learn from data by identifying patterns from large data sets to predict future outcomes and can be used for decision making. ML algorithms can be categorized into: supervised learning, unsupervised learning and reinforcement learning.

- **Supervised learning:** Supervised learning can be applied to problems where the data is labeled. The algorithm tries to find correlations between the input data and the outcome. The more data is used for the prediction model the more the system can learn which increases the accuracy of the outcome. Supervised learning algorithms can be divided into regression and classification problems. Regression is used if the output is a continuous variable, while classification is used for a categorical output.
- **Unsupervised learning:** Unsupervised learning is used when only the input data is known without a corresponding output. Unsupervised learning algorithms aim to find a underlying structure of the input data. Problems can be divided into clustering and association rule learning problems, where clustering tries to find groups to categorize the data and association rule learning tries to find relations between the variables in the data.
- **Reinforcement learning:** Reinforcement learning learns from previous decisions and adjust the strategy to improve the decision making. It differs from supervised learning in a way that the output is not known and the output is found by sequentially decisions.

In Figure C.1, an overview of common supervised and unsupervised ML algorithms is shown. In this research study, a supervised regression algorithm can only be used, since the output of the prediction model needs to be continuous which is the speed. The most supervised regression methods that can be used for this research problem are linear regression, support vector regression, gradient boosting method and neural network. All methods, except the neural network, are described below in more detail. A detailed description of the neural network can be found in section 3.3.

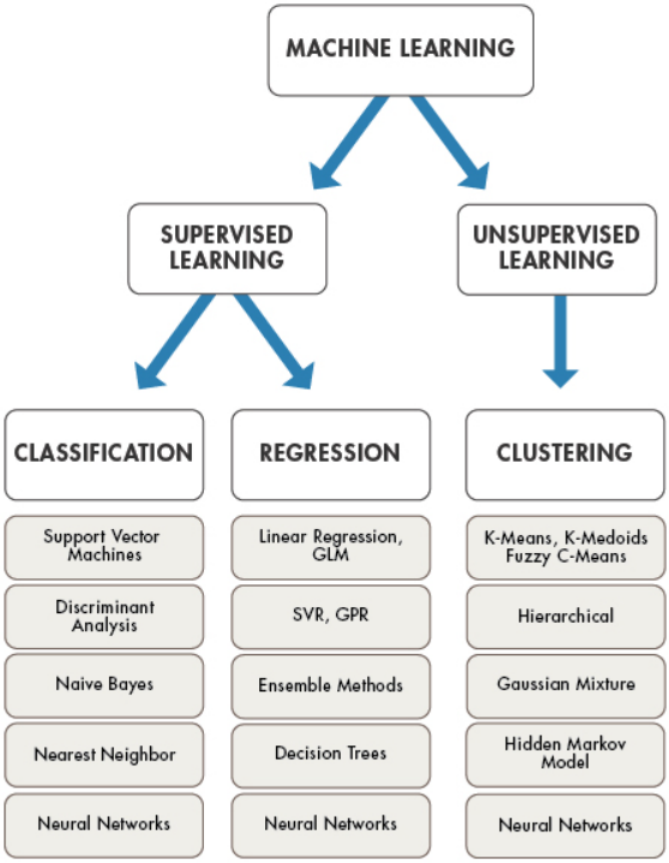

Figure C.1: Overview of the most popular supervised and unsupervised ML algorithms[114].

C.1. LINEAR REGRESSION

Linear regression is a simple type of regression and assumes linear relationships between the independent and dependent variables. In Equation C.1.1, a simple linear regression equation is shown where y is the dependent variable, x_1 the independent variable, β_0 a constant and β_1 a coefficient. When more independent variables affect the dependent variable y then a multi linear regression model can be used from which the equation is shown in Equation C.1.2. This equation has like the simple linear equation a constant β_0, but additional coefficients β and independent variables x.

$$y = \beta_0 + \beta_1 x_1 \tag{C.1.1}$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{C.1.2}$$

To find the best values for the coefficient β, a loss function is minimized which calculates the sum of squared errors also known as the residual sum of squares (RSS) and is shown in Equation C.1.3. The RSS function is minimized by using gradient descent which uses the partial derivatives of the RSS function with respect to the coefficients β to update the coefficients β.

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \tag{C.1.3}$$

However, when there are too many outliers, then the model might not be generalized well for the unseen/test data set resulting in an over-fitted model. This is illustrated in Figure C.2, where on the right image the model is over-fitted by fitting the model too much to the outliers increasing the model complexity. In Figure C.3, it

is illustrated that when the model complexity increases the error on the training set decreases, but on the test set increases. A good balance have to found to minimize the error on the set. The two most common regularization methods for linear regression are Ridge Regression and Lasso from which the formulas are shown in Equation C.1.4 and Equation C.1.5. Both regularization methods use the RSS function plus an additional term that penalizes the increase in coefficients β resulting in a decrease in complexity and over-fitting of the model. λ represents the tuning parameter which determines how much the complexity of the model is penalized. This is a hyperparameter where the optimal value can be found by applying a hyperparameter tuning method.

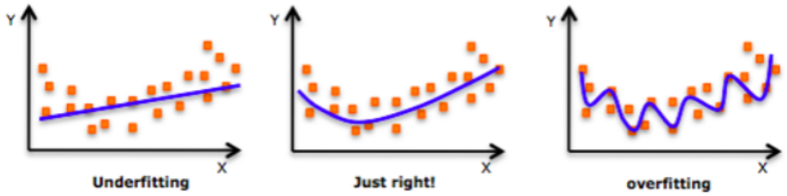

Figure C.2: Example of an under-fitted, fitted and over-fitted model[115].

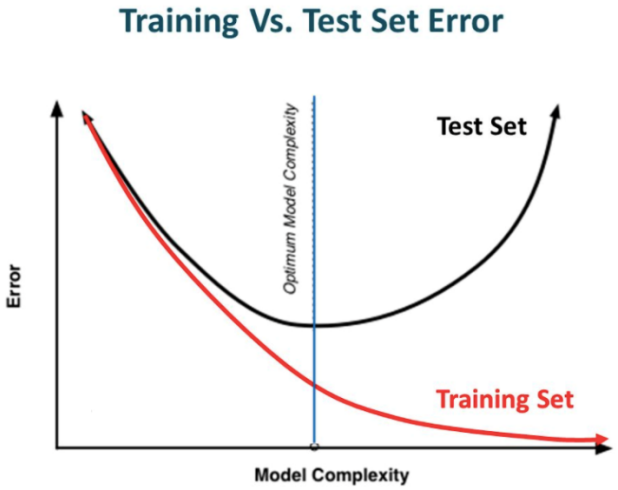

Figure C.3: Optimum model capacity where a further reduction of the training set error leads to over-fitting and an decrease in the model accuracy [115].

$$\text{Ridge Regression} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \tag{C.1.4}$$

$$\text{Lasso} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \tag{C.1.5}$$

C.2. SUPPORT VECTOR REGRESSION

Support Vector Regression is a regression method and is an implementation of Support Vector Machines (SVM) which is used for classification problems. SVR can be used for both linear and non-linear problems, where a kernel function is used to map the low dimensional data in a high-dimensional feature space. An example of an one dimensional linear SVR model is shown in Figure C.4. The middle line is the hyper plane and predicts the target value. The two dashed lines are the boundary lines and are constructed with an ε-deviation to create a margin. The SVR model tries to fit all observations within the two constructed boundary lines. However, when outliers are present, then this is not possible. The best fit can found by minimizing the error function shown in Equation C.2.1 taking into account the constraints shown below. Equation C.2.1 aims to minimize the generalization error, where the first term is used to minimize the complexity of the

model. The second term represents the error of the points that are outside the boundary lines. Where C is the regularization parameter, that determines how much the outliers are taken into account and ξ_i and ξ_i^* are the distances between observation i and the boundary lines.

$$\text{minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \tag{C.2.1}$$

Constraints:

$$\begin{aligned} &y_i - (\mathbf{w}\cdot\mathbf{x_i}) - b \leq \varepsilon + \xi_i && ,i = 1,...,m \\ &(\mathbf{w}\cdot\mathbf{x_i}) + b - y_i \leq \varepsilon + \xi_i^* && ,i = 1,...,m \\ &\xi_i, \xi_i^* \geq 0 && ,i = 1,...,m \end{aligned}$$

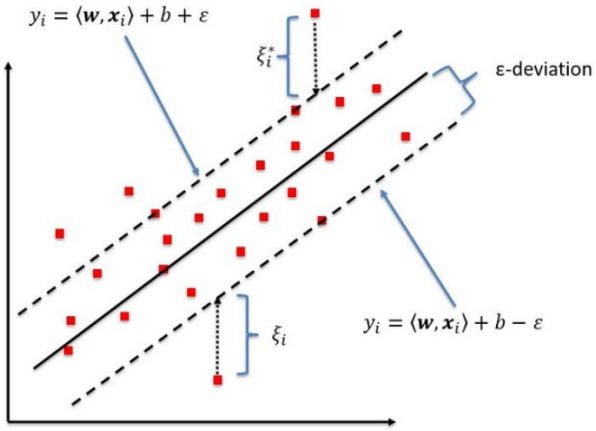

Figure C.4: Schematic of the one-dimensional support vector regression (SVR) model. only the points outside the dashed lines are used for making predictions[116].

C.3. RANDOM FOREST

Random forest is an ensemble learning method that merges multiple decision tree models to obtain a more stable and accurate model. The variance of one decision tree model can be quite high and depends considerably on the training set. By combining more decision trees, which are trained on different training sets, the average of all decision trees is taken and therefore the variance reduced. Despite one decision tree is easy to interpret, many decision trees together forming a random forest is hard to understand. Random forest can be used for both classification and regression where for regression the outcomes of all decision trees are averaged to predict the output y. A visualization of the random forest method is shown in Figure C.5. The mathematical representation of the random forest method is shown in Equation C.3.1, where $\hat{f}^b(\text{x})$ represents a decision tree b, $b = \{1,2,...,\text{B}\}$ and B the total number of decision trees. Random forest is known as a robust method for over-fitting where the error converges to a limit with increased number of decision trees[117].

The training data set for each decision tree is based on the bagging method or also called bootstrap aggregating. Bootstrap aggregating makes sure that the entire training data set is divided into random data subsets which improves the accuracy of the random forest model. Tunable features that can be changed to improve the model are:

1. Number or variables at each cut
2. Minimum size of the terminal nodes
3. Number of terminal nodes
4. Number of decision trees

The terminal nodes contain one of the outcomes of the decision tree and a cut splits the branch into 2 or more other branches. The size of the terminal nodes depends on how much of the training data is in the last node of the tree. If the size is too small, then the tree is over-fitted, but if the size is too large, then the tree is under-fitted. The number of decision trees should be large enough until a maximum accuracy is obtained. There is not a too large number of decision trees, since over-fitting of the model by the number of trees is not possible[117]. However, over-fitting due to the number of terminal nodes and the size of terminal nodes is possible.

$$y = \hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{b}(x) \tag{C.3.1}$$

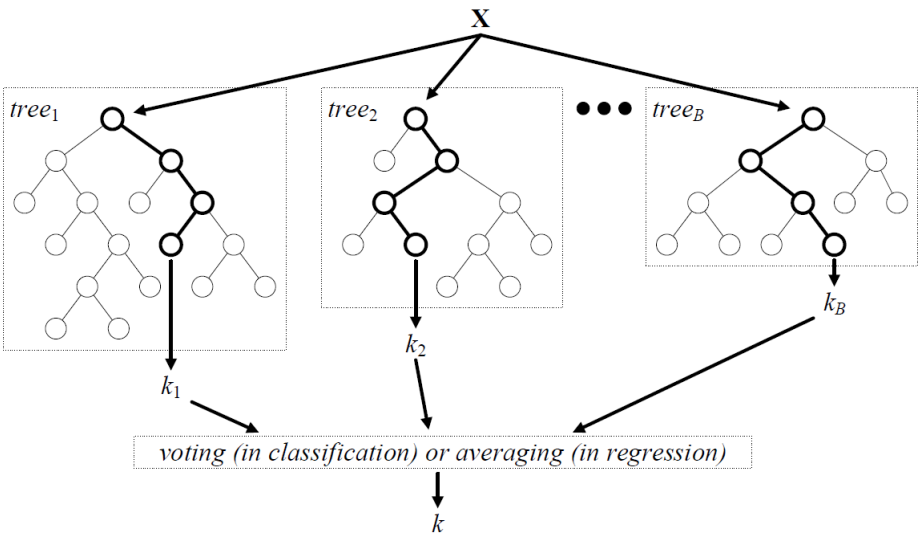

Figure C.5: Architecture of the random forest model[118].

C.4. BOOSTING METHODS

Boosting is also like random forest an ensemble technique which combines different weak learners to get a strong learner. The main difference between Boosting and random forest is that random forest creates decision trees in parallel and independently based on different training data subsets, while Boosting creates shallow decision trees sequentially and on the same training data set. A new decision tree that is added to the prediction model learns from the mistakes made by the previous tree(s) and uses this information to build a tree that increases the accuracy of the prediction model. The most popular Boosting methods are AdaBoost and Gradient Boosting and can both be used for both regression and classification problems

ADABOOST

AdaBoost, also called adaptive boosting, uses multiple decision trees with a single split as weak learners. The single split decision trees are built sequentially and learn by the incorrect classification of observations made by the previous trees. In Figure C.6, a simple example of the process of AdaBoost is shown. First, all pluses (+) and minuses (-) have an equal weight as shown in Box 1, after which the first decision tree is applied. As can be seen, 3 pluses are in the wrong region and therefore get a higher weight as shown in Box 2. Then the following decision tree takes the incorrect classification into account when making a new single split. This is repeated until a satisfactory classification is achieved as shown in Box 4.

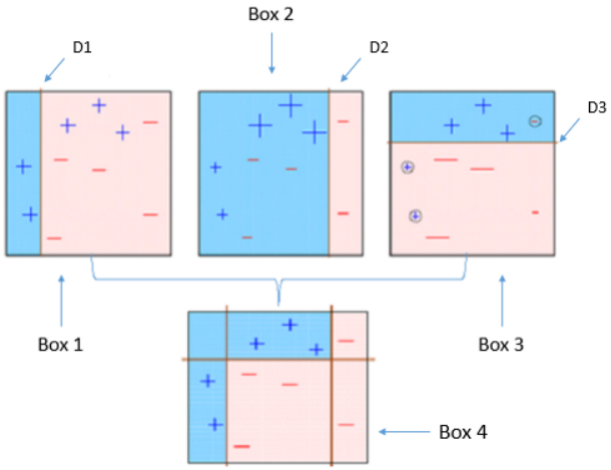

Figure C.6: Simple example of the process of AdaBoost[119].

GRADIENT BOOSTING

Gradient Boosting adds different shallow decision trees sequentially together to improve the accuracy of the model. Instead of changing the weights of each observation after a single split decision tree is applied like AdaBoost, Gradient Boosting builds a new predictor based on the residual errors of the previous build models. A visualization of this process is shown in Figure C.7, where each decision tree learns from the residual errors and improves the prediction model accuracy.

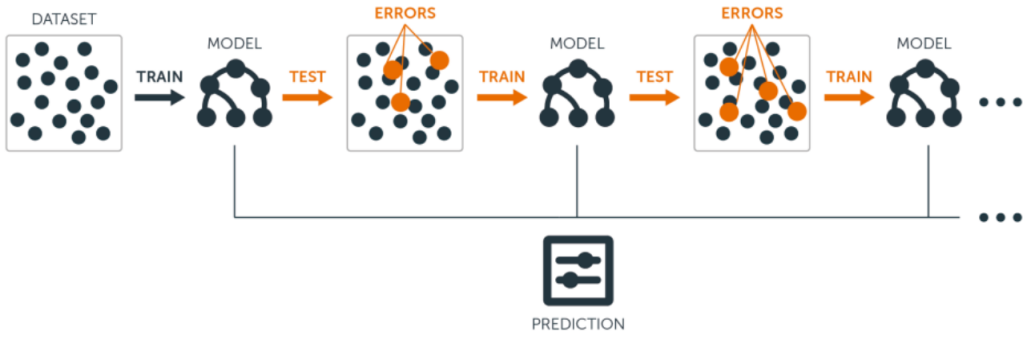

Figure C.7: Process of building a Gradient Boosting prediction model, where new learned decision trees are added to the prediction model[120].

The algorithm of Gradient Boosting is shown below[121]. Here $\hat{f}(x)$ is the prediction model, $\hat{f}^b(x)$ one decision tree, r_i the residual error between y_i and $\hat{f}(x_i)$, λ the shrinkage parameter, B the total number of decision trees and X the independent variables. First, the prediction model is set to zero and the residual errors set equal to y. Then a decision tree is fit to the training data (X,r), after which the prediction model is updated by adding a new decision tree. Then the residuals are updated by subtracting the output of the added decision tree, multiplied by λ, from the residuals. Step 2 is repeated until all B trees are added to the prediction model, after which the boosted model is built.

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For b = 1,2,...,B, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x) \tag{C.4.1}$$

(c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i) \tag{C.4.2}$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x) \tag{C.4.3}$$

The tunable features of the GBA that can be changed to improve the model are the following:

1. Number of decision trees B. It is possible to over-fit the model if B is too large, but goes slowly.
2. Value of the shrinkage parameter λ which is the learning rate. λ and B are correlated meaning that λ decreases when B increases and vice versa.
3. Number of cuts d, where too many cuts leads to over-fitting.

Most Gradient Boosting algorithms optimize there model by using the means squared error (MSE), which is the average between the predictions and the actual values[122]. Gradient boosting has a high flexibility due to the many parameters like number of iterations, tree depth, regularization parameters, etc. This requires tuning of these parameters to get the best performance and is explained in subsection 5.6.2.

A recent and famous implementation of Gradient Boosting is XGBoost, which stands for eXtrem Gradient Boosting[119]. This implementation is extremely popular at data science competition due to the outstanding speed and performance. This is due to the parallelization of tree construction, distribution of computations, out-of-core computation for data sets that do not fit in memory and optimization of data structures.

D

Sensitivity Analysis Method Trade-Off

To make sure that the outcomes of the method trade-offs in subsection 3.2.2 are robust, a sensitivity analysis was done. In total, four different cases were analyzed where the relative score of the neural network compared to the other models is disfavored by decreasing or increasing a weight or score by 1. The cases that were analyzed are:

- **Case 1:** Decreasing the *weight* of the criterion 'prediction accuracy' by 1, since the neural network scores here the best.
- **Case 2:** Increasing the *weight* of the criterion 'hyperparameter tuning' by 1, since the neural network score here the worst along with Gradient Boosting.
- **Case 3:** Increasing the *weight* of the criterion 'interpretability' by 1, since the neural network score here the worst.
- **Case 4:** Decreasing the *score* of the neural network for the criterion 'prediction accuracy' by 1, since the neural network scores here the best.

In Table D.1 the original method trade-off is shown where in Table D.2, Table D.3, Table D.4 and Table D.5 the method trade-offs with adjusted weight and scores are shown. From the adjusted trade-off tables, it can be concluded that in all cases the neural network still has the highest score.

Table D.1: Trade-off between multiple machine learning methods based on different criteria.

Criteria	**Weight**	**LR**	**RF**	**SVR**	**GB**	**NN**
Prediction Accuracy	5	1	3	4	4	5
Hyperparameter Tuning	2	5	4	2	2	1
Interpretability	1	5	3	2	3	1
Large Dataset	5	4	4	1	4	5
Total		40	46	31	47	**53**

Table D.2: Trade-off between multiple machine learning methods based on different criteria. The weight of the criterion prediction accuracy is decreased by 1 (Case 1).

Criteria	**Weight**	**LR**	**RF**	**SVR**	**GB**	**NN**
Prediction Accuracy	4 **(-1)**	1	3	4	4	5
Hyperparameter Tuning	2	5	4	2	2	2
Interpretability	1	5	3	2	3	1
Large Dataset	5	4	4	1	4	5
Total		39	43	29	43	**50**

Table D.3: Trade-off between multiple machine learning methods based on different criteria. The weight of the criterion hyperparameter tuning is increased by 1 (Case 2).

Criteria	**Weight**	**LR**	**RF**	**SVR**	**GB**	**NN**
Prediction Accuracy	5	1	3	4	4	5
Hyperparameter Tuning	3 **(+1)**	5	4	2	2	1
Interpretability	1	5	3	2	3	1
Large Dataset	5	4	4	1	4	5
Total		45	50	33	49	**54**

Table D.4: Trade-off between multiple machine learning methods based on different criteria. The weight of the criterion interpretability is increased by 1 (Case 3).

Criteria	**Weight**	**LR**	**RF**	**SVR**	**GB**	**NN**
Prediction Accuracy	5	1	3	4	4	5
Hyperparameter Tuning	2	5	4	2	2	1
Interpretability	2 **(+1)**	5	3	2	3	1
Large Dataset	5	4	4	1	4	5
Total		45	49	33	50	**54**

Table D.5: Trade-off between multiple machine learning methods based on different criteria. The score of neural network for criterion prediction accuracy is decreased by 1 (Case 4).

Criteria	**Weight**	**LR**	**RF**	**SVR**	**GB**	**NN**
Prediction Accuracy	5	1	3	4	4	4 **(-1)**
Hyperparameter Tuning	2	5	4	2	2	1
Interpretability	1	5	3	2	3	1
Large Dataset	5	4	4	1	4	5
Total		40	46	31	47	**48**

E

HYPERPARAMETER TUNING METHODS

To find the best setting for the hyperparameters, a search algorithm can be used. The most common search algorithms are grid search, random search and Bayesian optimization and are discussed below:

- **Grid search:** The simplest hyperparameter tuning method is grid search. This technique evaluates different combinations from a predefined list with values for each hyperparameter and returns the combination with the best performance. However, the computational effort of grid search is considerably high. For example, if there are 5 hyperparameters with each 5 different values to test, then there are 5^5 = 3125 combinations.
- **Random search:** Random search differs from grid search in a way that random combinations of hyperparameters are used instead of predefined hyperparameter combinations. The chance of finding the optimal value of hyperparameters with random search is higher, since it is not limited to a grid. In Figure E.1, a two-dimensional grid and random search, with an important and unimportant hyperparameter, for nine trials is shown. As can be seen, a change in value for the unimportant hyperparameter does not have a significant impact. A change in the important hyperparameter has a significant impact according to the green distribution. Because the grid search uses predefined values for the hyperparameters, the chance of finding an optimal value is less compared to random search. Generally, it also uses more computational time.

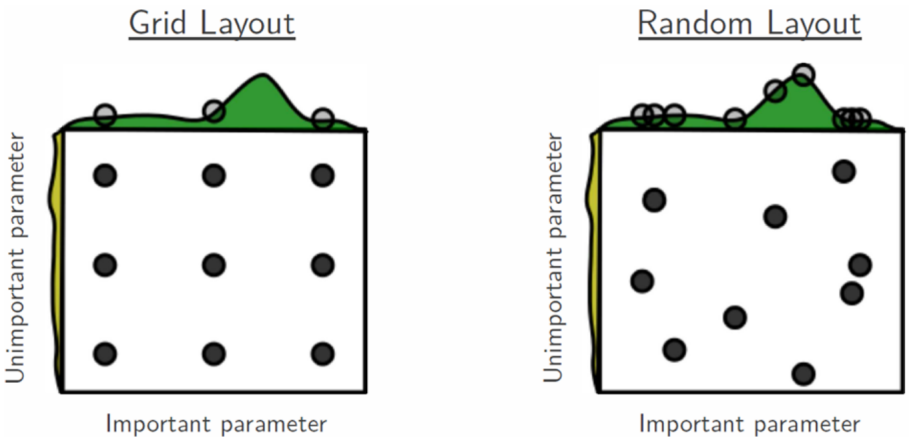

Figure E.1: Grid and random search of nine trials for optimizing a function[123].

- **Bayesian optimization:** The grid and random search are performed isolated and therefore do not learn from previous experiments to improve the next experiment. Unlike grid and random search, Bayesian optimization chooses the next hyperparameter values based on the previous experiment. In this way, less iterations are needed and an improved generalization performance on the test set is obtained. A

Gaussian process can be used to obtain a probability function of the previous scores with respect to the hyperparameter values. This function can be used to choose the next hyperparameter values that will highly likely improve the model performance. In Figure E.2, an illustration of three iterations of the Bayesian optimization procedure is shown. The green line shows the acquisition function. The green line is high where the model predicts a high objective and a high prediction uncertainty. The prediction uncertainty is illustrated by the purple area[124]. Both the acquisition function and the uncertainty are used to choose the following point to evaluate. In Figure E.2, the left remains unsampled due to little expected improvement compared to the right of the graph. For a more detailed explanation of the Bayesian optimization method, Shahriari et al. [124] can be consulted.

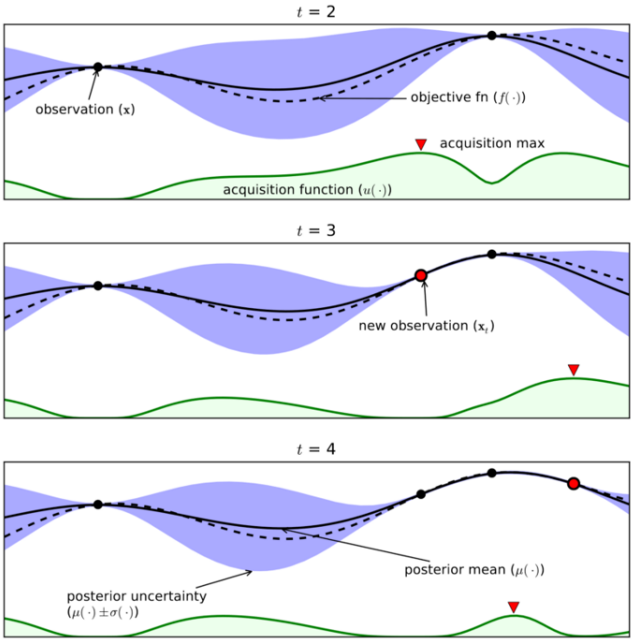

Figure E.2: Illustration of the Bayesian optimization procedure over three iterations[124]

F

Hyperparameter Tuning and Sensitivity Analysis - New Data

In Table F.1, the results of the sensitivity analysis, explained in subsection 5.6.3, are shown for all 21 neural network models trained on new data. From the results, it can be concluded that the models with dependent variable pace or $sMdAPE_{LF}$ do not have the same best hyperparameter setting as the Points-NN-MSE-Speed model. Therefore, an additional hyperparameter tuning process has to be done for these models. This will be done by repeating the process as done for the Points-NN-MSE-Speed model for one model with dependent variable pace (Points-NN-MSE-Speed model), and one model with $sMdAPE_{LF}$ (Points-NN-Speed-sMdAPE). After this, a sensitivity analysis will be done for the other models with dependent variable pace or $sMdAPE_{LF}$. The hyperparameters that will be tuned are the number of neurons and hidden layers. During the hyperparameter tuning process, the learning rate is fixed to 0.001 and the mini-batch size to 256. Whether a mini-batch size of 256 results in the highest model accuracy will be researched during the sensitivity analysis.

Hyperparameter Tuning and Sensitivity Analysis for Models with Dependent Variable Pace

To determine the right number of neurons and hidden layers for the models with dependent variable pace, the Trips-NN-Pace-MSE model is used. First, the number of neurons were determined, after which the best number of hidden layers were found. Table F.2 and Figure F.1 show that the best model accuracy, in MSE_{LF}, is obtained with 32 neurons. The results in Table F.3 and Figure F.2 show that the model accuracy in MSE_{LF} is further improved with 4 hidden layers.

A sensitivity analysis is done for all models with dependent variable pace. The combination of hyperparameters are shown in Table F.4. The results in Table F.5 show that all models, except the one trained with $sMdAPE_{LF}$, obtained the best model accuracy with the initial setting (32 neurons, 4 layers). Therefore the initial setting of hyperparameters will be used to for these models.

Hyperparameter Tuning for Models with $sMdAPE_{LF}$

To determine the right number of neurons and hidden layers for the models with loss function $sMdAPE_{LF}$, the Points-NN-Speed-sMdAPE model is used. First the number of neurons were determined and used to find the best number of hidden layers. Table F.6 and Figure F.3 show that the best model accuracy in $sMdAPE_{LF}$ is obtained with 32 neurons. The results in Table F.7 and Figure F.4 show that the model accuracy in MSE_{LF} is further improved with 7 hidden layers.

A sensitivity analysis is done for all models with loss function $sMdAPE_{LF}$. The combination of hyperparameters are shown in Table F.8. The results in Table F.9 show that all models obtain the best model accuracy with the initial setting (32 neurons, 7 layers). Therefore the initial setting of hyperparameters shown in Table F.8 will be used for these models.

Table F.1: Results of the sensitivity analysis for all 21 different neural network models for seven different combinations of hyperparameters and trained on new data with 5-fold CV. The best result of the seven hyperparameter combinations for each model is underlined.

	Point-Based Data - Speed				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	<u>216.6</u>	-	<u>9.368</u>	<u>0.4163</u>	0.1655
1	217.8	-	9.384	0.4313	0.1699
2	218.4	-	9.502	0.4522	<u>0.1602</u>
3	221.9	-	9,376	0.4355	0.1615
4	217.9	-	9.372	0.4283	0.1658
5	218.7	-	9.398	0.4317	0.1623
6	219.1	-	9.418	0.4411	0.1630
	Point-Based Data - Logspeed				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	<u>0.1535</u>	-	<u>0.2176</u>	<u>0.2581</u>	0.04529
1	0.1624	-	0.2196	0.2653	0.04621
2	0.1553	-	0.2185	0.2592	<u>0.04321</u>
3	0.1559	-	0.2211	0.2610	0.04541
4	0.1559	-	0.2199	0.2626	0.04419
5	0.1611	-	0.2201	0.2673	0.04501
6	0.1622	-	0.2206	0.2688	0.04489
	Trip-Based Data - Speed				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	<u>156.8</u>	-	<u>7.047</u>	<u>0.2562</u>	0.1307
1	158.3	-	7.152	0.2600	0.1354
2	162.8	-	7.092	0.2582	<u>0.1283</u>
3	161.7	-	7.182	0.2629	0.1376
4	158.8	-	7.120	0.2582	0.1292
5	159.2	-	7.108	0.2622	0.1364
6	158.2	-	7.121	0.2598	0.1382
	Trip-Based Data - Logspeed				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	<u>0.04002</u>	-	<u>0.1069</u>	<u>0.1101</u>	0.03828
1	0.04115	-	0.1177	0.1195	0.04012
2	0.04014	-	0.1101	0.1178	<u>0.03577</u>
3	0.04060	-	0.1089	0.1213	0.03921
4	0.04054	-	0.1112	0.1124	0.03627
5	0.04154	-	0.1181	0.1204	0.03821
6	0.04102	-	0.1150	0.1193	0.03701
	Trip-Based Data - Pace				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	0.007383	24.93	0.02510	0.2759	0.1532
1	0.007429	25.22	0.02556	0.2854	0.1594
2	<u>0.007062</u>	<u>22.45</u>	<u>0.02370</u>	<u>0.2578</u>	<u>0.1343</u>
3	0.007287	25.26	0.02409	0.2713	0.1547
4	0.007134	23.38	0.02481	0.2633	0.1432
5	0.007312	24.12	0.02477	0.2695	0.1495
6	0.007166	24.85	0.02434	0.2663	0.1506

Table F.2: Model accuracy in MSE_{LF} and 5-fold CV training time versus number of hidden neurons for the model Trips-NN-Speed-MSE. The number of hidden neurons ranges between 1 and 128, lr = 0.001, number of hidden layers = 2 and mini-batch size = 256. The best result is underlined.

Hidden Neurons	**Model Accuracy (MSE_{LF})**	**Time [seconds]**
1	0.007705	1750
2	0.007582	1540
4	0.007410	1632
8	0.007158	1580
16	0.007120	2340
32	<u>0.007063</u>	3132
64	0.007105	3853
128	0.007090	5126

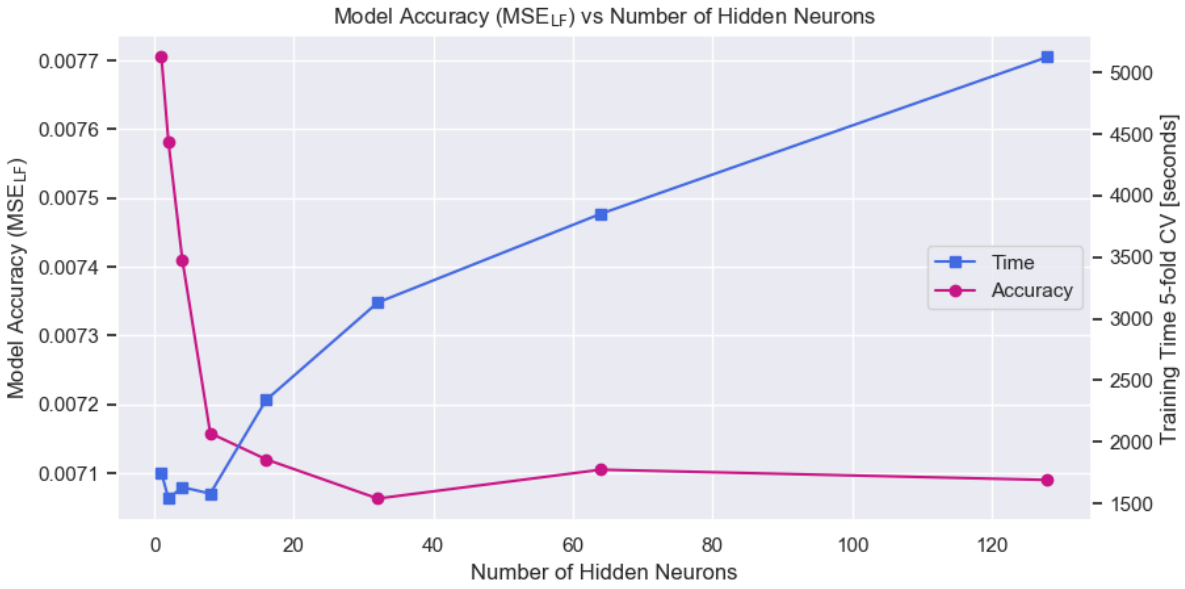

Figure F.1: Model accuracy in MSE_{LF} (purple circle) and 5-fold CV training time (blue square) versus number of hidden neurons for the model Trips-NN-Speed-MSE. The number of hidden neurons ranges between 1 and 128, lr = 0.001, number of hidden layers = 2 and mini-batch size = 256.

Table F.3: Model accuracy in MSE_{LF} and 5-fold CV training time versus number of hidden layers for the model Trips-MSE-Speed. The number of hidden layers ranges between 1 to 8, lr = 0.001, number of neurons = 32 and mini-batch size = 256. The best result is underlined.

Hidden Layers	**Model Accuracy (MSE_{LF})**	**Time [seconds]**
1	0.007207	2510
2	0.007063	3132
3	0.007042	3343
4	<u>0.007011</u>	3512
5	0.007092	3020
6	0.007143	2980
8	0.007284	2612

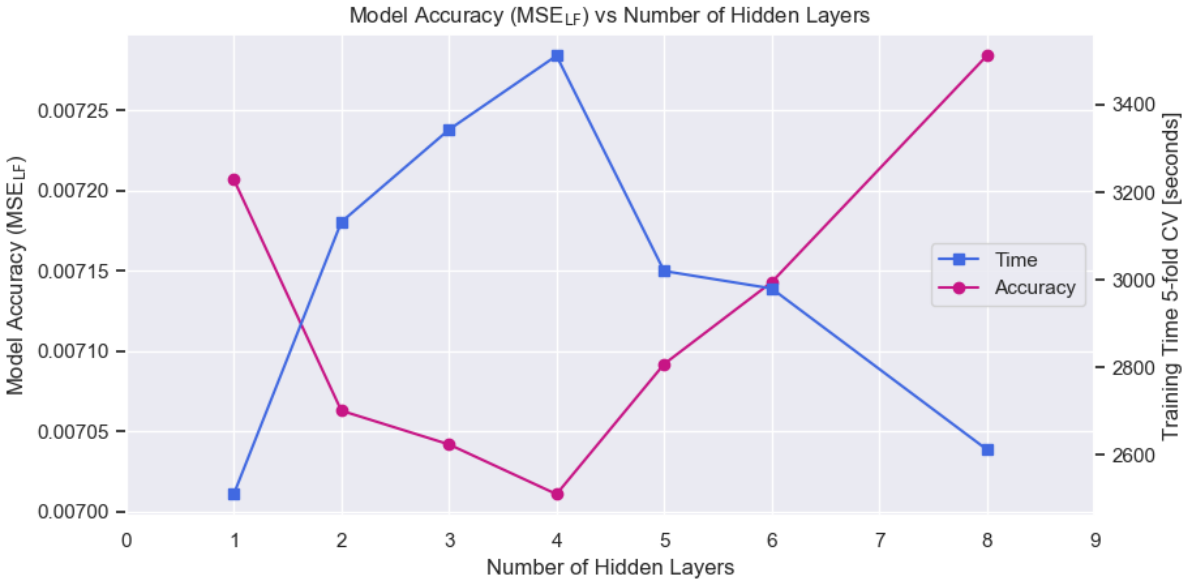

Figure F.2: Model accuracy in MSE$_{LF}$ (purple circle) and 5-fold CV training time (blue square) versus number of hidden layers for the model Trips-NN-Speed-MSE. The number of hidden layers ranges between 1 to 8, lr = 0.001, number of neurons = 32 and mini-batch size = 256.

Table F.4: Seven different hyperparameter combinations to research whether the initial setting for models with dependent variable pace produces the best model accuracy.

Combination	**Learning Rate**	**Hidden Neurons**	**Hidden Layers**	**Mini-Batch Size**
Initial Setting	0.001	32	4	256
1	0.001	64	4	256
2	0.001	16	4	256
3	0.001	32	5	256
4	0.001	32	3	256
5	0.001	32	4	128
6	0.001	32	4	512

Table F.5: Results of sensitivity analysis of 5 different neural network models with dependent variable pace. The results are for seven different combinations based on the number of hidden neurons, number of hidden layers and mini-batch size with 5-fold CV.

	Trip-Based Data - Pace				
Combination	**MSE**	**MAPE**	**MAE**	**sMAPE**	**sMdAPE**
Initial Setting	0.006922	21.77	0.02346	0.2403	0.1338
1	0.007126	22.35	0.02412	0.2554	0.1295
2	0.006996	22.20	0.02384	0.2476	0.1336
3	0.007159	21.85	0.02415	0.2493	0.1268
4	0.007288	21.97	0.02447	0.2499	0.1319
5	0.007125	22.31	0.02415	0.2512	0.1380
6	0.007168	22.94	0.02453	0.2572	0.1349

Table F.6: Model accuracy in MSE_{LF} and 5-fold CV training time versus number of hidden neurons for the model Points-NN-Speed-sMdAPE. The number of hidden neurons ranges between 1 and 128, lr = 0.001, number of hidden layers = 2 and mini-batch size = 256. The best result is underlined.

Hidden Neurons	**Model Accuracy (MSE_{LF})**	**Time [seconds]**
1	0.1942	2483
2	0.1762	2822
4	0.1692	3162
8	0.1672	2905
16	0.1598	3282
32	<u>0.1557</u>	3460
64	0.1582	3622
128	0.1572	3919

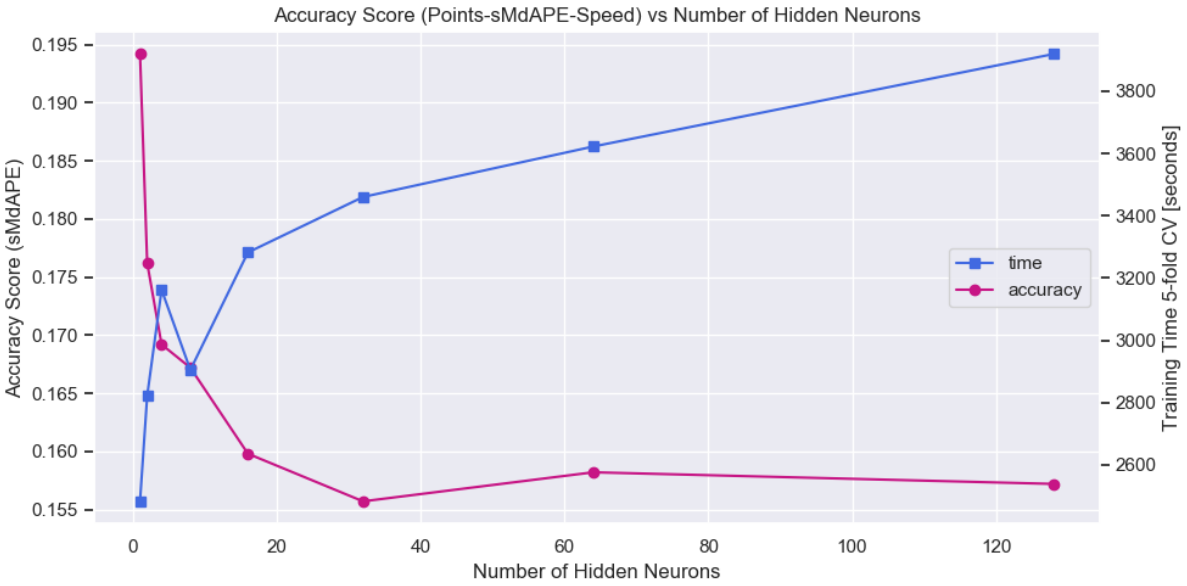

Figure F.3: Model accuracy in MSE_{LF} (purple circle) and 5-fold CV training time (blue square) versus number of hidden neurons for the model Points-NN-Speed-sMdAPE. The number of hidden neurons ranges between 1 and 128, lr = 0.001, number of hidden layers = 2 and mini-batch size = 256.

Table F.7: Model accuracy in MSE_{LF} and 5-fold CV training time versus number of hidden layers for the model Points-NN-Speed-sMdAPE. Number of hidden layers ranges between 1 to 10, lr = 0.001, number of neurons = 32 and mini-batch size = 256. The best result is underlined.

Hidden Layers	**Model Accuracy (MSE_{LF})**	**Time [seconds]**
1	0.1581	2510
2	0.1557	3132
3	0.1510	3343
4	0.1526	3512
5	0.1512	3020
6	0.1489	2980
7	<u>0.1472</u>	2612
8	0.1491	3741
10	0.1539	3817

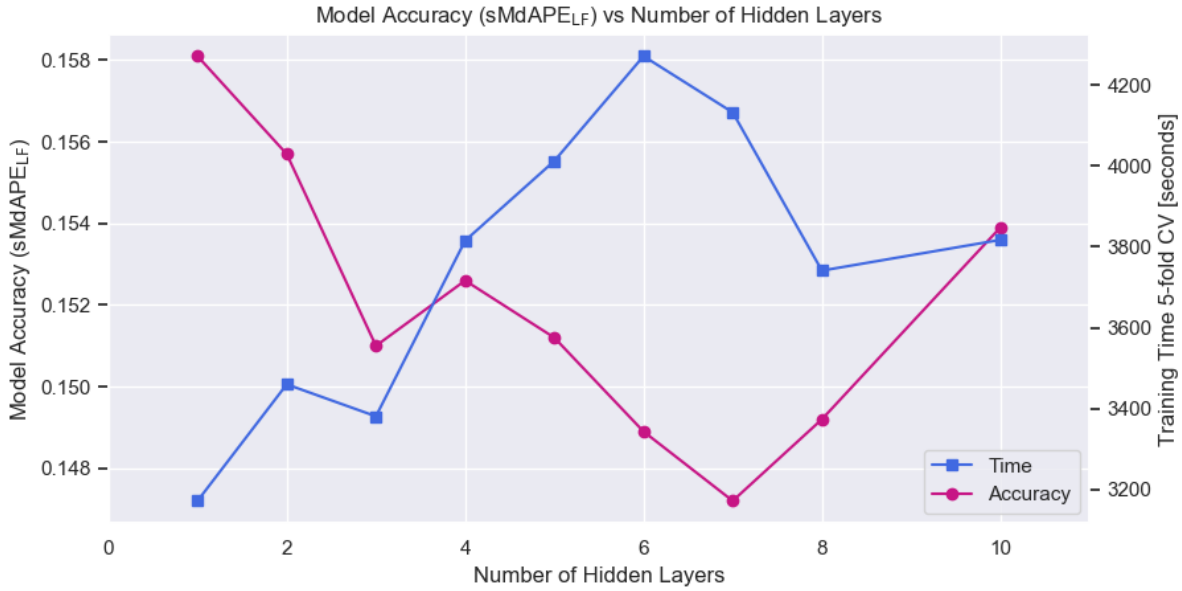

Figure F.4: Model accuracy in MSE$_{LF}$ (purple circle) and 5-fold CV training time (blue square) versus number of hidden layers for the model Points-NN-Speed-sMdAPE. Number of hidden layers ranges between 1 to 10, lr = 0.001, number of neurons = 32 and mini-batch size = 256.

Table F.8: Seven different hyperparameter combinations to research, whether the initial setting for models with loss function sMdAPE$_{LF}$ produces the best model accuracy.

Combination	**Learning Rate**	**Hidden Neurons**	**Hidden Layers**	**Mini-Batch Size**
Initial Setting	0.001	32	7	256
1	0.001	64	7	256
2	0.001	16	7	256
3	0.001	32	8	256
4	0.001	32	6	256
5	0.001	32	7	128
6	0.001	32	7	512

Table F.9: Result sensitivity analysis for 5 different neural network models with sMdAPE$_{LF}$ using 5-fold CV.

	Point-Based Data		**Trip-Based Data**		
Combination	**Speed**	**LogSpeed**	**Speed**	**LogSpeed**	**Pace**
Initial Setting	0.1557	0.0422	0.1297	0.03424	0.1253
1	0.1701	0.0461	0.1316	0.03532	0.1280
2	0.1613	0.0432	0.1352	0.03601	0.1271
3	0.1572	0.0451	0.1307	0.03592	0.1293
4	0.1608	0.0437	0.1299	0.03499	0.1301
5	0.1643	0.0471	0.1327	0.03588	0.1283
6	0.1686	0.0498	0.1281	0.03517	0.1334

G

SENSITIVITY ANALYSIS - OLD DATA

In this appendix, the sensitivity analysis for the models of the old data set will be discussed. For the new data set, it was found that the models with either dependent variable pace or $sMdAPE_{LF}$, require a different set of hyperparameters than the other neural network models (Appendix F). Because the old data has the same format and range of values for the independent and dependent variables, it is expected that the best hyperparameters of the new data, is the same for the old data. Therefore, the best hyperparameters for the models of the new data will be used as initial setting for the sensitivity analysis. The initial settings and combinations that will be used for the sensitivity analysis of the old data are shown in Table G.1, Table G.2 and Table G.3.

The results of the sensitivity analysis are shown in Table G.4 and show that the initial setting of hyperparameters is the best for all models. Therefore, the initial settings that were found for the models of the new data will also be used for the models of the old data.

Table G.1: Seven different hyperparameter combinations to research whether the initial setting for all models, without dependent variable pace and $sMdAPE_{LF}$, is the best for the old data. The initial setting is obtained from the models of the new data set.

Combination	Learning Rate	Hidden Neurons	Hidden Layers	Mini-Batch Size
Initial Setting	0.001	128	5	256
1	0.001	256	5	256
2	0.001	64	5	256
3	0.001	128	6	256
4	0.001	128	4	256
5	0.001	128	5	128
6	0.001	128	5	512

Table G.2: Seven different hyperparameter combinations to research whether the initial setting for all models with dependent variable pace, is the best for the old data. The initial setting is obtained from the models of the new data set.

Combination	Learning Rate	Hidden Neurons	Hidden Layers	Mini-Batch Size
Initial Setting	0.001	32	4	256
1	0.001	64	4	256
2	0.001	16	4	256
3	0.001	32	5	256
4	0.001	32	3	256
5	0.001	32	4	128
6	0.001	32	4	512

Table G.3: Seven different hyperparameter combinations to research whether the initial setting for all models with $sMdAPE_{LF}$, is the best for the old data. The initial setting is obtained from the models of the new data set.

Combination	Learning Rate	Hidden Neurons	Hidden Layers	Mini-Batch Size
Initial Setting	0.001	32	7	256
1	0.001	64	7	256
2	0.001	16	7	256
3	0.001	32	8	256
4	0.001	32	6	256
5	0.001	32	7	128
6	0.001	32	7	512

Table G.4: Results of sensitivity analysis for all 21 different neural network models trained on old data with 5-fold cross-validation. The best result of the seven hyperparameter combinations for each model is underlined.

	Point-Based Data - Speed				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	<u>252.6</u>	-	<u>9.785</u>	<u>0.3312</u>	<u>0.1278</u>
1	253.5	-	9.802	0.3424	0.1299
2	252.9	-	9.811	0.3381	0.1388
3	254.1	-	9.828	0.3359	0.1302
4	253.9	-	9.805	0.3432	0.1396
5	253.1	-	9.837	0.3372	0.1322
6	254.9	-	9.814	0.3402	0.1425
	Point-Based Data - Logspeed				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	<u>0.1336</u>	-	<u>0.2041</u>	<u>0.1906</u>	<u>0.03702</u>
1	0.1452	-	0.2156	0.2014	0.03723
2	0.1395	-	0.2106	0.1962	0.03832
3	0.1446	-	0.2154	0.1997	0.03928
4	0.1388	-	0.2088	0.2029	0.03841
5	0.1475	-	0.2123	0.2019	0.03902
6	0.1361	-	0.2077	0.2025	0.03825
	Trip-Based Data - Speed				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	<u>108.8</u>	-	<u>6.867</u>	<u>0.1359</u>	<u>0.08622</u>
1	109.9	-	6.882	0.1388	0.08691
2	110.6	-	6.962	0.1489	0.08742
3	111.2	-	6.901	0.1501	0.08750
4	110.9	-	6.976	0.1490	0.08838
5	110.5	-	6.927	0.1523	0.08775
6	111.7	-	6.982	0.1484	0.08828
	Trip-Based Data - Logspeed				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	<u>0.02371</u>	-	<u>0.07637</u>	<u>0.05655</u>	<u>0.1032</u>
1	0.02394	-	0.07701	0.05801	0.1173
2	0.02491	-	0.07783	0.05745	0.1095
3	0.02404	-	0.07691	0.05772	0.1126
4	0.02459	-	0.07699	0.05791	0.1192
5	0.02515	-	0.07791	0.05871	0.1125
6	0.02439	-	0.07799	0.05841	0.1172
	Trip-Based Data - Pace				
Combination	**MSE_{LF}**	**$MAPE_{LF}$**	**MAE_{LF}**	**$sMAPE_{LF}$**	**$sMdAPE_{LF}$**
Initial Setting	<u>0.004249</u>	<u>14.86</u>	<u>0.01352</u>	<u>0.01716</u>	<u>0.7997</u>
1	0.004382	15.14	0.01386	0.01792	0.8032
2	0.004281	16.43	0.01512	0.01850	0.8152
3	0.004414	15.95	0.01429	0.01812	0.8201
4	0.004324	16.01	0.01424	0.01777	0.8090
5	0.004385	15.26	0.01472	0.01884	0.8217
6	0.004302	16.77	0.01418	0.01759	0.8089

H

Additional Results Speed and Travel Time Prediction Accuracy

In Table H.1, the travel time prediction accuracies before compensation of the systematic bias $sMdPE_{TT}$ are shown. The best value in each column, and the values that have a maximum absolute difference of 1% with this best value, are highlighted in purple. The results are expressed in $sMdAPE_{TT}$, $sMdPE_{TT}$ and IQR sPE_{TT}, where $sMdAPE_{TT}$ is the main indicator of the travel time prediction accuracy. The best travel time prediction model for the new data is Points-RF-Speed with $sMdAPE_{TT}$ 12.4%. For the old data, the Trips-NN-Logspeed-MSE model performs the best with $sMdAPE_{TT}$ 14.5%.

Table H.1: Travel time prediction accuracy of all trained neural network models and two benchmarks. The models are evaluated on the travels in the test set where a lower $sMdAPE_{TT}$ means a better accuracy.

	New Data			Old Data		
Model	**$sMdAPE_{TT}$ (%)**	**$sMdPE_{TT}$ (%)**	**IQR sPE_{TT} (%)**	**$sMdAPE_{TT}$ (%)**	**$sMdPE_{TT}$ (%)**	**IQR sPE_{TT} (%)**
Benchmark Points-RF-Speed	**12.4**	**-1.0**	**28.0**	18.7	-17.7	30.4
Benchmark Points-RF-Logspeed	28.9	26.0	**28.1**	**14.8**	4.2	**28.5**
Points-NN-Speed-MSE	**13.0**	**-0.1**	28.6	18.7	-18.3	30.0
Points-NN-Speed-MAE	16.2	-13.7	31.8	18.6	-16.5	29.9
Points-NN-Speed-sMAPE	16.7	-14.4	32.7	30.3	-30.2	32.3
Points-NN-Speed-sMdAPE	23.5	22.7	38.4	28.9	7.5	37.7
Points-NN-Logspeed-MSE	28.1	25.1	**27.3**	**14.6**	**3.1**	**28.8**
Points-NN-Logspeed-MAE	17.6	7.5	32.0	18.9	-17.5	30.1
Points-NN-Logspeed-sMAPE	16.3	-13.5	32.7	27.2	-26.7	31.6
Points-NN-Logspeed-sMdAPE	23.0	-22.2	39.4	37.7	-37.7	37.0
Trips-NN-Speed-MSE	20.4	8.0	37.2	15.7	-12.3	33.0
Trips-NN-Speed-MAE	34.0	29.6	41.6	17.0	-14.4	31.6
Trips-NN-Speed-sMAPE	22.9	16.7	33.4	18.1	-16.5	31.7
Trips-NN-Speed-sMdAPE	17.3	-7.5	35.2	28.1	-27.4	36.4
Trips-NN-Logspeed-MSE	44.0	41.6	37.2	**14.5**	**-3.0**	32.0
Trips-NN-Logspeed-MAE	37.2	33.9	38.8	17.1	-15.0	32.3
Trips-NN-Logspeed-sMAPE	19.3	13.1	28.8	17.0	-15.6	31.4
Trips-NN-Logspeed-sMdAPE	13.6	-9.3	29.0	27.5	-27.2	34.3
Trips-NN-Pace-MSE	26.7	24.2	36.3	19.9	9.5	34.6
Trips-NN-Pace-MAPE	23.3	-22.6	34.4	31.0	-30.9	33.2
Trips-NN-Pace-MAE	30.1	26.5	32.4	16.7	-14.7	31.4
Trips-NN-Pace-sMAPE	30.8	27.2	35.6	19.4	-18.6	32.1
Trips-NN-Pace-sMdAPE	13.9	-9.6	29.2	29.6	-29.4	34.2

In the research of den Heijer[1], it was found that moving the $sMdPE_{TT}$ to 0.0%, the $sMdAPE_{TT}$ of the majority of the models are improved. This can also be seen as compensating for the systematic bias and is explained in subsection 6.3.1. In Table H.2, the travel time prediction accuracy after compensation of the systematic bias $sMdPE_{TT}$ is shown. The best value in each column, and the values that have a maximum absolute difference of 1% with this best value, are highlighted in purple. The best travel time prediction model for the new data is Points-RF-Speed with $sMdAPE_{TT}$ 12.6%. For the old data, the Points-RF-Logspeed is the best model with

$sMdAPE_{TT}$ 13.8%. The Points-NN-Logspeed-MSE also has $sMdAPE_{TT}$ 13.8% for the old data, however the errors are more spread than the Points-RF-Logspeed model with IQR sPE_{TT} 28.8%, compared to 28.5%.

Table H.2: Travel time prediction accuracy of all compensated neural network models and two benchmarks where the sMdPE is equal to 0. The models are evaluated on the travels in the test set where a lower $sMdAPE_{TT}$ means a better accuracy.

	New Data			Old Data		
Model	**$sMdAPE_{TT}$ (%)**	**$sMdPE_{TT}$ (%)**	**IQR sPE_{TT} (%)**	**$sMdAPE_{TT}$ (%)**	**$sMdPE_{TT}$ (%)**	**IQR sPE_{TT} (%)**
Benchmark Points-RF-Speed	**12.6**	0.0	**28.0**	**14.1**	0.0	30.8
Benchmark Points-RF-Logspeed	14.3	0.0	**28.5**	**13.8**	0.0	**28.5**
Points-NN-Speed-MSE	**13.0**	0.0	**28.6**	**14.1**	0.0	30.4
Points-NN-Speed-MAE	14.5	0.0	29.1	**14.2**	0.0	30.2
Points-NN-Speed-sMAPE	14.7	0.0	33.0	15.3	0.0	33.2
Points-NN-Speed-sMdAPE	17.6	0.0	39.1	18.7	0.0	37.8
Points-NN-Logspeed-MSE	13.9	0.0	**27.8**	**13.8**	0.0	**28.8**
Points-NN-Logspeed-MAE	16.0	0.0	32.0	**14.3**	0.0	30.4
Points-NN-Logspeed-sMAPE	14.5	0.0	33.0	14.9	0.0	32.3
Points-NN-Logspeed-sMdAPE	18.2	0.0	40.1	18.0	0.0	38.6
Trips-NN-Speed-MSE	18.3	0.0	37.2	15.5	0.0	33.2
Trips-NN-Speed-MAE	21.2	0.0	42.5	15.2	0.0	31.8
Trips-NN-Speed-sMAPE	16.8	0.0	33.6	15.0	0.0	32.0
Trips-NN-Speed-sMdAPE	17.0	0.0	35.3	17.3	0.0	37.3
Trips-NN-Logspeed-MSE	20.1	0.0	36.4	15.2	0.0	32.0
Trips-NN-Logspeed-MAE	19.8	0.0	39.8	15.4	0.0	32.5
Trips-NN-Logspeed-sMAPE	14.3	0.0	28.9	**14.8**	0.0	31.7
Trips-NN-Logspeed-sMdAPE	**13.2**	0.0	29.2	16.3	0.0	35.1
Trips-NN-Pace-MSE	17.5	0.0	35.6	17.2	0.0	34.7
Trips-NN-Pace-MAPE	15.9	0.0	35.0	16.1	0.0	34.2
Trips-NN-Pace-MAE	16.4	0.0	32.9	**14.7**	0.0	31.6
Trips-NN-Pace-sMAPE	18.1	0.0	36.2	**14.8**	0.0	32.5
Trips-NN-Pace-sMdAPE	**12.9**	0.0	29.0	16.3	0.0	35.1

In Figure H.1, Figure H.2 and Figure H.3, additional results of the speed versus travel time prediction accuracy can be found. The relationships between MAE_{Speed} and $sMAPE_{Speed}$ with respect to $sMdAPE_{TT}$ seem to be quite random. A lower MAE_{Speed} or $sMAPE_{Speed}$ does not necessarily result in a lower $sMdAPE_{TT}$. Therefore, these speed prediction accuracies should not be used as indicator for the travel time prediction accuracy. The relationships between $sMdAPE_{Speed}$ and $sMdAPE_{TT}$ in Figure H.3 confirm that these are weakly correlated, as concluded by den Heijer[1]. The results are contradictory, since the $sMdAPE_{TT}$ generally improves with a worse $sMdAPE_{Speed}$.

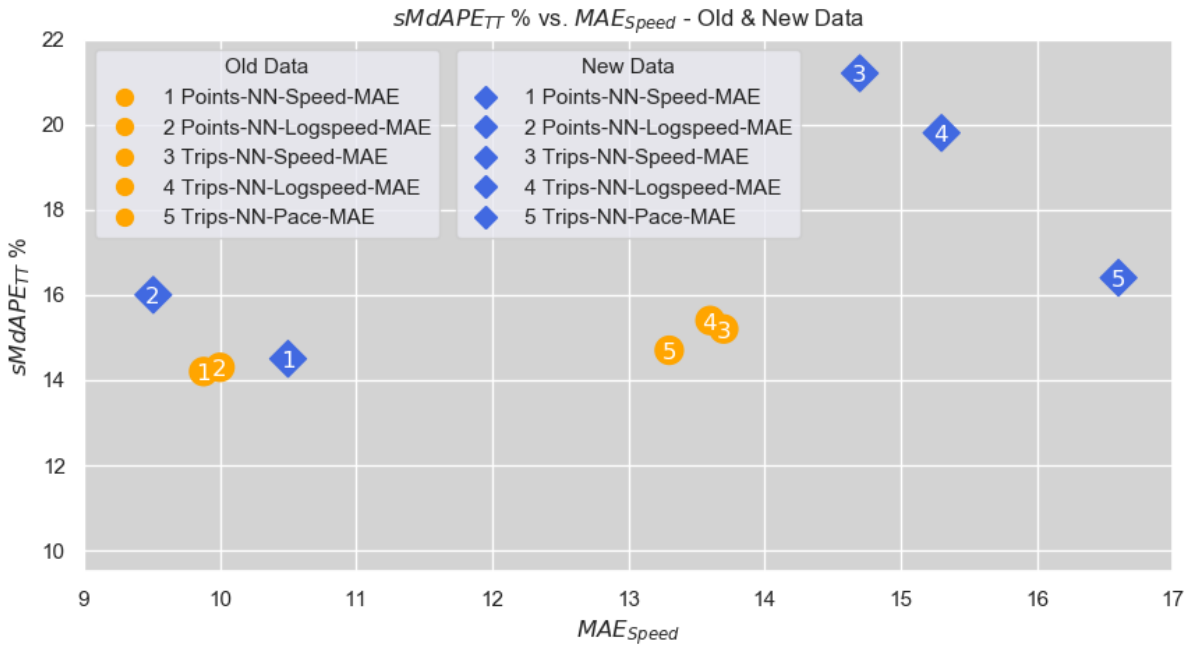

Figure H.1: Travel time prediction accuracy in $\text{sMdAPE}_{\text{TT}}$ versus speed predictions in $\text{MAE}_{\text{Speed}}$.

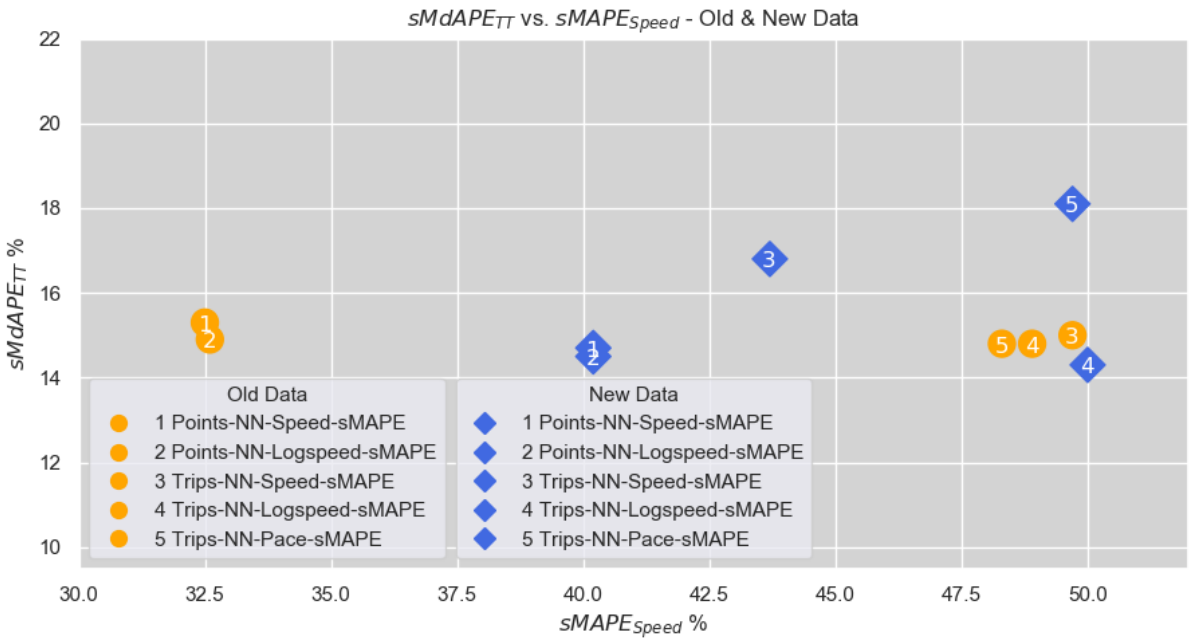

Figure H.2: Travel time prediction accuracy in $\text{sMdAPE}_{\text{TT}}$ versus speed predictions in $\text{sMAPE}_{\text{Speed}}$.

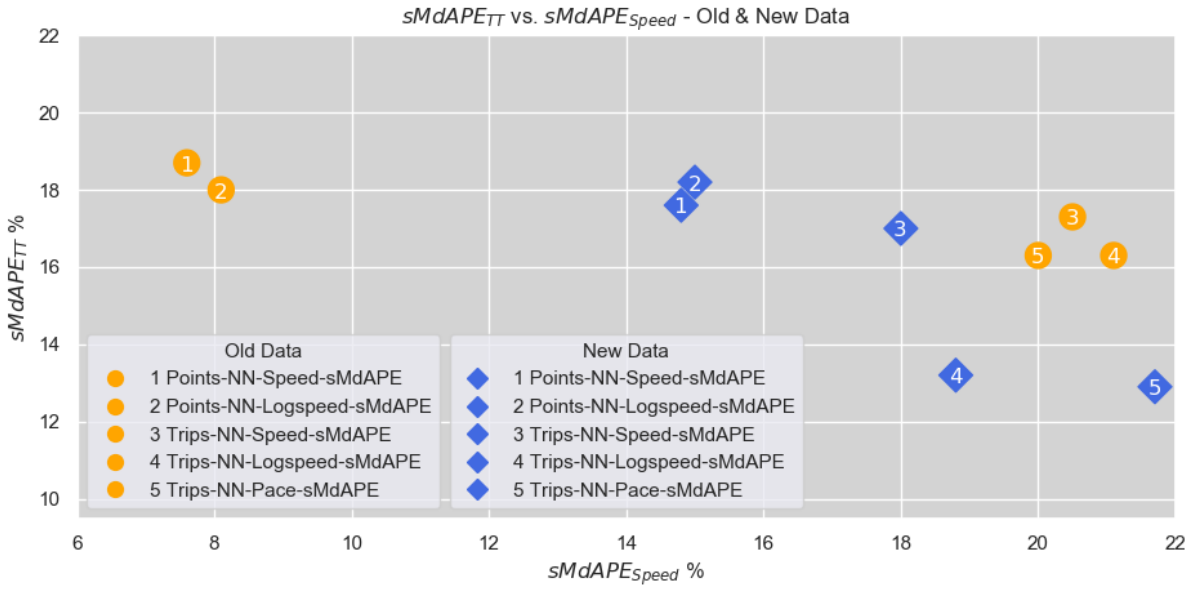

Figure H.3: Travel time prediction accuracy in $\text{sMdAPE}_{\text{TT}}$ versus speed predictions in $\text{sMdAPE}_{\text{Speed}}$.

BIBLIOGRAPHY

[1] B. den Heijer, *Improving Travel Time Predictions for Trucks with Historical GPS-Data*, Master's thesis, University of Amsterdam (2018).

[2] D. Schultes and P. Sanders, *Dynamic highway-node routing,* in *International Workshop on Experimental and Efficient Algorithms* (Springer, 2007) pp. 66–79.

[3] F. Wang and Y. Xu, *Estimating o–d travel time matrix by google maps api: implementation, advantages, and implications,* Annals of GIS **17**, 199 (2011).

[4] A. Jensen and T. Larsen, *Travel-time estimation in road networks using gps data,* Unpublished **8** (2014).

[5] M. Bai, Y. Lin, M. Ma, and P. Wang, *Travel-time prediction methods: A review,* in *International Conference on Smart Computing and Communication* (Springer, 2018) pp. 67–77.

[6] S. Takaba, T. Morita, T. Hada, T. Usami, and M. Yamaguchi, *Estimation and measurement of travel time by vehicle detectors and license plate readers,* in *Vehicle Navigation and Information Systems Conference, 1991*, Vol. 2 (IEEE, 1991) pp. 257–267.

[7] M. Ben-Akiva, M. Bierlaire, D. Burton, H. N. Koutsopoulos, and R. Mishalani, *Network state estimation and prediction for real-time traffic management,* Networks and spatial economics **1**, 293 (2001).

[8] A. Skabardonis and N. Geroliminis, *Real-time estimation of travel times on signalized arterials*, Tech. Rep. (2005).

[9] N. R. Juri, A. Unnikrishnan, and S. T. Waller, *Integrated traffic simulation–statistical analysis framework for online prediction of freeway travel time,* Transportation Research Record **2039**, 24 (2007).

[10] Z. Xiong, D. Rey, T. Mao, H. Liu, V. V. Dixit, and S. T. Waller, *A three-stage framework for motorway travel time prediction,* in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (IEEE, 2014) pp. 816–821.

[11] C. Seybold, *Calibration of fundamental diagrams for travel time predictions based on the cell transmission model*, Master's thesis, Linköping University (2015).

[12] H. Ji, A. Xu, X. Sui, and L. Li, *The applied research of kalman in the dynamic travel time prediction,* in *2010 18th International Conference on Geoinformatics* (IEEE, 2010) pp. 1–5.

[13] J. Kwon, B. Coifman, and P. Bickel, *Day-to-day travel-time trends and travel-time prediction from loop-detector data,* Transportation Research Record **1717**, 120 (2000).

[14] X. Zhang and J. A. Rice, *Short-term travel time prediction,* Transportation Research Part C: Emerging Technologies **11**, 187 (2003).

[15] H. Sun, H. X. Liu, H. Xiao, R. R. He, and B. Ran, *Short term traffic forecasting using the local linear regression model,* in *82nd Annual Meeting of the Transportation Research Board, Washington, DC* (2003).

[16] T. Oda, *An algorithm for prediction of travel time using vehicle sensor data,* in *Third International Conference on Road Traffic Control, 1990.* (IET, 1994) pp. 40–44.

[17] J. Xia, M. Chen, and W. Huang, *A multistep corridor travel-time prediction method using presence-type vehicle detector data,* Journal of Intelligent Transportation Systems **15**, 104 (2011).

[18] M. Chen and S. I. Chien, *Dynamic freeway travel-time prediction with probe vehicle data: Link based versus path based,* Transportation Research Record **1768**, 157 (2001).

[19] L. L. Ojeda, A. Y. Kibangou, and C. C. de Wit, *Online dynamic travel time prediction using speed and flow measurements,* in *2013 European Control Conference (ECC)* (IEEE, 2013) pp. 4045–4050.

[20] X. Liu, S. I. Chien, and M. Chen, *An adaptive model for highway travel time prediction,* Journal of Advanced Transportation **48**, 642 (2014).

[21] D. Park and L. R. Rilett, *Forecasting freeway link travel times with a multilayer feedforward neural network,* Computer-Aided Civil and Infrastructure Engineering **14**, 357 (1999).

[22] N. Wisitpongphan, W. Jitsakul, and D. Jieamumporn, *Travel time prediction using multi-layer feed forward artificial neural network,* in *2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks* (IEEE, 2012) pp. 326–330.

[23] 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).

[24] X. Li, C. Wang, and H. Shi, *A travel time prediction method: Bayesian reasoning state-space neural network,* in *The 2nd International Conference on Information Science and Engineering* (IEEE, 2010) pp. 936–940.

[25] S.-Y. Yun, S. Namkoong, J.-H. Rho, S.-W. Shin, and J.-U. Choi, *A performance evaluation of neural network models in traffic volume forecasting,* Mathematical and Computer Modelling **27**, 293 (1998).

[26] B. Abdulhai, H. Porwal, and W. Recker, *Short term freeway traffic flow prediction using genetically-optimized time-delay-based neural networks,* (1999).

[27] Y. Duan, Y. Lv, and F.-Y. Wang, *Travel time prediction with lstm neural network,* in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)* (IEEE, 2016) pp. 1053–1058.

[28] Y. Liu, Y. Wang, X. Yang, and L. Zhang, *Short-term travel time prediction by deep learning: A comparison of different lstm-dnn models,* in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)* (IEEE, 2017) pp. 1–8.

[29] C.-H. Wu, J.-M. Ho, and D.-T. Lee, *Travel-time prediction with support vector regression,* IEEE transactions on intelligent transportation systems **5**, 276 (2004).

[30] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, *Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions,* Expert systems with applications **36**, 6164 (2009).

[31] P. Gao, J. Hu, H. Zhou, and Y. Zhang, *Travel time prediction with immune genetic algorithm and support vector regression,* in *2016 12th World Congress on Intelligent Control and Automation (WCICA)* (IEEE, 2016) pp. 987–992.

[32] S. Lim and C. Lee, *Data fusion algorithm improves travel time predictions,* IET intelligent transport systems **5**, 302 (2011).

[33] J. Wang, K. Wong, and Y.-Y. Chen, *Short-term travel time estimation and prediction for long freeway corridor using nn and regression,* in *2012 15th International IEEE Conference on Intelligent Transportation Systems* (IEEE, 2012) pp. 582–587.

[34] S. Tak, S. Kim, S. Oh, and H. Yeo, *Development of a data-driven framework for real-time travel time prediction,* Computer-Aided Civil and Infrastructure Engineering **31**, 777 (2016).

[35] B. Hamner, *Predicting travel times with context-dependent random forests by modeling local and aggregate traffic flow,* in *2010 IEEE International Conference on Data Mining Workshops* (IEEE, 2010) pp. 1357–1359.

[36] Y. Zhang and A. Haghani, *A gradient boosting method to improve travel time prediction,* Transportation Research Part C: Emerging Technologies **58**, 308 (2015).

[37] B. Yu, H. Wang, W. Shan, and B. Yao, *Prediction of bus travel time using random forests based on near neighbors,* Computer-Aided Civil and Infrastructure Engineering **33**, 333 (2018).

[38] B. Gupta, S. Awasthi, R. Gupta, L. Ram, P. Kumar, B. R. Prasad, and S. Agarwal, *Taxi travel time prediction using ensemble-based random forest and gradient boosting model,* in *Advances in Big Data and Cloud Computing* (Springer, 2018) pp. 63–78.

[39] J. Anderson, *Travel time prediction in urban road networks,* IFAC Proceedings Volumes **30**, 1109 (1997).

[40] M. T. Asif, J. Dauwels, C. Y. Goh, A. Oran, E. Fathi, M. Xu, M. M. Dhanya, N. Mitrovic, and P. Jaillet, *Unsupervised learning based performance analysis of n-support vector regression for speed prediction of a large road network,* in *2012 15th International IEEE Conference on Intelligent Transportation Systems* (IEEE, 2012) pp. 983–988.

[41] A. Duval, *Explainable artificial intelligence (xai),* (2019).

[42] D. Gunning, *Explainable artificial intelligence (xai),* Defense Advanced Research Projects Agency (DARPA), nd Web **2** (2017).

[43] J. Rodriguez, *Interpretability vs. accuracy: The friction that defines deep learning,* https://towardsdatascience.com/interpretability-vs-accuracy-the-friction-that-defines-deep-learning-dae16c84db5c (2018), [Online; accessed 2019-10-10].

[44] K. Ito and R. Nakano, *Optimizing support vector regression hyperparameters based on cross-validation,* in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, Vol. 3 (IEEE, 2003) pp. 2077–2082.

[45] A. Jain, *Complete machine learning guide to parameter tuning in gradient boosting (gbm) in python,* https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/ (2016), [Online; accessed 2019-10-10].

[46] E. D'Agaro, *Artificial intelligence used in genome analysis studies,* The EuroBiotech Journal **2**, 78 (2018).

[47] https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network.

[48] M. A. Nielsen, *Neural networks and deep learning*, Vol. 25 (Determination press San Francisco, CA, USA:, 2015).

[49] W. L. y. F. Jianjun Luo, Sushma Tayanna and Z. Huang, *An interactive-voting based map matching algorithm,* https://www.slideshare.net/YousefFadila/interactive-voting-based-map-matching-algorithm-80527904 (2017), [Online; accessed 2019-05-26].

[50] U. government, *Gps accuracy,* https://www.gps.gov/systems/gps/performance/accuracy/ (2017), [Online; accessed 2019-05-26].

[51] J. Wolf, *Applications of new technologies in travel surveys,* in *Travel survey methods: Quality and future directions* (Emerald Group Publishing Limited, 2006) pp. 531–544.

[52] F. Van Diggelen and P. Enge, *The worlds first gps mooc and worldwide laboratory using smartphones,* in *Proceedings of the 28th international technical meeting of the satellite division of the institute of navigation (ION GNSS+ 2015)* (2015) pp. 361–369.

[53] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, *Current map-matching algorithms for transport applications: State-of-the art and future research directions,* Transportation research part c: Emerging technologies **15**, 312 (2007).

[54] D. Bernstein, A. Kornhauser, *et al.*, *An introduction to map matching for personal navigation assistants,* (1996).

[55] C. E. White, D. Bernstein, and A. L. Kornhauser, *Some map matching algorithms for personal navigation assistants,* Transportation research part c: emerging technologies **8**, 91 (2000).

[56] H. Yang, S. Cheng, H. Jiang, and S. An, *An enhanced weight-based topological map matching algorithm for intricate urban road network,* Procedia-Social and Behavioral Sciences **96**, 1670 (2013).

[57] J. S. Greenfeld, *Matching gps observations to locations on a digital map,* in *81th annual meeting of the transportation research board*, Vol. 1 (Washington, DC, 2002) pp. 164–173.

[58] M. A. Quddus, W. Y. Ochieng, L. Zhao, and R. B. Noland, *A general map matching algorithm for transport telematics applications,* GPS solutions **7**, 157 (2003).

[59] Y. Zhao, *Vehicle location and navigation systems* (1997).

[60] H. G. B.V., *Fleet telematics route matching,* http://documentation.developer.here.com/pdf/locations_data_to_route_hlp/2.5.9/Fleet%20Telematics%20Route%20Matching%20v2.5.9%20Developer%27s%20Guide.pdf (2018), [Online; accessed 2019-05-26].

[61] J. Macangus, C. Milligan, J. Montufar, and L. Belluz, *Speed characteristics on manitoba's national highway system roads using weigh-in-motion data,* in *2012 Conference and exhibition of the transportation association of Canada-transportation: innovations and opportunities* (2012).

[62] K. Sigakova, *Road freight transport travel time prediction*, Master's thesis, Blekinge Institute of Technology (2012).

[63] O. for Economic Co-operation and Development, *Speed Management* (OECD, 2006).

[64] C. A. Lopez, *Procedures for Establishing Speed Zones* (Texas Department of Transportation, 2006).

[65] W. Brilon and M. Ponzlet, *Variability of speed-flow relationships on german autobahns,* Transportation Research Record **1555**, 91 (1996).

[66] C. Systematics, *Traffic congestion and reliability: Trends and advanced strategies for congestion mitigation*, Tech. Rep. (United States. Federal Highway Administration, 2005).

[67] F. J. Camacho, A. García, and E. Belda, *Analysis of impact of adverse weather on freeway free-flow speed in spain,* Transportation Research Record **2169**, 150 (2010).

[68] M. André and U. Hammarström, *Driving speeds in europe for pollutant emissions estimation,* Transportation Research Part D: Transport and Environment **5**, 321 (2000).

[69] A. T. Ibrahim and F. L. Hall, *Effect of adverse weather conditions on speed-flow-occupancy relationships*, 1457 (1994).

[70] M. Kyte, Z. Khatib, P. Shannon, and F. Kitchener, *Effect of weather on free-flow speed,* Transportation Research Record **1776**, 60 (2001).

[71] M. Kyte, Z. Khatib, P. Shannon, and F. Kitchener, *Effect of environmental factors on free-flow speed,* US National Academy of Sciences Transportation Research Board **18**, 10 (2000).

[72] S. Bassan, *Overview of traffic safety aspects and design in road tunnels,* IATSS research **40**, 35 (2016).

[73] Y. Xu and W. Guo, *Effects of bridge motion and crosswind on ride comfort of road vehicles,* Journal of Wind Engineering and Industrial Aerodynamics **92**, 641 (2004).

[74] P. J. Andueza, *Mathematical models of vehicular speed on mountain roads,* Transportation Research Record (2000).

[75] G. e. a. Allaire, *Massachusetts Highway Department: Project Development & Design Guide* (EOT, 2006).

[76] M. Martens, S. Compte, and N. A. Kaptein, *The effects of road design on speed behaviour: a literature review,* (1997).

[77] T. Ben-Bassat and D. Shinar, *Effect of shoulder width, guardrail and roadway geometry on driver perception and behavior,* Accident Analysis & Prevention **43**, 2142 (2011).

[78] R. Shallam and M. A. Ahmed, *Operating speed models on horizontal curves for two-lane highways,* Transportation Research Procedia **17**, 445 (2016).

[79] J. Edquist, C. Rudin-Brown, and M. G. Lenne, *Road design factors and their interactions with speed and speed limits,* Monash University Accident Research Centre **30** (2009).

[80] E. R. A. Inc., *Driver Attitude to Speeding and Speed Management: A Quantitative and Qualitative Study - Final Report* (EKOS Research Associates Inc., 2005).

[81] F. Fylan, S. Hempel, B. Grunfeld, M. Conner, and R. Lawton, *Effective interventions for speeding motorists,* Contract Number PPAD **9**, 031 (2006).

[82] S. G. Gabany, P. Plummer, and P. Grigg, *Why drivers speed: The speeding perception inventory,* Journal of Safety Research **28**, 29 (1997).

[83] KNMI, *Klimatologie,* https://projects.knmi.nl/klimatologie/uurgegevens/selectie.cgi (n.d.), [Online; accessed 2019-05-27].

[84] J. S. Armstrong and F. Collopy, *Error measures for generalizing about forecasting methods: Empirical comparisons,* International journal of forecasting **8**, 69 (1992).

[85] A. Davydenko and R. Fildes, *Forecast error measures: critical review and practical recommendations,* Business Forecasting: Practical Problems and Solutions. Wiley **34** (2016).

[86] S. Makridakis, *Accuracy measures: theoretical and practical concerns,* International Journal of Forecasting **9**, 527 (1993).

[87] S. Varma and S. Das, *Deep learning,* https://srdas.github.io/DLBook/ (2018), [Online; accessed 2019-05-26].

[88] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting,* The Journal of Machine Learning Research **15**, 1929 (2014).

[89] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT press, 2016).

[90] A. S. Walia, *Types of optimization algorithms used in neural networks and ways to optimize gradient descent,* https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f (2017), [Online; accessed 2019-05-26].

[91] N. Donges, *Gradient descent in a nutshell,* https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0 (2018), [Online; accessed 2019-05-26].

[92] M. Craven and D. Page, *Variance reduction methods,* http://pages.cs.wisc.edu/~spehlmann/cs760/_site//project/2017/05/04/intro.html (n.d.), [Online; accessed 2019-05-26].

[93] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, *On large-batch training for deep learning: Generalization gap and sharp minima,* arXiv preprint arXiv:1609.04836 (2016).

[94] S. Karsoliya, *Approximating number of hidden layer neurons in multiple hidden layer bpnn architecture,* International Journal of Engineering Trends and Technology **3**, 714 (2012).

[95] Z. Boger and H. Guterman, *Knowledge extraction from artificial neural network models,* in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, Vol. 4 (IEEE, 1997) pp. 3030–3035.

[96] M. J. Berry and G. S. Linoff, *Data mining techniques: for marketing, sales, and customer relationship management* (John Wiley & Sons, 2004).

[97] A. Blum, *Neural networks in C++: an object-oriented framework for building connectionist systems* (John Wiley & Sons, Inc., 1992).

[98] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural network design*, Vol. 20 (Pws Pub. Boston, 1996).

[99] J. Heaton, *Introduction to neural networks with Java* (Heaton Research, Inc., 2008).

[100] S. Ruder, *An overview of gradient descent optimization algorithms,* arXiv preprint arXiv:1609.04747 (2016).

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

[102] S. Jadon, *Introduction to different activation functions for deep learning,* https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092 (2018), [Online; accessed 2019-05-26].

[103] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and accurate deep network learning by exponential linear units (elus),* arXiv preprint arXiv:1511.07289 (2015).

[104] M. Bilgili, B. Sahin, and A. Yasar, *Application of artificial neural networks for the wind speed prediction of target station using reference stations data,* Renewable Energy **32**, 2350 (2007).

[105] P. Gupta, *Cross-validation in machine learning,* https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f (2017), [Online; accessed 2019-05-26].

[106] S. Norena, *Python model tuning methods using cross validation and grid search,* https://medium.com/@sebastiannorena/some-model-tuning-methods-bfef3e6544f0 (2018), [Online; accessed 2019-05-26].

[107] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, Vol. 112 (Springer, 2013).

[108] N. Latysheva and C. Ravarani, *Scanning hyperspace: how to tune machine learning models,* https://cambridgecoding.wordpress.com/2016/04/03/scanning-hyperspace-how-to-tune-machine-learning-models/ (2016), [Online; accessed 2019-05-26].

[109] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, *Lstm: A search space odyssey,* IEEE transactions on neural networks and learning systems **28**, 2222 (2016).

[110] L. N. Smith, *A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay,* arXiv preprint arXiv:1803.09820 (2018).

[111] A. Klein, E. Christiansen, K. Murphy, and F. Hutter, *Towards reproducible neural architecture and hyperparameter search,* (2018).

[112] J. Bergstra and Y. Bengio, *Random search for hyper-parameter optimization,* Journal of Machine Learning Research **13**, 281 (2012).

[113] J. Snoek, H. Larochelle, and R. P. Adams, *Practical bayesian optimization of machine learning algorithms,* in *Advances in neural information processing systems* (2012) pp. 2951–2959.

[114] MathWorks, *Machine learning in matlab,* https://in.mathworks.com/help/stats/machine-learning-in-matlab.html?w.mathworks.com (2019), [Online; accessed 2019-05-26].

[115] S. Jain, *An overview of regularization techniques in deep learning (with python code),* https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/ (2018), [Online; accessed 2019-05-26].

[116] T. Kleynhans, M. Montanaro, A. Gerace, and C. Kanan, *Predicting top-of-atmosphere thermal radiance using merra-2 atmospheric data with deep learning,* Remote Sensing **9**, 1133 (2017).

[117] L. Breiman, *Random forests,* Machine learning **45**, 5 (2001).

[118] A. Verikas, E. Vaiciukynas, A. Gelzinis, J. Parker, and M. Olsson, *Electromyographic patterns during golf swing: Activation sequence profiling and prediction of shot effectiveness,* Sensors **16**, 592 (2016).

[119] J. D'Souza, *A quick guide to boosting in ml,* https://medium.com/greyatom/a-quick-guide-to-boosting-in-ml-acf7c1585cb5 (2018), [Online; accessed 2019-05-26].

[120] *Introduction to boosted trees,* https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/ (2017), [Online; accessed 2019-05-26].

[121] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, Vol. 112 (Springer, 2013).

[122] *Introduction to boosted trees,* https://www.tottadatalab.nl/2019/01/07/gradient-boosting-algoritme/ (2019), [Online; accessed 2019-05-26].

[123] J. Bergstra and Y. Bengio, *Random search for hyper-parameter optimization,* Journal of Machine Learning Research **13**, 281 (2012).

[124] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, *Taking the human out of the loop: A review of bayesian optimization,* Proceedings of the IEEE **104**, 148 (2015).