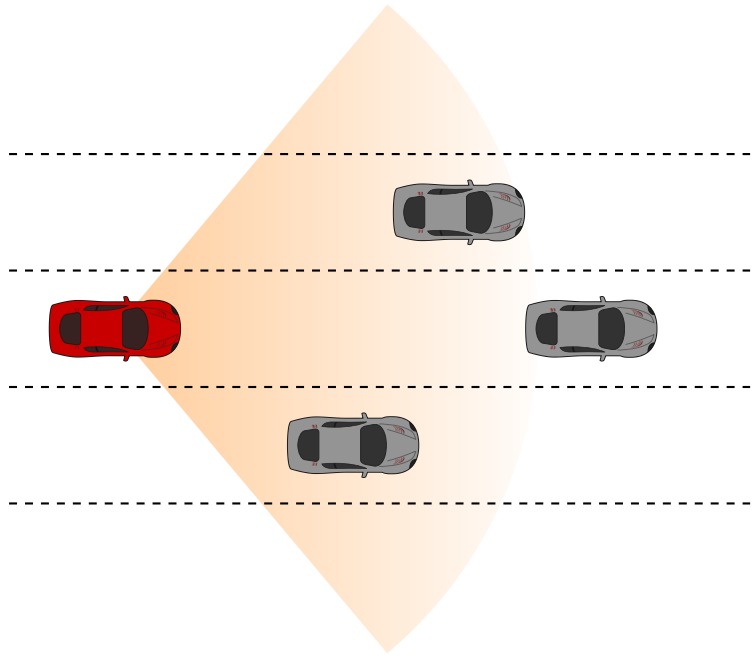
CHALMERS
UNIVERSITY OF TECHNOLOGY



Deep Normal Driving

A deep dive into driver modelling using a data-driven approach

Master's thesis in Systems, Control and Mechatronics

GUSTAV ANDERSSON
EDVIN ASPELIN

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

Deep Normal Driving

A deep dive into driver modelling using a data-driven approach

GUSTAV ANDERSSON
EDVIN ASPELIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Deep Normal Driving
A deep dive into driver modelling using a data-driven approach
GUSTAV ANDERSSON
EDVIN ASPELIN

Supervisor: Anders Ödblom, Volvo Car Corporation
Supervisor: Carl Toft, Computer Vision and Medical Image Analysis
Examiner: Fredrik Kahl, Computer Vision and Medical Image Analysis

Master's Thesis 2019
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An example of a multi-lane road-state.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

Deep Normal Driving
A deep dive into driver modelling using a data-driven approach
GUSTAV ANDERSSON
EDVIN ASPELIN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Self-driving cars seek to relieve the driver from the task of driving, and the new features that will come from the development of autonomous vehicles will introduce a whole new paradigm for the automotive industry. Developing these features calls for smart and efficient testing. These tests are performed more and more in simulation environments to decrease costs and to increase the efficiency. Hence, new technologies tangential to the research in autonomous drive will need to have further developed toolboxes for testing through simulation. Enabling the simulation of normal drivers would thus be beneficial in this process.

This research proposes a methodology for modelling human behaviour in a multi-lane highway environment using a data-driven approach. The model makes use of a Long Short-Term Memory (LSTM) network to gather ego, object and road variables, and then predict the future trajectory of the ego vehicle. Studying the ability to navigate through traffic, it is important for the model to both plan the direct path on the road and the ability to identify future actions, like changing lanes. Further, performance measures are introduced in order to investigate the behavioural outcome of the developed model. The performance is measured both based on the quality of the prediction given the input, and then also the performance of predicting future manoeuvres that are not yet initialised.

Thus, the research topics do not only reflect an investigation for methods modelling a normal driver but also underscores the methods of verifying the result. The proposed model is shown to, with a high precision, be able to predict the future trajectory of the ego vehicle and also predict future lane changes. The model is also verified to be aware of surrounding objects in the predictions.

Keywords: Trajectory Prediction, Driver Intention, Machine Learning, Deep Learning, Artificial Neural Networks, RNN, LSTM.

Acknowledgements

Hi,

When you are reading this report, keep in mind that the work done would not have been possible without some great people's involvement and their interest in our work.

We have had the pleasure of working with two tremendous supervisors from both Volvo Cars Corporation and Chalmers University of Technology. Anders Ödblom from Volvo Cars Corporation has been our major contact through the work, and with his keen eyes, he has always asked the hard questions that made us both apply more critical thinking and strive for a higher level. Carl Toft, our supervisor at Chalmers, being the bright spirit he is and always giving great support and input in our work, he was always there to answer any of our questions. We were always amazed by the level of engagement from our supervisors and we are very grateful that we had the opportunity to work with them.

We would also like to highlight the work made by Alexander Bükk and Rickard Johansson, creating the start point of this research. Without their help, we would still be in the cave of segmenting tons of data sequences.

Finally, we would like to thank everyone in the office at Volvo Cars Corporation that made this last part of our degrees a truly memorable time. With that said, hope you will enjoy the results we produced during this time period.

Best regards,

Gustav Andersson & Edvin Aspelin, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1

Introduction

In recent years, the idea of self-driving cars has motivated researchers all over the world and is a popular topic in technological media. Relieving the driver from the task of driving will be an entirely new paradigm for the automotive industry and is expected to radically transform means of transportation. Furthermore, self-driving cars also move the driver's responsibility of driving safely and environmentally friendly to the car manufacturers instead, opening up for more sophisticated work towards increasing safety and effectiveness.

As always, we find ourselves in a position where verification is necessary to deliver safe and reliable products. With new technology raising the level of automation, there is a need to create appropriate methods in order to verify the safety level of these technical solutions. Now, modelling a *normal* driver will enable verification methods where new automotive vehicles can be tested in software, letting it interact in a synthetic traffic environment.

These new tests will depend heavily on the behaviour model for interacting vehicles, and the goal of this research is to propose a data-driven machine learning approach to create a driver model which mimics the human behaviour and his or her interaction in traffic.

1.1 Background

A data-driven approach to trajectory predictions using machine learning has shown exceptional results in previous works, [1–3]. In fact, neural networks in general have proven to be a prevalent choice in almost any prediction-based task. Nowadays machine learning can be seen in tasks ranging everywhere from image classifications to teaching a robot complex movement skills, [4,5]. Also, mimicking physical motion in humans is a popular usage of machine learning, for example, networks that imitate human handwriting and speech, [6, 7].

In other words, neural networks are excellent at learning *the how*. How to talk, how to walk, and how to write. However, teaching someone letters and sentences does not mean that they can write a novel. The complexity of teaching a network to write a novel and evaluating the result would grow rampantly in comparison to the letters and sentences. And how would you measure the performance of such a

network and the produced novel? What is a good novel? If you knew exactly how to write the best novel, the Pulitzer prize would not be far away - if you have the answer, please let us know.

Making the same analogy in driving, there is a difference between knowing *how* to overtake another vehicle and knowing *when* to do it. In autonomous vehicles, this could be a crucial distinction when interacting with other vehicles. It is therefore essential to investigate how to predict the intention of a driver. We would like to introduce a network that innately combines both prediction and intention, using the network structure and processing of the input data to achieve that. Uniting trajectory prediction and intention prediction in a network like this has become the vision of this project, and it seeks to investigate both the *how* and *when* in a network's performance.

1.2 Aim

The aim of this research is to model driver intention and trajectory prediction, using a supervised deep learning approach. The goal is to design a model which makes use of sequential input of data collected from the surroundings of a vehicle to mimic human behaviour. The research will use a vanilla Long Short-Term Memory (LSTM) network as a starting point and further develop and evaluate it to propose a structured methodology for creating a network suited for highway driving. This research attempts to evaluate how the LSTM network can benefit from altering the network structure and preprocess the dataset to skew its focus towards surrounding objects. Thus, there are two focus points of the research; data preprocessing and network structure. How the processing of the dataset affects the performance outcome of the network will need to be empirically tested, as the dataset is very much unique in regards of sensor disruptions, data structures etc. Evaluating the network structure will include performance measures of different network structures and hyper-parameters. Developing means of evaluating both the quality of trajectory predictions and driver intentions is key to analysing the performance.

It is a necessity that the data is interpretable and properly invariant for the network to function as intended. Thus the aim includes exploring possibilities of using alternative methods of expressing the road state. In the used dataset, surrounding objects are described as solely longitudinal and lateral coordinates relative to the ego vehicle. A challenge with the dataset is that it uses gathered sensor data as ground truth. Even though the measurements may be close to the true value, the network may inherit any faults from the data. Hence we will investigate if the performance can be improved with other methods of describing the input data.

The essential challenges and research topics to investigate are:

- Data transformation and pre-processing.
- Implementation of neural network models.
- Investigating performance of network structures.

- Training and evaluation of models.
- From the results, determine what network has the best performance and evaluate its strengths and weaknesses.
- Evaluate the model's performance correlations in regards to input features and make an assessment of the environmental importance of surrounding objects.

1.3 Limitations

The project limits itself to only observe data from highway scenarios. In other words, the data is selected from datasets from where the ego vehicle operates in a multi-lane environment with a velocity greater than 70 km/h. The dataset contains 6500 recorded driving logs with solely frontal field of view, where each log contains roughly 80 seconds of data and includes at least one lane change. Currently, a subset of the collected data parameters has been processed and restored, for example interpolating missing lane markers. The processed dataset contains; ego yaw rate and velocity, ego lateral distance from lane markers, and relative spatial displacement of surrounding objects.

Deciding on network architecture is also limited to only investigating structures of LSTM networks.

1.4 Related work

It is not obvious to use machine learning for vehicle trajectory predictions. In [8], Houenou et al. instead uses motion modelling to predict the trajectory of surrounding objects. The vehicles are given a constant yaw rate and constant acceleration motion model and then combines it with a manoeuvre recognition model to make predictions about future positions. Rather than having a neural network help with decisions, they use a deterministic model for manoeuvre recognition. The model is solely based on kinematic measurements and road geometry detections. A downside of this model is that it will not scale with more data, and may become a limitation in the performance of the trajectory predictions.

Other models, such as hidden Markov models and dynamic Bayesian networks, have also been used for various prediction tasks. Ye et al. [9] suggest a hidden Markov model in combination with linear layers to make vehicle trajectory prediction and Wöllmer et al. [10] use a dynamic Bayesian network combined with an LSTM network for artificial listening. Both these implementations use another model than a neural network to handle some sub-task in the tool-chain.

Another similar work has been done by Altché and La Fortelle [11], where a Long Short-Term Memory is used to predict a vehicle's future trajectory. It shows the possibility of predicting trajectories based on real driver log data. The results show that they can do so with a reasonable error, and the implications from the

network performance seem promising. However, there are not any further analysis of the results other than the loss. Alexander Bükk and Rickard Johansson have also created an implementation of an LSTM network to model the driver, which proved potent in predicting future trajectories [12]. It was however shown that the model was not interaction aware, in the sense that surrounding objects did not affect the prediction. We now want to investigate if it is possible to achieve a high-performance neural network that can mimic those interactions from human driving.

2

Theory

Neural networks are a convincing alternative for almost any analytic task. The results are continuously improving and new network structures are invented by the day. According to the *Universal approximation theorem* [13], a single layer neural network can approximate *any* given function. The size of the network would in many cases be unfeasibly large and the network may fail to learn from the input data, but it somewhat reflects the great expressiveness of neural networks.

This paper will not attempt to improve on the massively iterated explanation of the fundamentals in neural networks. Instead, this chapter will go through the underlying theory used to model, train and evaluate the Long Short-Term Memory (LSTM) networks that are used in this research. The desired outcome is to give the reader an understanding of how LSTM networks can be implemented to perform in a satisfying manner in situations related to the research topics.

However, if you are unfamiliar with Artificial Neural Networks in general, or just need a quick refresher, Ian Goodfellow et al. [14] made an exceptional introduction to deep learning and the basics of neural networks.

2.1 Recurrent Neural Networks

> *Humans dont start their thinking from scratch every second. [...]*
> *Your thoughts have persistence.*
>
> — *Christopher Olah, [15]*

A major shortcoming with traditional neural networks is their disability to remember previous inputs. Remembering the input is necessary for any sequence-based information input, e.g. semantics in text or approximating velocity from position coordinates. Recurrent Neural Networks (RNN) seeks to address this issue by introducing a continuous memory in each cell. With its internal memory, the cell is able to remember the input history by updating the memory with each sequential input. It does not remember the history exactly, per se, but instead expresses it in its own high dimensional space through the sequence.

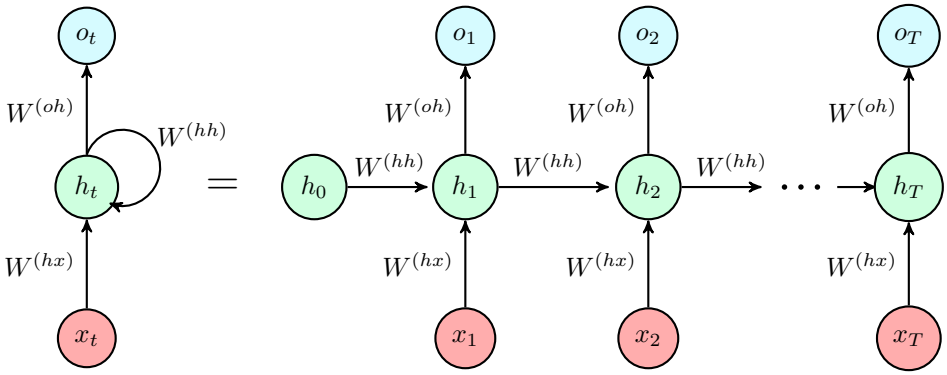


Figure 2.1: Two visualisations of a single recurrent neuron. The left part represents a rolled RNN for each time step and the right part the corresponding unrolled RNN.

The RNN cell is visualised in Fig. 2.1 both with a loop going from the output back as an input, or an unrolled version where the sequential input and output can be seen more clearly. Note that both visualisations are only different visual takes on the same cell structure. The memory content is passed through each iteration and is used to determine the output, thus allowing for content to be conserved from one step to another. The cells use the sigmoid function, σ, as activation function. Each step in the sequence can be computed by the following equations, with the matrices W and b being the weights and biases respectively.

$$\sigma(x) = \frac{e^x}{1 + e^x} \tag{2.1}$$

$$h_t = \sigma(W^{(hh)} h_{t-1} + W^{(hx)} x_t + b^{(h)}) \tag{2.2}$$

$$o_t = \sigma(W^{(oh)} \cdot h_t + b^{(o)}) \tag{2.3}$$

Each weight matrix W is optimised in the training process, and it can be seen in Eq. (2.2)-(2.3) how the memory, h_t, affects the output, o_t. In supervised learning, the optimal outcome of the network is known. Through penalising an error in the prediction with a loss function, for example with a Mean Square Error (MSE), the weights can be adjusted to make better predictions. Weight optimisation is then done through back-propagating the network through time, which is a method to calculate the gradient needed for the weight update. The proof of back-propagation through time is rather long and is not reflected in the topics in this research, but it is fundamental in weight optimisation for RNN and B. Mehlig presents an exquisite formulation of it in [16].

In specific fields, RNN networks have been truly groundbreaking, and the tremendous results are touched upon in Andrej Karpathy's *The Unreasonable Effectiveness of Recurrent Neural Networks* [1]. But a simple RNN does not come without some uncomfortable bumps, with the most adverse one being the vanishing and exploding gradient problem. It basically stems down to the results from the back-propagation through time, where the weight matrix $W^{(hh)}$ decides if the gradient is going towards

zero or infinity. At the time t, the gradient in regards to some previous time step k, is calculated like the following,

$$\frac{\partial h_t}{\partial h_k} = \prod_{t \geq i \geq k} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{t \geq i \geq k} W^{(hh)^T} diag\Big(\frac{d}{dh}\sigma(h_{i-1})\Big) \tag{2.4}$$

In conclusion from Eq. (2.4), when increasing the time frame of the sequence, $W^{(hh)}$ will either make the gradient increase or decrease exponentially. This quickly infers problems when increasing the sequence length of the input and/or output. Trying to find an optimal weight matrix with a tiny gradient will result in a large increase in training time, and with a huge gradient, the optimizer have a hard time finding optimal points. One can reduce the impact of the exploding gradient by various methods, for example, clipping the gradient at a certain threshold. However, the vanishing gradient problem persists and one needs to look at other cell structures to address this problem.

2.2 Long Short-Term Memory

In 1997, Hochreiter and Schmidhuber presented their work in [17], which aims to address the issues of classic RNN structures. More precisely, they proposed a solution for the gradient problems. The fundamental idea of the proposed solution, the LSTM cell, is to truncate the back-propagation gradient, where the truncation does no harm in performance. LSTM cells will thus enforce a constant error flow through the sequence with its use of *Constant Error Carousels*, (CEC), which is the central feature of the LSTM cell. The special structure of LSTM cells allows for the training of multiplicative gate units to create a communication channel from the input and memory cell to the constant error flow, to significantly increase its learning rate capabilities compared to a classic RNN.

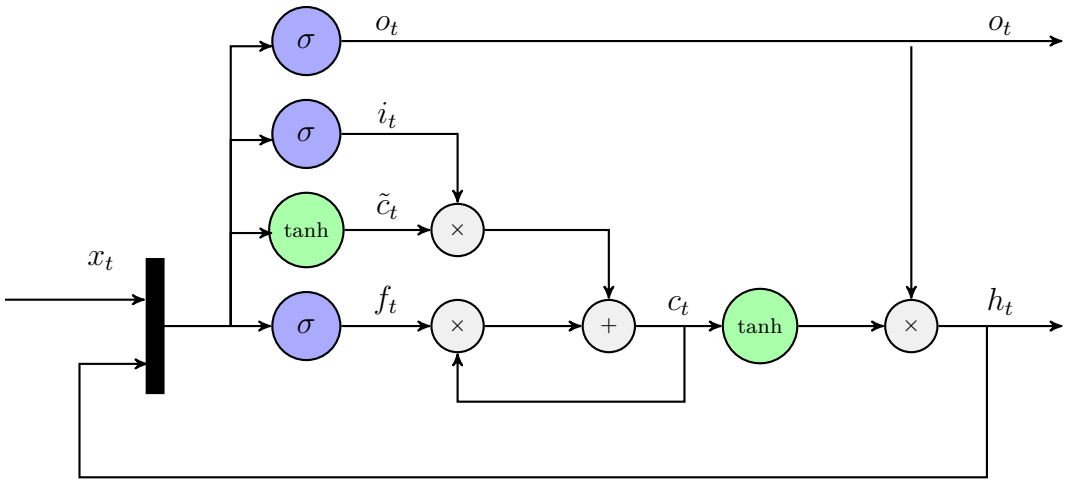


Figure 2.2: A basic LSTM cell with linear operators (grey), and sigmoid (blue) and tanh (green) activation functions.

The implementation of an LSTM cell in Fig. 2.2 is not very different from implementing a recurrent neuron. It basically proposes an alteration of the memory update equation, Eq. (2.2).

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \tag{2.5}$$
$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \tag{2.6}$$
$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \tag{2.7}$$
$$\tilde{c}_t = \tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c) \tag{2.8}$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{2.9}$$
$$h_t = o_t \cdot \tanh(c_t) \tag{2.10}$$

With the altered equations, the cell is able to internally pass it's constant error flow in the CEC-loop, which can be observed in Fig. 2.2. How the cell updates it's internal states can be described step-by-step. Firstly, the cell decides what it is going to forget from it's former internal content state, c_{t-1}. This is done by the forget gate, f_t, which is seen in both Eq. (2.6) and (2.9). Then the cell decides what information should be added to the cell state with the use of both the candidate $\tilde{c}_t$ and the input gate, i_t, in Eq. (2.5) and (2.8). Finally, it calculates the output h_t in Eq. (2.10), which is also affected by the internal cell state, c_t.

The main features of the LSTM cell have made it a popular choice for sequenced-based data inputs because of its innate ability to store the cell memory over a long period of time. There exist some variations of the LSTM cell structure, but the one used in this research is the one presented in this chapter and is also the one mainly referred to as the standard LSTM cell.

2.3 Linear layer

A linear layer is used as a simple network layer that only applies a linear transformation to the input, with the weight matrix W and the bias vector b. The linear layer is sometimes also called a dense layer or a fully connected layer, but the structure and the linear operation is still the same and can be seen in Fig. 2.3 and Eq. (2.11) respectively.

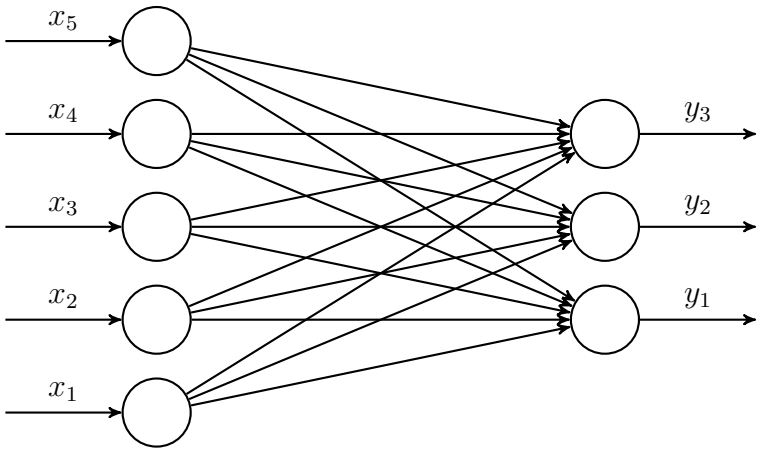


Figure 2.3: Visual representation of a linear layer in a network.

$$y = W \cdot x + b \tag{2.11}$$

From the previous walk-through of the LSTM cell structure, we learned that the LSTM cell outputs the hidden state, h_t. With the hidden state most likely not being the same dimension as the desired output, a final layer is applied to map the LSTM output to the correct form. Input x to the linear layer is then set to h_t. The weight matrix W and vector b is then optimised along with all the other parameters in the network training process.

2.4 Data normalisation

Normalisation in neural networks is a method that aims to increase learning speed, make the network more robust, and increase its performance. It assures that there are no activation functions that, relative to others, go very high or low, creating a rippling effect of bad comparisons through the network. With the range of different input channels not far apart, the normalised input circumvent this behaviour. Making it easier for the network to compare values also allows for higher learning rates and work as a catalyst for better performance.

In neural networks, the input data is often normalised to have both mean and variance of value 0 and 1 respectively, and the normalisation is done separately on

each input channel. For a dataset with n input channels, $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(n)}\}$, where $\mathbf{x}$ contains all values for that channel in the dataset, with a total number of data sample points K, the new normalised input channels are calculated by the following equations.

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_K^{(i)} \end{bmatrix} \tag{2.12}$$

$$\bar{x}^{(i)} = \frac{1}{K} \sum \mathbf{x}^{(i)}, \ \ \forall i \in \{1, 2, ..., \ n\} \tag{2.13}$$

$$\sigma^{(i)^2} = \frac{(\mathbf{x}^{(i)} - \bar{x}^{(i)})^T (\mathbf{x}^{(i)} - \bar{x}^{(i)})}{K - 1} \tag{2.14}$$

$$\tilde{\mathbf{x}}^{(i)} = \frac{\mathbf{x}^{(i)} - \bar{x}^{(i)}}{\sqrt{\sigma^{(i)^2}}} \tag{2.15}$$

$$\Rightarrow \tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}^{(1)}, \tilde{\mathbf{x}}^{(2)}, ..., \tilde{\mathbf{x}}^{(n)}\} \tag{2.16}$$

Thus, $\tilde{\mathbf{X}}$ is the new normalised input to the network. As the normalisation is calculated solely for the training set, the same normalisation parameters are later used to normalise the validation data. The reason for the validation data not included in the normalisation parameters is to reassure that the validation data not by any means affect the training data.

3

Methods

The goal is to present a network that reflects the abilities of a human driver. Doing so includes the whole tool-chain, from raw dataset to performance measures and behavioural analysis of the proposed model. Thus, the insights of the methods used to develop the driver model are presented to thoroughly walk through each step of the process. It also gives a description of the dataset used to produce the results and the methodology for constructing the network model. Furthermore, it covers the methods of evaluation and all of the performance metrics.

3.1 Dataset

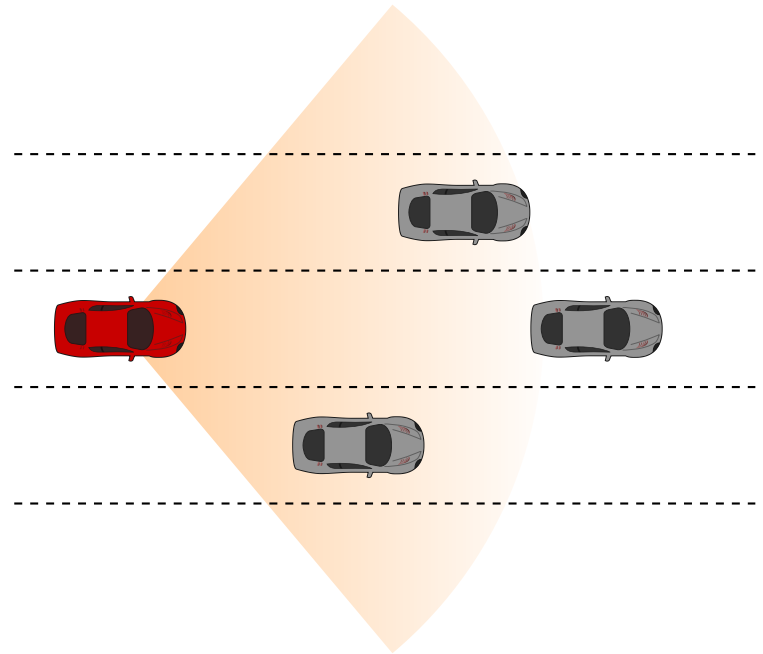


Figure 3.1: The used data are sampled with an integrated sensor system, where the field of view of the equipment covers the front part of the ego vehicle and can detect objects up to R_{max} meters ahead.

The data later used for training the model is comprised of data logs gathered from driver expeditions made to various destinations around the globe. Consisting of 6500 recorded sequences, roughly 80 seconds each, both the driver and the environment around the vehicle are varying in the dataset. Each sequence contains roughly 320 data collection samples with a sampling frequency of 4 Hz. Of the total number of gathered logs, logs have been specifically selected to only consist of a multi-lane environment, with a speed constantly above 70 km/h, and throughout each sequence at least one overtake is present. Objects are represented by xy-coordinates, which are sensor estimations of where the objects are located relative to the ego vehicle. Also, the sampled data only contains forward-looking object detections, which means that only objects in front of the car, up to a set distance R_{max} ahead, will be a part of the dataset. A top view of the sensor vision used to gather the data logs in the dataset is represented in Fig. 3.1.

Furthermore in the classification, an object includes the object types; cars, trucks, motorbikes, or an unclassified road object. Lane markers in the dataset are approximated as N-degree polynomials with parameters a, as in Eq. (3.1). Where x is the longitudinal distance along the road in ego body frame coordinates, starting from ego position $x = 0$. Then for each lane marker, j:

$$y_j = \sum_{i=0}^{N} a_{i,j} x^i = f_j(x) \tag{3.1}$$

Lane marker approximations are calculated and updated in each time step, assuring that only the latest approximation is the one used. The approximation is done for each lane marker present on the road. Summarising the contents of the collected data gives the following features:

- Yaw rate of ego vehicle, $\dot{\psi}$
- Ego forward velocity, v
- Object positions relative to ego vehicle, (x_i, y_i)
- Lane marker polynomials, $y_j = f_j(x)$

Figure 3.2: Top view of a single lane with ego vehicle (red) and a single vehicle (grey). Note that relative coordinates of surrounding objects, (x_k, y_k), are given in the ego vehicle's body frame, where the x-axis is along the forward direction of the ego vehicle and the y-axis perpendicular to it in a right-handed coordinate system.

The logs also underwent some data processing where missing road line estimations have been interpolated to maintain the integrity of the lane markers. Also, objects identified outside of the outer lane markers are discarded. The dataset is split with a 90/10 ratio, where 90% of the dataset is placed in the training set and the remaining 10% in the validation set.

3.1.1 Data distribution

In this section, the complete training and validation dataset is analysed to facilitate the opportunity to explore what behaviour the network may inherit from fitting to the training data. It also enables an investigation of if the data is enough to properly train the network to make vehicle trajectory predictions.

Analysing ego velocity and object detections for each time sample in the training set and validation set respectively, the distribution of said data in the dataset is presented in Fig. 3.3 and 3.4. The interval of the velocity is shown to be within 70 km/h up to 160 km/h. In order to mask the performance of the sensor setups object detection, objects have been grouped together. This means that each group, g_i, can include various object types and/or number of objects. However, times where the sensor setup detects no objects are not masked.

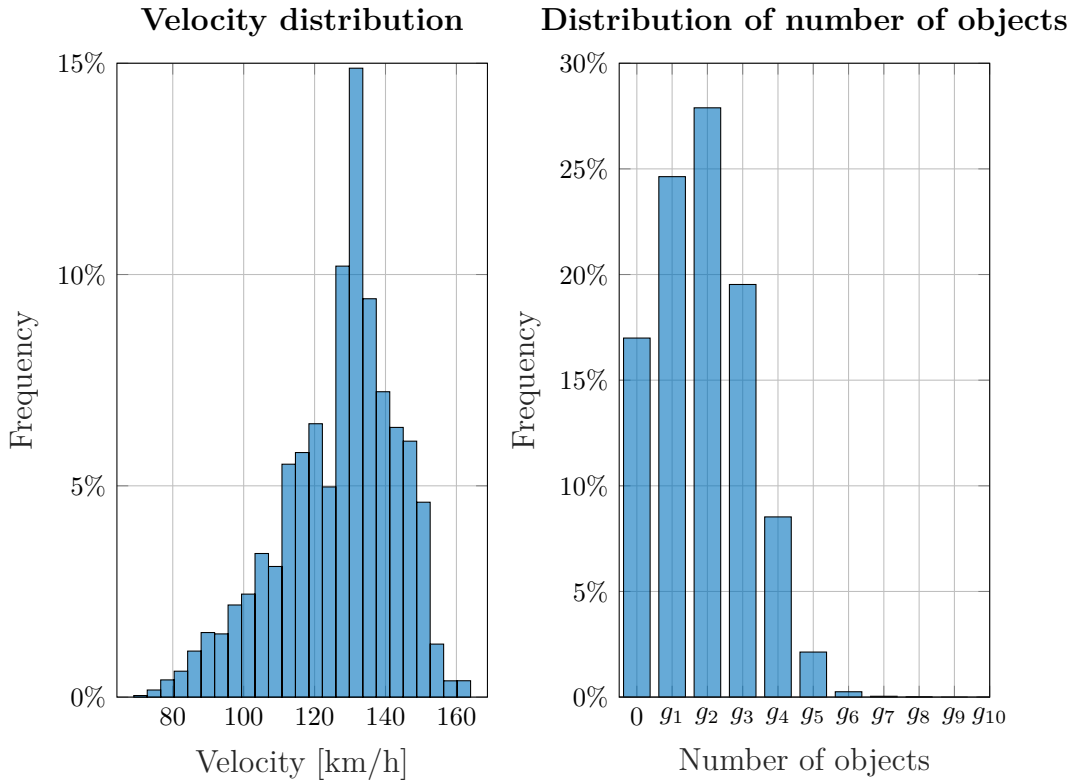


Figure 3.3: Histograms showing the distribution of ego velocity and the exposure to multiple objects throughout the dataset. The statistics are made from analysing each time sample in the complete training dataset.

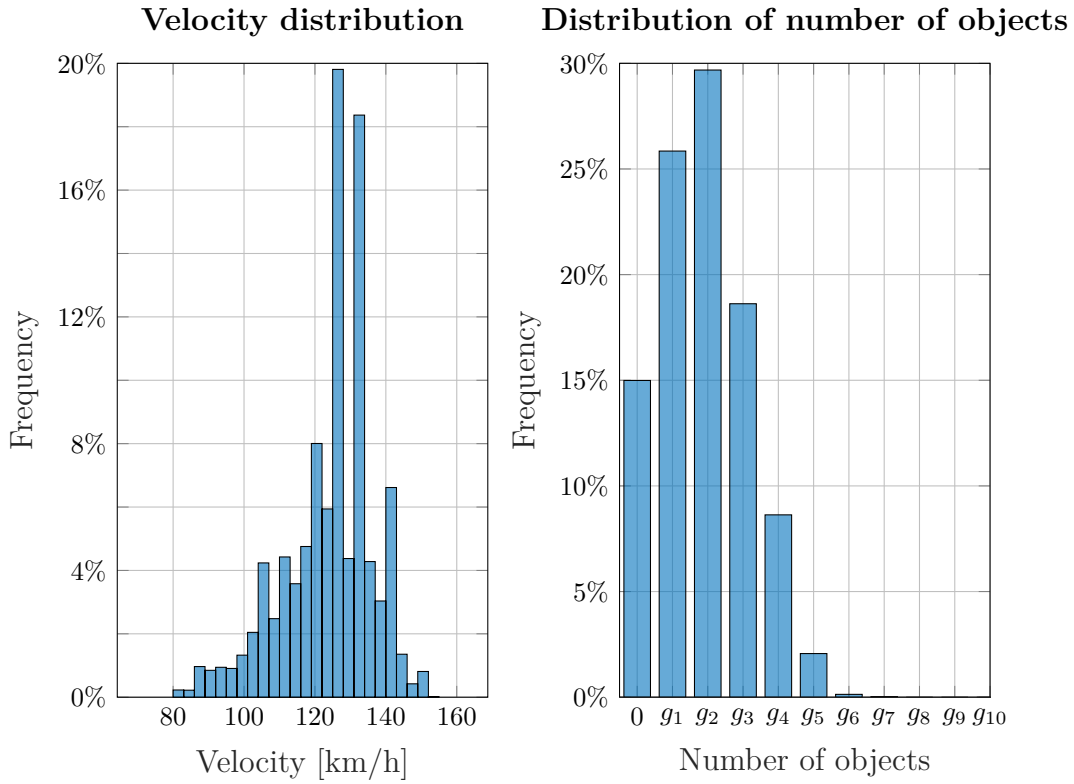


Figure 3.4: Histograms showing the distribution of ego velocity and the exposure to multiple objects throughout the dataset. The statistics are made from analysing each time sample in the complete validation dataset.

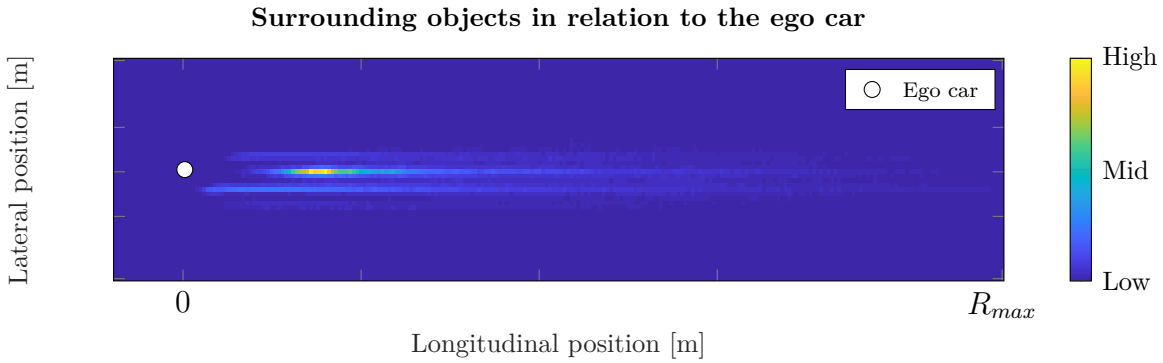


Figure 3.5: Each time sample in the training set is analysed to produce a heat map over where surrounding objects are located throughout the training dataset.

In both Fig. 3.5 and 3.6 the positions of surrounding objects are presented. In Fig. 3.5, the intensity displayed is based on each time sample. However, a single object travelling in the same speed right in front of the ego vehicle has a greater time exposure than an object which the ego vehicle overtakes, but in reality, they are still just one object. Instead, Fig. 3.6 shows where objects have travelled on a *sequence-based* time measure. Meaning each sequence in the dataset is analysed to create a binary grid of locations where objects have been located in that specific sequence, and then each location grid for all sequences are added together.

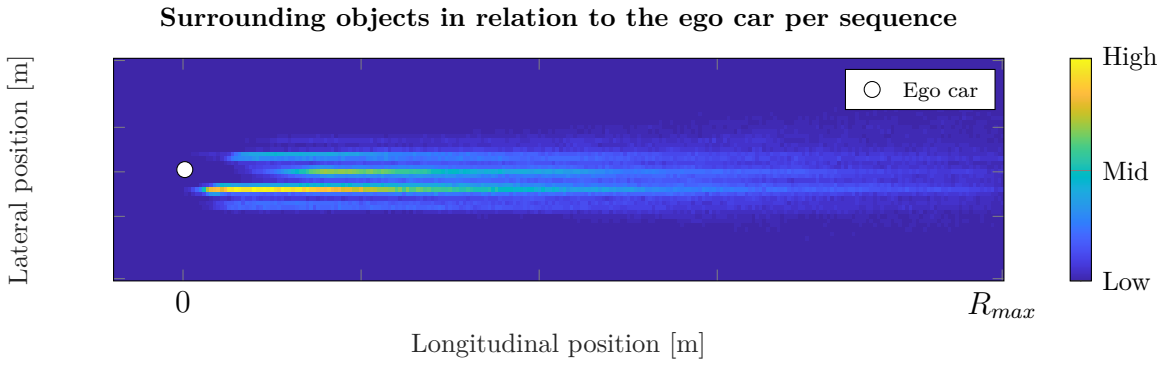


Figure 3.6: To emphasise how objects travel in relation to the ego vehicle, the heat map is produced by observing in what coordinates there has been an object throughout a each single sequence and then summarising the data sequence-wise. This is done on the training set.

Fig. 3.7 shows the distribution of surrounding objects at the time instance of the lane change. The distinct spread in lateral offset over distance is caused by the varying curvature of the road and the yaw angle of the vehicle, as the data is presented in ego-vehicle body coordinates and not in the road frame.

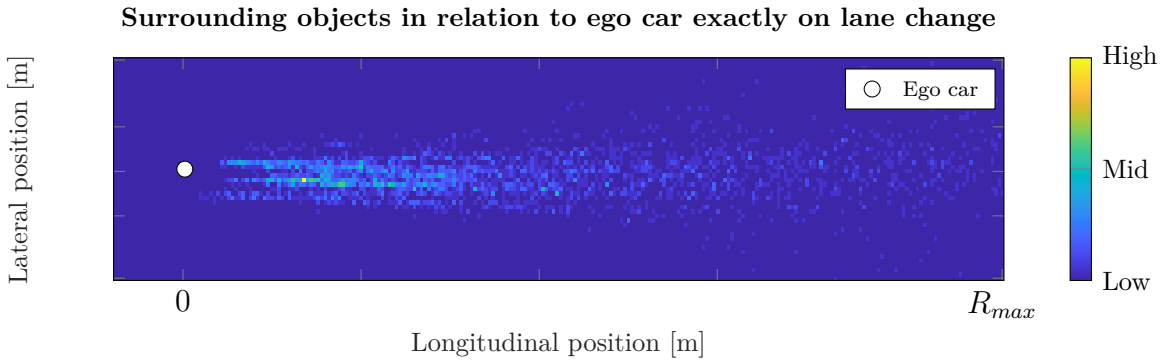


Figure 3.7: Heat map showing where objects are detected relative to the ego vehicle at the moment the ego vehicle crosses a lane marker, analysed through the training set.

Lastly, the number of lane changes are presented in Tab. 3.1. A lane change is considered as the ego vehicle crossing a lane marker.

Table 3.1: Summarising the average number of left and right ego vehicle lane changes in each sequence.

Turn type	Average per sequence
Left	0.895
Right	0.895
Total	**1.79**

Each sequence has an average of nearly two lane changes, presented in Tab. 3.1. This means that the selected dataset contains about two lane changes in each sequence of 80 seconds and contains equally many lefts as right turns.

3.1.2 Summary

From the dataset analysis, it can be seen that the ego vehicle has plenty of multi-object exposure throughout the dataset. The speed of the ego vehicle is not uniformly distributed, but it still somewhat follows a normal distribution. Also, the ego vehicle has a tendency to have other cars in its right lane compared to the left.

The question still stands if the data is sufficient to properly train a network to mimic the driver's movements. But the conclusion is that there is no obvious lack of data in the dataset, other than the low exposure of some object groups, that may inhibit the training process.

3.2 Network architecture

Designing a neural network could almost be considered half science half art. There are a lot of parameters that can be optimised and they can vary in vast ranges. And when something structural changes, such as hidden dimensions or the number of cells in the network, the optimisation probably needs to be readjusted. This section describes the methods used in this research to model the final vehicle trajectory prediction network. It will also go through how the network is trained to achieve a satisfying result.

3.2.1 Network modelling

From the dataset, ground truth positions of where the ego vehicle travelled in each sequence have been extracted. Positions are given as longitudinal and lateral co-

ordinates, such that each position corresponds to two data points, (x_i, y_i). These positions are then used as the target output of the network. The network is thereby trained to predict future positions relative to the current position. An overview of the scenario can be seen in Fig. 3.8.

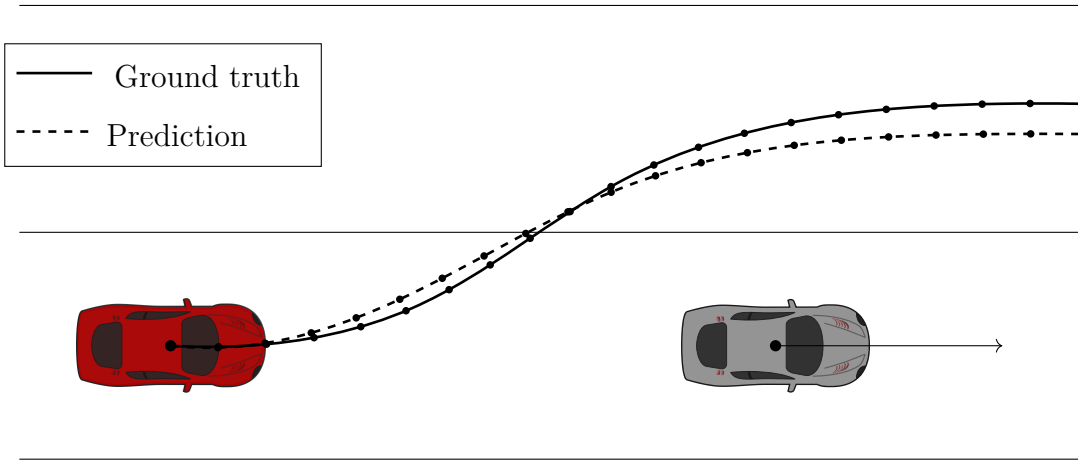


Figure 3.8: Top view of an overtake, comparing the ground truth trajectory and a prediction. Each dot on the lines represents a coordinate point, sampled and predicted with a frequency of 4 Hz.

An LSTM network has the feature that it can be used to model input-output as sequence to sequence. This means that when the sequential input goes through the network, the output can also be given as a sequence. In vehicle trajectory prediction, the input would be given as a sequential data input gathered from the previous history of the ego vehicle and its surroundings. From history, the network will then output a sequence of predicted future positions. The sequential input and output do not necessarily need to be of the same length, which allows for the history and prediction to be of different time intervals. The relationship of the sequence lengths will later be analysed in terms of performance to investigate how time affects the performance of the network model, see section 4.4.

The time interval of the output is decided to be 5 seconds. This is due to that the network analysis includes performance measurements of driver intention, which is based on that the network can predict manoeuvres that are not yet initialised. To set the output time interval is a somewhat subjective decision, but is based on the research of driver behaviour in a highway environment [18]. There the average lane change manoeuvre is about 7 seconds long, and by that, 5 seconds should be sufficient to assure that the network has to predict far into a manoeuvre that is not yet initialised. As the sampling frequency of the data is 4 Hz, the output length is then set to 20.

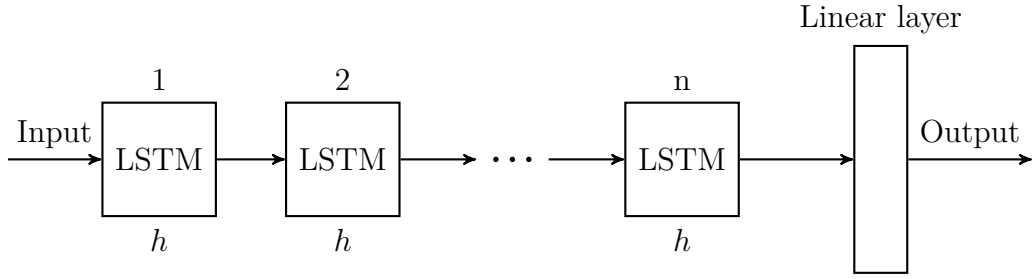


Figure 3.9: Data flow in an LSTM network with n number of cells, hidden dimension h and a linear output layer.

The LSTM network itself is constructed as a sequence of LSTM cells, all sharing the same hidden dimension. Before the output vector, the output of the last LSTM cell is passed through a linear layer which maps the hidden state to the final output in ego-vehicle coordinates. The number of cells and the size of the hidden dimension will later be analysed to find the best performance. Data flow through a network with hidden dimension h and n number of cells can be seen in Fig. 3.9. The size of the linear layer is set by the hidden dimension of the LSTM cells and the output length. With a hidden dimension h and an output length of m, the linear layer will have the size $m \times h$.

3.2.2 Data input

In order to justify the changes done to the network input, a benchmark model is used to explain the fundamental problem with using the raw dataset as input to the network model. The benchmark model will also be used to compare the final performance of the later proposed network model and preprocessing methods.

Benchmark model

In order to compare the results, a benchmark model is used. This model uses the unprocessed data from the dataset, such that performance improvements from the preprocessing also is captured in the results.

The model is a vanilla LSTM model with 2 LSTM cells, where both cells have a hidden dimension of 100.

From the dataset, the data is preprocessed such that information about the road state is either transformed or removed. Starting with the objects on the road, the first input channels to the network is describing objects in front of the vehicle. With one object position described with two values, x and y, two channels corresponds to one object. Although the input to the network then goes up to a certain number of vehicles, the road state is not always exposed to the maximum amount of objects. Thus, the network will need to have a null input to the empty channels at those times, but as the input needs to be scalars one has to choose another value than null.

Instead, the empty channels are set to a distance which the dataset never reaches, $(x, y) = (R_{max}, R_{max})$. Not choosing $(0, 0)$ as the empty value is due to that the network should easily learn that low values are of more importance, as objects very close to the ego vehicle should imply greater importance. Also, an unnecessary large input may infer with the weight scaling of the network.

Through training an LSTM network with the raw dataset as input and then removing the objects in the validation step, the object dependency is analysed. In Fig. 3.10, it is clear that the benchmark network performs similarly both with and without the use of surrounding objects. The conclusion is thus that the network can make accurate predictions with the use of only internal states of the vehicle, such as yaw rate and lateral velocity. A normal driver would certainly care about the current status of the road. Hence, both the yaw rate and the lateral distance to the lane markers are removed as in-data to the network in order to force the network to navigate with respect to other objects.

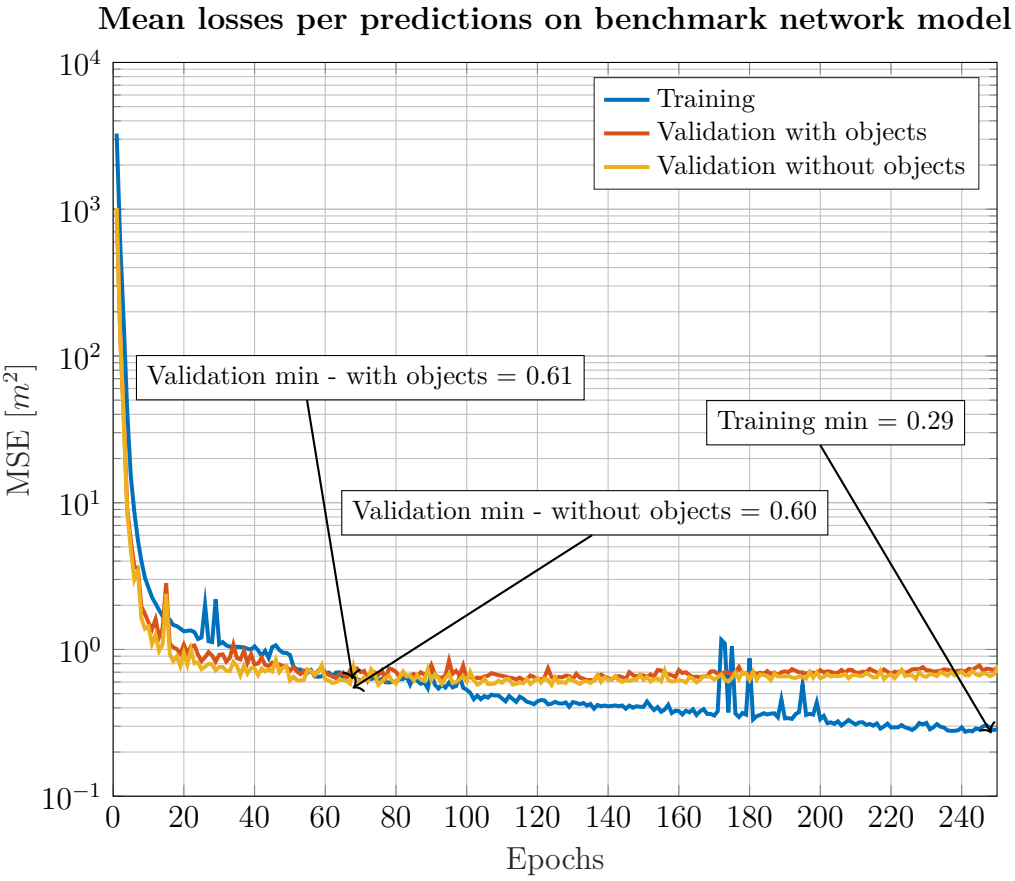


Figure 3.10: Benchmark performance of LSTM network with 2 number of cells and with hidden dimension of 100, trained on raw dataset including the objects using Mean Square Error (MSE) loss function. Showing results both including and excluding objects in the input of the predictions.

In the raw dataset, each lane marker is given as an N-degree polynomial. The

goal is to remove the initial lateral distance from each lane marker relative to the ego vehicle, and that the lane marker input to the network is four x- and y-coordinates for each lane marker. Hence, to remove the lateral velocities, or the lateral offsets, from the input data the polynomials are converted to real-world coordinates. This is done for each sequence, by firstly calculating the closest distance to the lane marker for each time sample, $y = f(0)$, and then calculating global lane marker positions for the whole sequence. As the global vehicle position is also known, the x- and y-coordinates of the lane markers at a set longitudinal distance can be extracted at all vehicle positions. However, the x- and y-positions still carries the information where the ego vehicle is on the road. Thus each lane marker is offset such that the first point of each lane marker is (0, 0), as seen in Fig. 3.11.

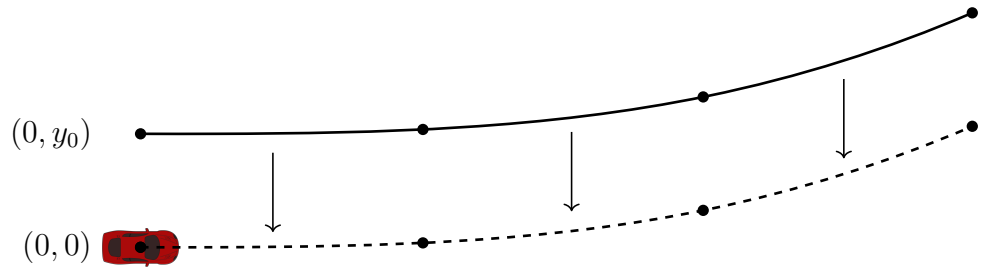


Figure 3.11: The lane markers are adjusted to not have an initial lateral offset relative to the ego vehicle.

The final input will thus not include any lateral offsets or lateral velocity for the ego vehicle, forcing the network to adjust its prediction in regards to surrounding objects' movement. Furthermore, as the global lane marker positions are known, object detections outside of the set road width can be filtered out. To summarise the data input to the network after the preprocessing:

Before preprocessing:	After preprocessing:
• Ego yaw rate • Ego forward velocity • Object positions • Lane marker polynomial	• Ego forward velocity • Object positions • Lane marker positions without lateral offset at first point

Table 3.2: Summary of the categorisation in the input channels to the network.

$x_1 - x_n$	$x_{n+1} - x_{2n}$	x_{2n+1}	$x_{2n+2} - x_{end}$
Object x-positions	Object y-positions	Velocity	Lane markers

A summary of the input channels after the preprocessing is presented in Tab. 3.2. Overall of the input channels, a subset of them are considered object channels. There are *2n* object channels which can handle n objects at once, as one position

takes two coordinates. All channels are coupled such that x- and y-positions of a single object is always related through the input indices. Throughout the dataset, objects are fluctuating between different object channels, as no tracking has been implemented to keep the same objects in a single pair of channels. In order for the network to learn the object channels more easily, a simple tracking algorithm is applied to all objects. If the objects are changing within a distance ϵ between two time samples, those two positions are considered the same object and set to be in the same input channel.

$$\sqrt{x^2 + y^2} < \epsilon \tag{3.2}$$

The tracking in Eq. (3.2) is then applied to the whole dataset.

3.2.3 Network training

When training the network, the weights are updated as the network is encountering data. That means that with every prediction in the training, the weights are directly updated with respect to the outcome. It is important to note that this is not done in the validation step, as it would let the network cheat in its validation metrics. The learning rate is updated at various epochs in the training, to let the network converge smoothly and fast to its optimum. At the set epochs, the learning rate is scaled by a certain factor. In all of the training, *Adam* is chosen as optimisation algorithm and the loss is chosen to be the Mean Square Error (MSE) between the ground truth trajectory points and the predicted points. All parameters set when training the network is summarised in Tab. 3.3.

Table 3.3: Summary of learning parameters used to train the network.

Parameters	Values
Epochs	500
Optimiser	Adam
Initial learning rate	0.001
Learning rate update epochs	[50, 100, 200, 400]
Learning rate update factor	0.5

A potential problem with the object channels is that certain channels may be exposed to different scenarios, which are not certain to be in the same channel in the future. The problem is fixed by at each epoch, object channels are shuffled such that all channels are exposed to all object inputs. The results of the network training are presented in section 4.1.

3.3 Performance measures

When constructing any kind of network, the evaluation methods are crucial in determining if the model behaved as intended or not. To begin with, a differentiable objective function, or loss function, is the backbone of the network's performance evaluation. It is natural for the network to adjust towards the loss function as it is the measurement that it uses to adjusts its weights, hence the need for differentiability.

However, sometimes it is not enough to evaluate a network's performance by only looking at one number. Manoeuvring a vehicle puts multiple crucial actions on the driver, that when done poorly may impede the safety in traffic severely. As unintended actions may compromise the safety on the road, it is vital to test the algorithm further. Thus, the network is evaluated with several performance measurements, which are presented and elaborated upon in this section.

3.3.1 Driver intention

Evaluating if the network knows when to do actions is done by analysing the performance of lane changes. The network predicts the future trajectory from a time point where the lane change manoeuvre has not yet been initialised, and as the ground truth data points are known the quality of the prediction can then be calculated. This performance number is then compared to the same performance measure when driving straight. Thus, there is a comparison between how the performance of lane changes and straight driving differentiates from one another.

For the performance measure to function properly, the lane change manoeuvre is not yet to be initialised when the network predicts its trajectory. The classification of a lane change is thus defined as when the last ground truth trajectory point is on another lane than the second last. In other words, if the car crosses the lane marker in its last time step of the ground truth trajectory, seen in Fig. 3.12. Cases not classified as lane changes are considered straightway driving.

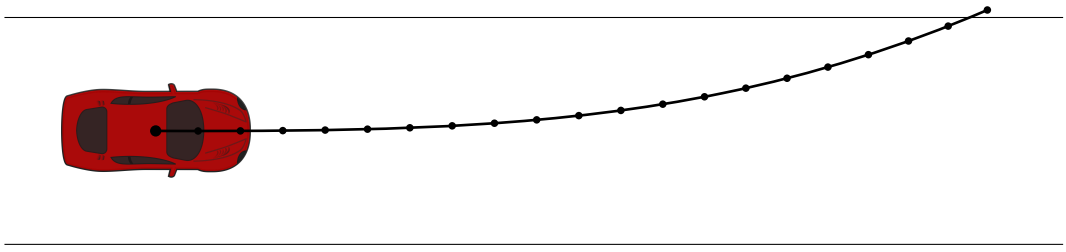


Figure 3.12: Example of a lane change scenario. Note that the last and second last ground truth data point are in different lanes.

The formula for the performance measure calculates the absolute distance between the last point in the prediction and the last point of the ground truth tra-

jectory. If (x, y) is the last ground truth data point and $(\hat{x}, \hat{y})$ is the last predicted data point, then the lane change performances is given by Eq. (3.3)-(3.5).

$$l = ||(x, y) - (\hat{x}, \hat{y})||_2 \tag{3.3}$$
$$l_x = ||x - \hat{x}|| \tag{3.4}$$
$$l_y = ||y - \hat{y}|| \tag{3.5}$$

When driving on a highway, and as seen in the dataset, the forward velocity is often over 100 km/h. The forward velocity is thus much larger compared to the lateral velocity the ego vehicle will reach, for example in a lane change. It is therefore logical that the error also should be much larger longitudinally versus laterally. Thus, individual errors are of interest as well. **In the performance analysis, both the longitudinal and lateral errors will be investigated to make a conclusion about the model's range of error.** An important note is also that an increase in lateral error is often more significant in regards of safety than the same increase in longitudinal error, as a lateral difference could mean that the ego vehicle is in a different lane.

3.3.2 Safety evaluation

To make an assessment of the safety aspect of the driver model, the output of the network is analysed to verify that there are no collisions in the prediction. The distances from surrounding vehicles will therefore be calculated in regards to the predicted trajectory instead of the ground truth trajectory, for each prediction separately. Thus, following the predicted trajectory tells how close the model is manoeuvring around the other vehicles on the road if it would take control itself. The relative object positions are therefore adjusted to match the object distances in the vehicle frame if the vehicle would follow the predicted trajectory. If another vehicle, at time steps k, has a relative position to ego vehicle of (x_k, y_k), the difference between the ground truth and the prediction, $\Delta(x_k, y_k)$, will be adjusted in the relative positions of the surrounding objects.

$$(\hat{x}_k, \hat{y}_k) = (x_k, y_k) - \Delta(x_k, y_k) \tag{3.6}$$

The new object positions from Eq. 3.6, $(\hat{x}_k, \hat{y}_k)$, can then tell if the model is predicting a trajectory close to already existing objects, or if it will collide. Creating a plot with the results opens up for the opportunity to see any differences in the behaviour of the ground truth data and the predicted trajectories, see section 4.6.

3.3.3 Prediction fluctuation

Now, the robustness for the predictions is analysed by investigating the spatial fluctuations for each subsequent prediction. For each prediction, there will be some difference between the current prediction and the one in the next time step. As all coordinates are relative to the ego vehicle, the distance between the predictions will not be affected by that the vehicle is moving from sample to sample. Rather the predictions will change based on the environmental change around the vehicle, i.e change in surrounding objects and lane markers. For each prediction, $p = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, the fluctuation is the distance visualised in Fig. 3.13.

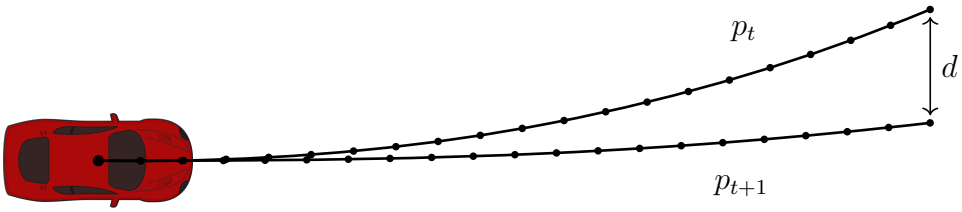


Figure 3.13: Visualisation of the fluctuations between two following predictions. The distance d can also be divided in its lateral and longitudinal distances to further describe the fluctuations.

As each time prediction has a sampling frequency of 4 Hz, the distance between each prediction can be expressed as a velocity, v_p. Assume f as the sampling frequency, then the velocity between each sample can be calculated like the following.

$$v_p = d \cdot f \tag{3.7}$$

Eq. (3.7) is then applied to all predictions in all sequences. The prediction velocity is calculated for all predictions that have a temporal relationship to each other. Consequentially, there will not be any spikes in the velocity between different data log sequences. These results are presented in section 4.7

4

Results

To revisit the question if it is possible for a network to learn how to drive a car, the proposed network model is evaluated on its performance measures in order to make a conclusion about its driving abilities. The key features of interest are analysed to make an effort in answering the question if the network, in fact, can manoeuvre a vehicle. Furthermore, the characteristics of the network will be thoroughly analysed to give a perspective of how well the network performs in various environments.

Firstly, the most general performance of the network, the loss, is evaluated by analysing the Mean Square Error (MSE) of the predictions. This is to show how well the network model can predict the ground truth data, by calculating the average error between all data points in the prediction. Secondly, there will be an investigation of the object dependencies of the network. In other words, show how the surrounding objects around the vehicle affect the network performance. Thirdly, making a study of how the characteristics of the performance measures change in regards to how far into the future the prediction is made.

Throughout the results, the longitudinal direction is regarded as the x-axis of the ego vehicle body frame, and the lateral direction is along the y-axis.

4.1 Network performance and training progress

Tuning an LSTM-network with different parameters includes seeking a balance between the ability to learn complex scenarios without risking to overfit the model to the data. Increasing the complexity, or expressiveness, of the model often means increasing the hidden dimension of the cells and/or increasing the number of cells. A drawback of increasing the model complexity is an increase in computation time due to more trainable parameters, but once again, the aim is not to optimise the model in regards to computation time. Rather, the focus lies in finding a model which is able to mimic human behaviour. With that said, the validation loss of various LSTM model sizes is presented in Tab. 4.1. All models have in common that all cells share the same hidden dimension size in each separate model.

Hidden dimensions \ Number of cells	1	2	3	4	5
25	0.553	0.415	0.425	0.409	0.427
50	0.432	0.425	0.409	0.422	0.43
75	0.441	0.413	0.427	0.428	0.405
100	0.437	0.418	0.434	0.423	0.432
150	0.41	0.417	0.448	0.433	0.425
200	0.398	0.423	0.422	0.432	0.434
300	0.419	0.411	0.424	0.441	0.448

Table 4.1: The lowest mean square error of an epoch evaluated on the validation set, trained for 500 epochs with different number of cells and hidden dimensions. Note that all cells share the same hidden dimension size in each separate model.

Hereby, the future results will be based on the network with 300 hidden dimensions and 3 cells. The decision is based on that the model performs marginally equal to the others, but with higher complexity. The model could thus scale better with more training data. When increasing the size of the dataset, the results from the network may thus improve. This way, the model analysed is also ready to be trained with more data than currently available.

After the data-processing been executed and the network model has been trained a comparison with the benchmark network is made. Fig. 4.1 shows the training progress for both the chosen network and the benchmark. As can be seen, the proposed model performs marginally better than the benchmark.

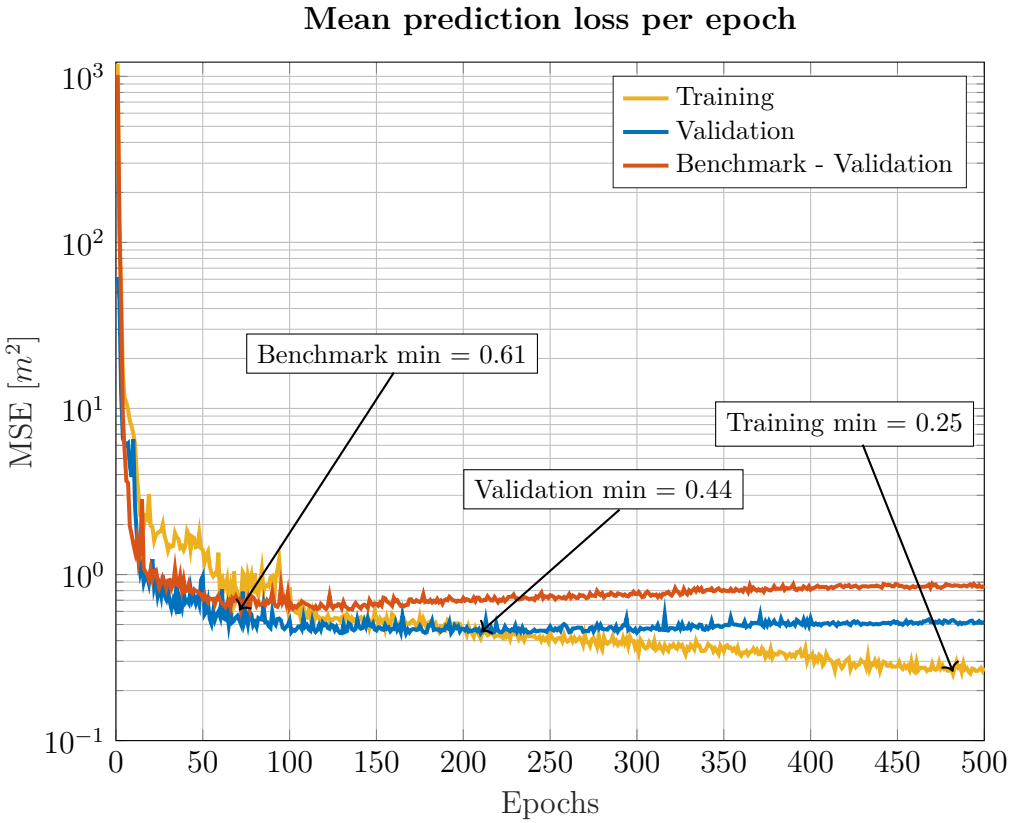


Figure 4.1: The training progress of the network over number of epochs for the training and validation dataset, compared with the the benchmark model.

From Fig. 4.1, the model is overfitting the training data and can be seen when the training loss goes down while the validation loss increases. This is caused by the over-exposure of the training data, and as the network only can measure its own performance on the training data the network gets more and more specialised in its skills. But as the network gets more skilled on the training data, it also trades off its generalisation abilities of the new data presented in the validation set. **As this is an unwanted behaviour, the best performing model on the validation set is saved for further analysis.**

4.2 Object dependence in model

When driving a car, a human's choices are largely affected by surrounding vehicles. You have to compromise with others about the drivable space on the road. Mimicking human behaviour thus calls for the objects to have an importance in the trajectory prediction. To analyse if the surrounding objects are important to achieve a low loss, the mean losses for each epoch are presented in Fig. 4.2. The figure shows the mean losses of the network when training with objects, and then excluding them in the validation predictions. Remember from the preprocessing, a non-existing ob-

ject is set to the coordinate (R_{max}, R_{max}). It is clear that without the objects, the network is not able to predict the next movement at all. However, this does not imply that one must have surrounding objects to achieve low loss, rather, it tells that the network *cares* about the objects in the predictions. That means the object data is not superfluous for the network when learning to drive like a normal driver. The opposite can be observed in the benchmark model, where the objects do not make any difference in the training progress, and are almost completely redundant as an input.

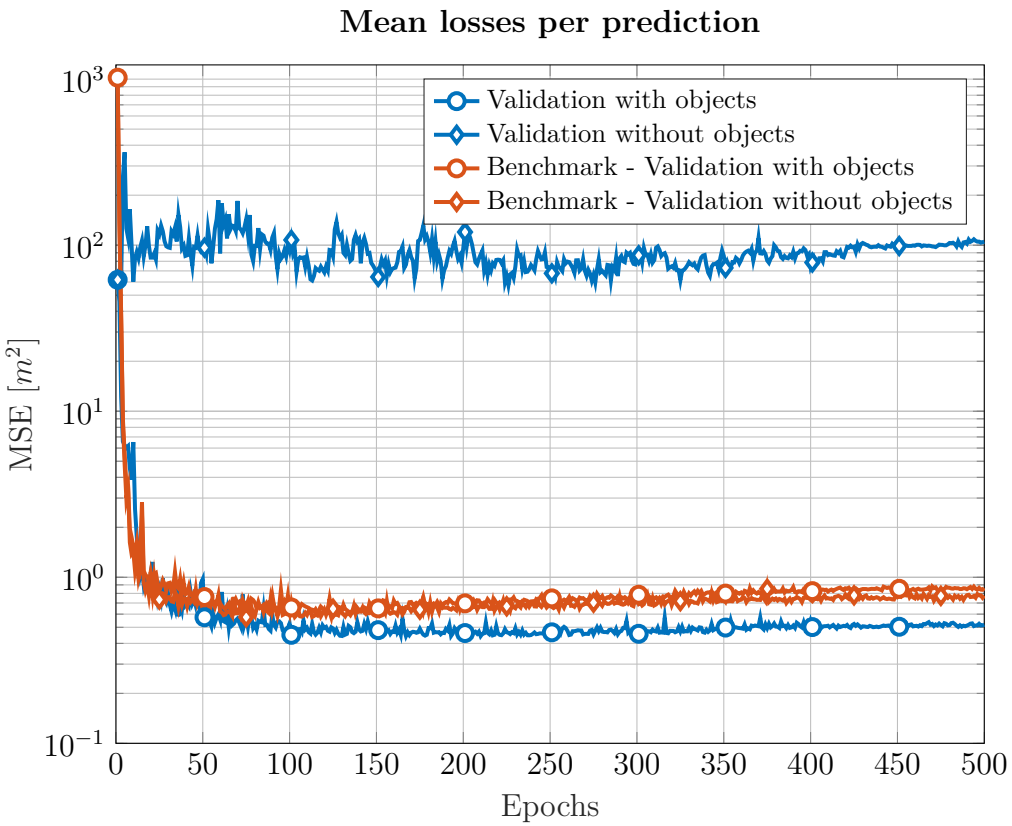


Figure 4.2: Mean losses per prediction after each epoch for the training set and the validation set with and without objects as an input to the network. The figure highlights that the objects are necessary to predict the next movement. When training the network without any objects, the network fails to predict the next movement.

To confirm the importance of objects, an analysis of how the number of objects affects the predictions is necessary. Indeed, the complexity of a traffic scenario increases with more objects. It is therefore interesting to analyse if scenarios with more objects increase the average loss, or if no surrounding objects make the network useless, as could be implied from Fig. 4.2. Fig. 4.3 present a violin plot where the loss is compared to the object groups. A violin plot is very similar to a box plot, but the data point distribution is also visualised. The conclusion is that the mean loss

does not increase significantly between scenarios with zero objects and object groups g_{1-5}. Note, the average loss gets higher with object groups g_6 and g_7. However, due to the lack of scenarios with those object groups, it is understandable that so is the case. It is therefore a risk that the dataset is not well exposed to those object groups. One can also observe spikes (outliers) for all scenarios but it is not possible to see any correlation between when the spikes occur and the object groups, other than that the max error seems to be lower for when no other objects are present.

Thus, it is confirmed that the proposed network model consider surrounding objects in its predictions, while not depending on their presence. It implies that the normal driver changes his/her behaviour when other objects are around and that the network model can identify that.

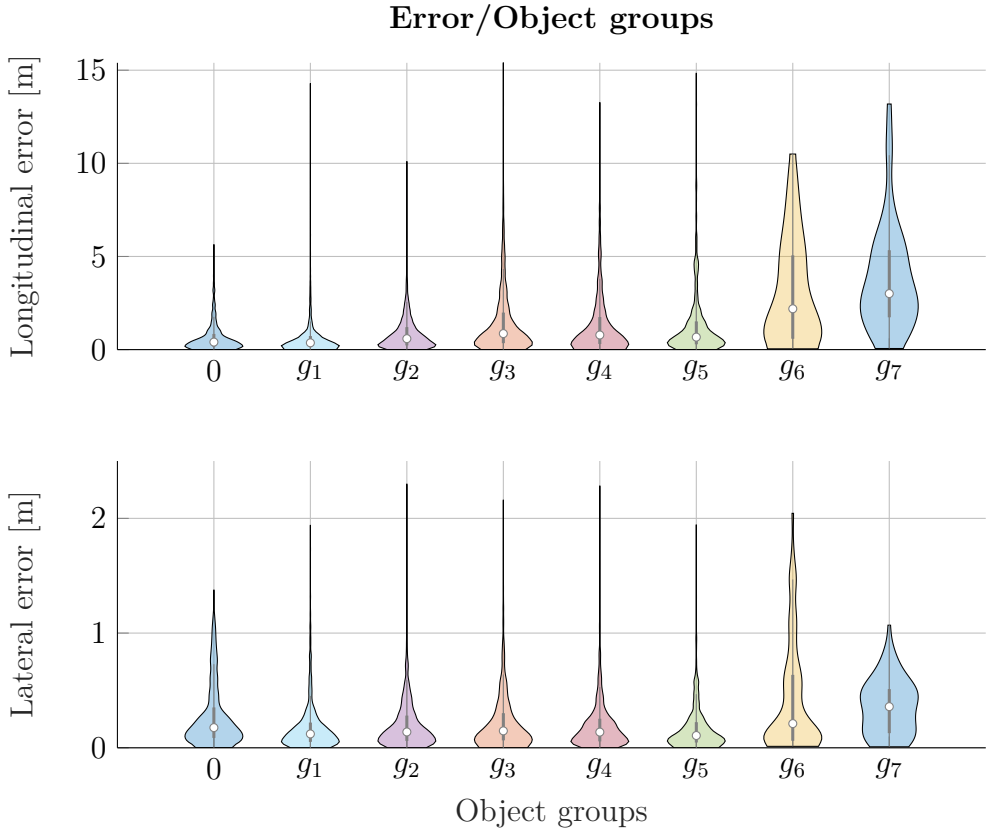


Figure 4.3: The density of data points with the error after 5 seconds into the future in relation to the object group in the scenario. Each point inside the distribution represents a data point with the white circle being the mean error. Observe from Fig. 3.4 the exposure of all object groups. The presence of cases with g_{8-10} is considered negligible and it can also be seen that g_6 and g_7 has lower exposure than the rest.

Furthermore, based on Fig. 4.4 closer objects leads to worse predictions. The behaviour could be explained by the increasing level of compromise about the drivable

surface around the vehicle when another object gets closer. As a tangent from the results in Fig. 4.4, the network may reflect the driver in the sense that close objects are of greater importance and that the driver changes his/her behaviour according to those objects. Note that the errors presented in the figures are the error of the predictions after 5 seconds, not the mean error over each prediction.

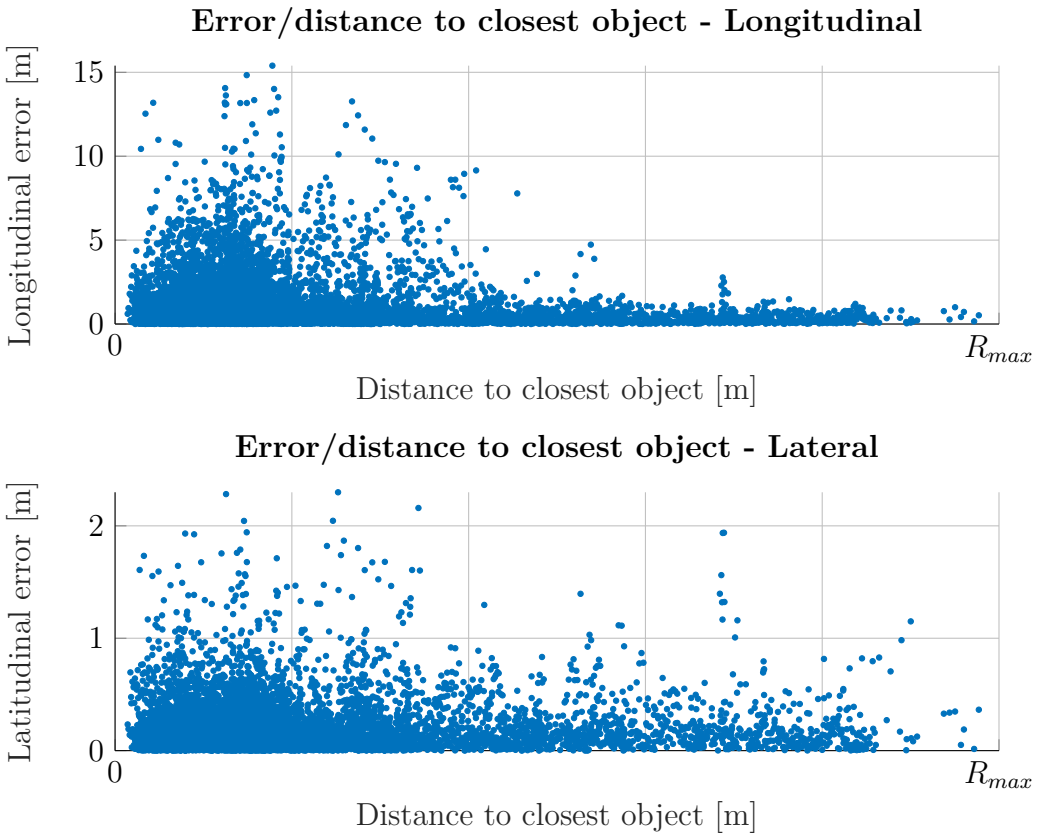


Figure 4.4: Comparing the closest distance to surrounding objects versus the error, 5 seconds into the future. Each point represent a single prediction in the validation set.

It can be seen in both Fig. 4.3 and 4.4 that there are some troublesome outliers present where the error is large compared to the mean error. A huge error in the prediction could cause a formidable accident in traffic. Thus, one must highlight when these worst-case predictions occur, and if it is possible to prevent them. However, the large error values often come from the same scenario in a sequence of 1-2 seconds. These cases have been individually analysed and the error stems from cases where another vehicle acts with high unpredictability. For example, cases with cut-ins or fast deceleration. When analysing the video feed from the specific sequences, it is humanly impossible to predict the movement of the object that causes the error. As the normal driver would not be able to predict this movement pattern in surrounding vehicles, it is possible that these cases are not learnable for the network given the data input.

4.3 Feature correlations in prediction performance

The correlation between longitudinal and lateral error to the velocity of the ego car is presented in Fig. 4.5. No obvious correlation can be observed, and it is not clear to draw any conclusions that the current velocity would affect the quality of the predictions. Note, since the distribution of the number of events where the ego car drives around 120 km/h are larger, it is probable that there also should be more cases with large error in that velocity interval. Once again, the errors presented in the figures are the error after 5 seconds into the future in the predictions.

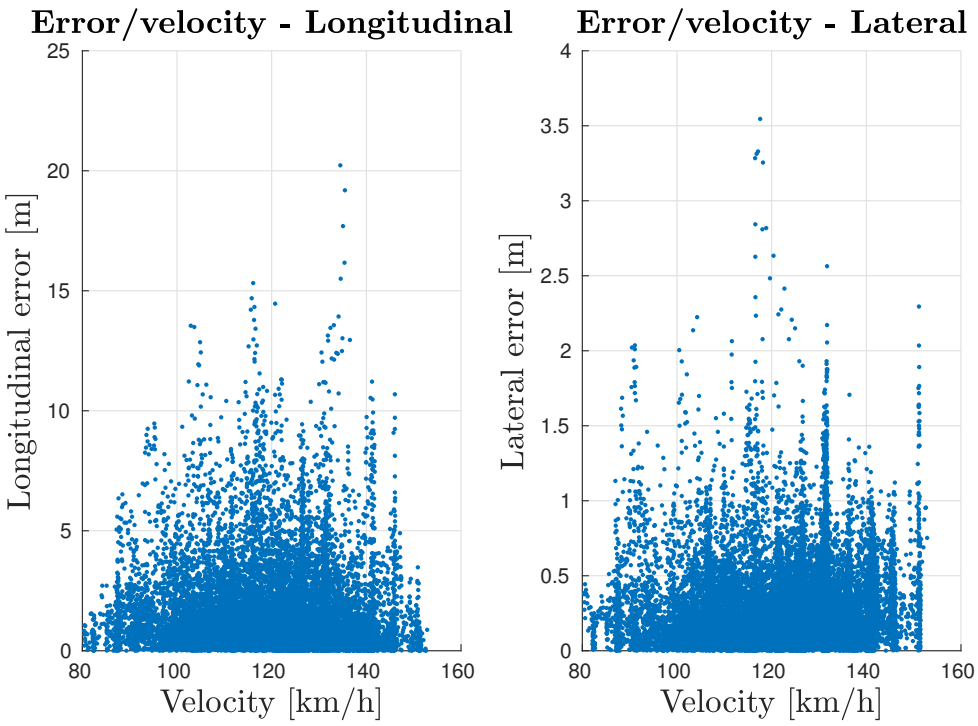


Figure 4.5: The longitudinal respective the lateral error in relation to the current velocity of the ego-car. Each point represents a single prediction in the validation set.

Furthermore, the correlation between the curvature of the road and the error is presented in Fig. 4.6. This to analyse if the error correlates with the steepness of the curves.

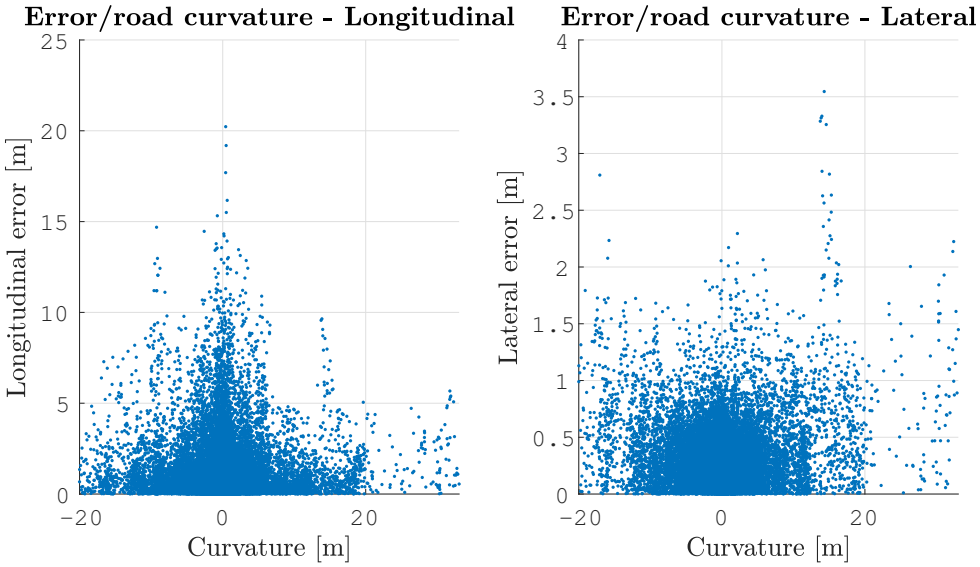


Figure 4.6: The longitudinal and lateral error in relation to the curvature of the road. Each point represents a single prediction in the validation set. The curvature is defined as the lateral distance between the first lane marker point and the last lane marker point given as input to the network.

Seen in the left plot in Fig. 4.6, the loss actually has a tendency to decrease with greater steepness of the curve. A cause for this could be that large errors occur in situations where other vehicles intrude with the planned path. However, other vehicles may be less likely to make any actions in a curve.

4.4 Output sequence length

Predicting the future trajectory has to this point been limited to predicting 5 seconds into the future, or 20 samples forward with a sampling frequency of 4 Hz. As this limit is set by the need of analysing the network's ability to predict the driver's intention, it is not necessarily needed to always plan the trajectory 5 seconds ahead. However, with a smaller time horizon, the performance will not be eligible for the same analysis previously presented. But it is still interesting to investigate the performance if the time horizon is changed, while keeping the input history of 5 seconds. As it is harder to predict further into the future, it is obvious that the error also will follow that trend. But, characterising the error curve for various time horizons is still important to verify that the error is smaller when predicting not as far in time. Also, when implementing the algorithm you may want to update the prediction more often than once every 5 seconds.

Fig. 4.7 and Fig. 4.8 represent the longitudinal and the lateral error respectively, between ground truth and the predictions in time for different percentiles. And as previously mentioned, the error is in fact increasing together with the output length, i.e. it's more difficult for the network to predict far ahead in the future.

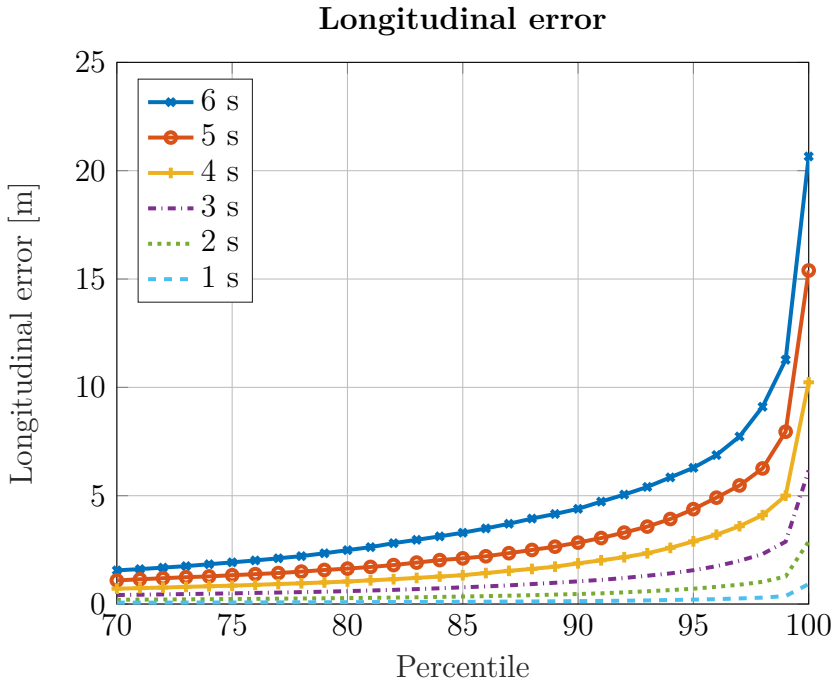


Figure 4.7: Longitudinal error between the prediction and the ground truth data, at the last point of the predictions. The error tends to be higher the further into the future one predicts. All errors are evaluated on the validation set.

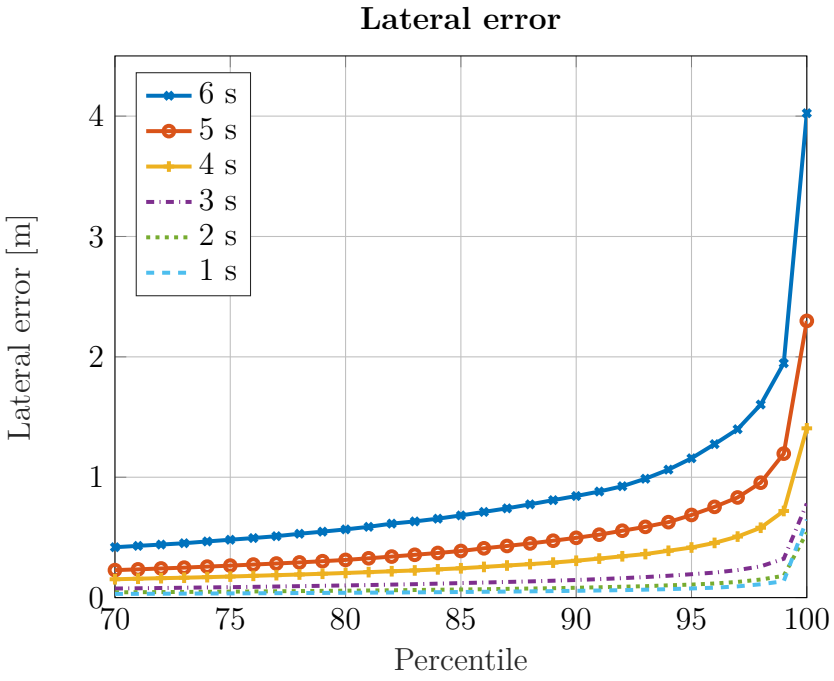


Figure 4.8: Lateral error between the prediction and the ground truth data, at the last point of the predictions. The error tends to be higher the further into the future one predicts. All errors are evaluated on the validation set.

Remember that during each prediction, the network is not updated with any new information. And as the predictions are made seconds in advance, the reason for a large error is possibly because the traffic situation on the road changes substantially within these seconds. At least, substantially enough that the ego driver needs to react accordingly, inferring an error in the previous prediction. The previous prediction may be justified for that specific traffic situation, but as the situation changes it may become infeasible.

As the driver intention cannot be analysed through the previous means with a lower time frame in the prediction, an analysis of a model that has been trained for predicting 5 seconds ahead in needed. Thus in Fig. 4.9, the network has been trained to predict 5 seconds in the future, or 20 samples ahead. The error in each time sample is then analysed to give the performance of the model over time. One can observe that the error increases exponentially over time and that an implementation of the model may benefit from rapidly updating the prediction when driving.

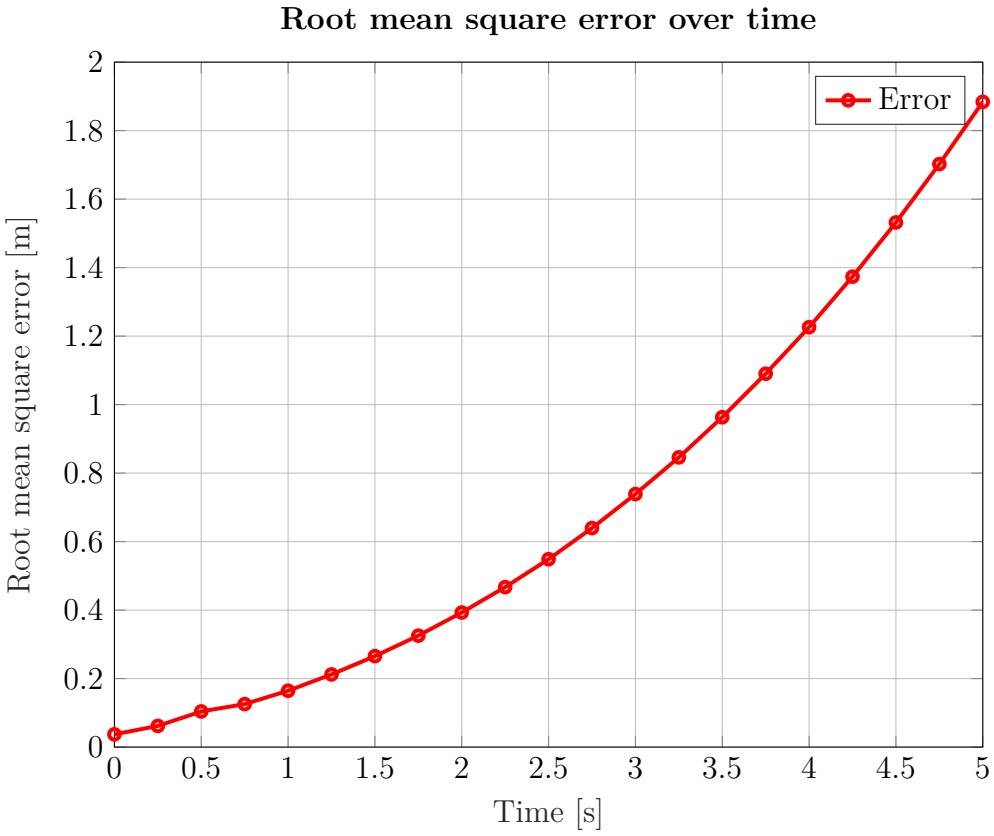


Figure 4.9: Root mean square error of the model output from a network model trained to predict the trajectory 5 seconds into the future and with a sample rate of 4 Hz. The error is the total error in the specific time instance, not coupled with the other time instances. The errors are evaluated on the validation set.

4.5 Lane change predictions

An important quality within the network is the ability to mimic a real driver's ability to make decisions in different scenarios, and to some extent, predict the future of the traffic flow. So far, the analysis has indicated that it is, to some point into the future, possible to perform highway driving in the set environment. Anyway, a distinct difference between a real driver and a network is its ability to know *how* to make a lane change and *when* to do it. This calls to analyse if the network knows when to perform a lane change.

Fig. 4.10 shows the error for lane change driving as well as straightway driving, represented as histograms. The error is evaluated for predictions five seconds in the future and the method used to calculate the error is covered in section 3.3.1.

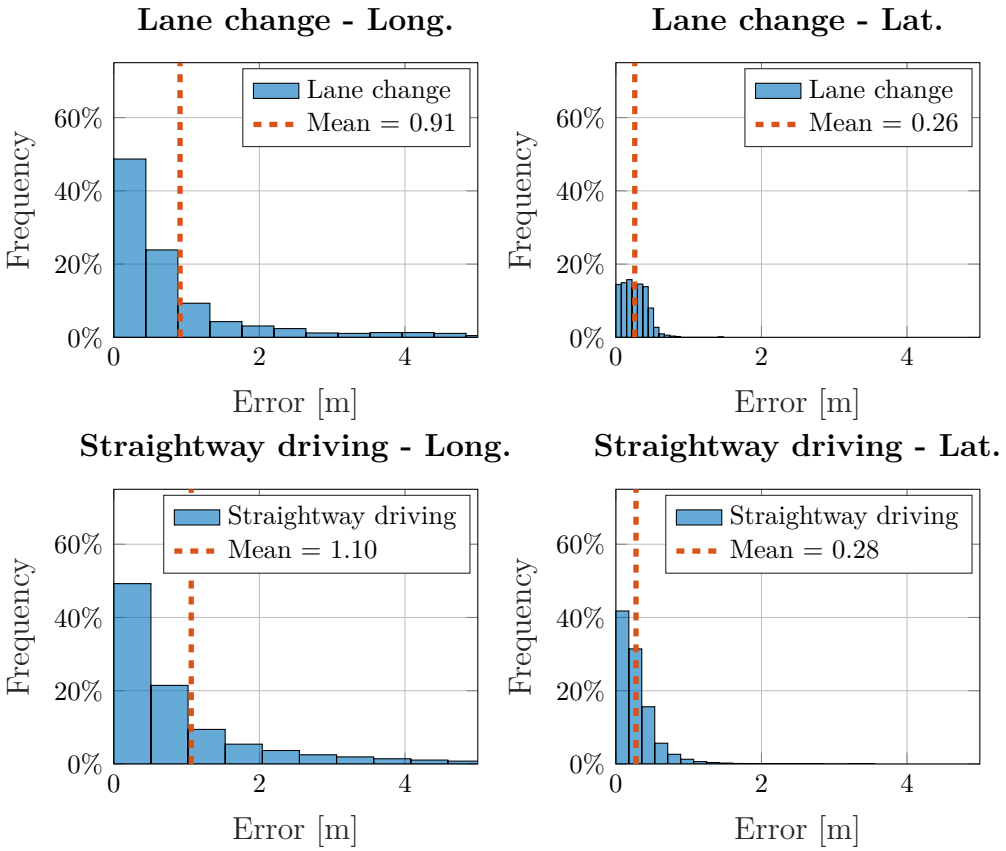


Figure 4.10: Histograms over the longitudinal and lateral error respectively, in scenarios with and without lane changes, using the proposed model.

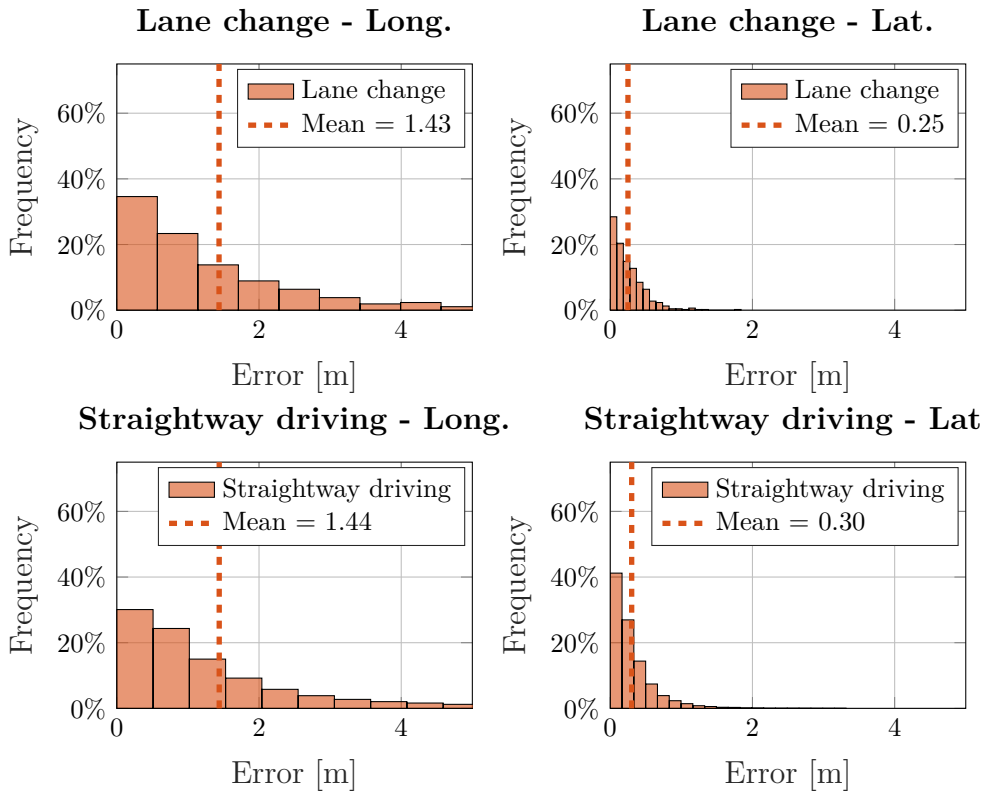


Figure 4.11: Histograms over the longitudinal and lateral error respectively, in scenarios with and without lane changes, using the benchmark model.

From Fig. 4.10, the proposed network model actually performs marginally both longitudinally and laterally in lane changes compared to straightway driving. As previously stated, predicting a lane change 5 seconds in the future means that the lane change is not yet initialised at the moment of prediction. Thus, the network shows its ableness of knowing *when* to make a lane change. Also, when comparing the lane change performance with the benchmark model, the proposed model performs better in almost all measures. With the exception of the lateral error when changing lanes, where the models perform near equally, the proposed model is outperforming the benchmark, in Figure 4.11.

The important note here is that the model performs better in the lane changes than the straightway driving. If we can conclude from the network training progress, in section 4.1, that we are able to mimic the human driving with high precision in the general case - then we are also able to identify that the model can predict future lane changes as well as, if not better, the straightway driving. Thus, with that conclusion, the model is able to mimic the driver intention from the data logs.

4.6 Safety assessment of predicted trajectory

From the model output, the predicted trajectory is compared to the ground truth target trajectory in each prediction separately. If then, the vehicle would follow the predicted path, would it manoeuvre into or close to another object? The comparison is done by calculating the new relative object positions from the predicted trajectory instead of following the ground truth, by the methodology presented in section 3.3.2. This is to verify that the model does not make dangerous manoeuvres that may compromise the safety in the road environment. From the error distributions presented in section 4.3, large deviations from the ground truth position may infer a security risk.

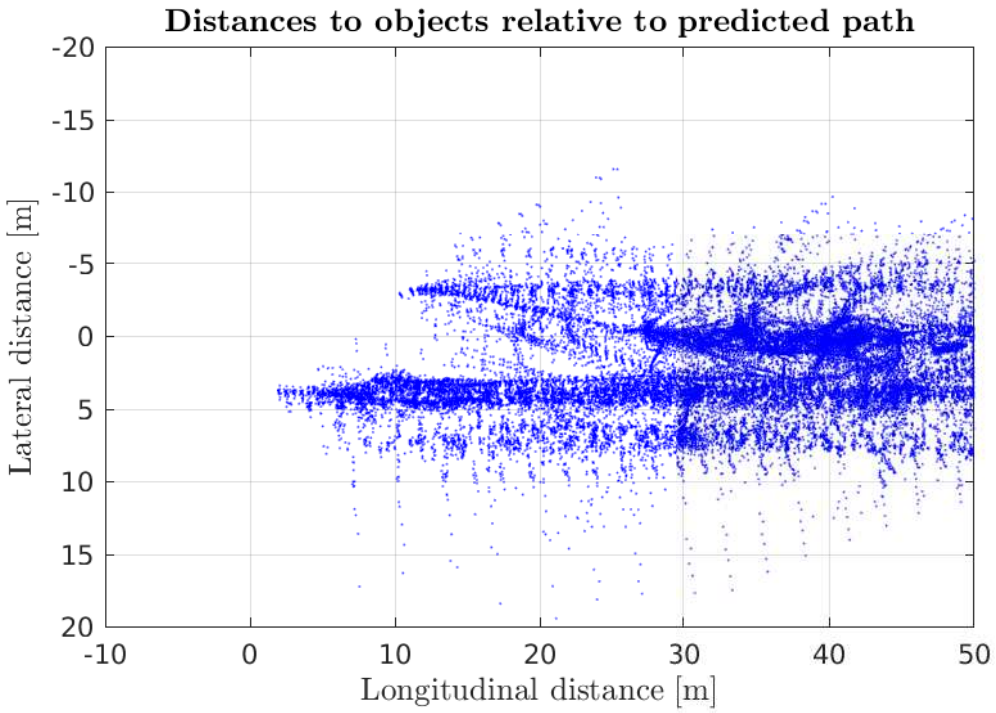


Figure 4.12: Relative object position data points in the body frame of the ego vehicle, when following the predicted trajectory from the network model. Each point represents an object position relative to ego car (0,0) at some time in the entire validation set. A point is thus only the object detection point with no spatial information other than the point position.

In Fig. 4.12, the ego-car is placed in the origin (0,0), and all surrounding objects are expressed relative to this point and in the ego vehicle body frame. When analysing the relative object positions the closest object longitudinally to the ego

vehicle is 7.5 meters away and the closest laterally is 3.5 meters away, thus no direct collisions with surrounding objects. This shows that while the network still has some predictions with large error, there are no collisions in those predictions.

4.7 Evaluation of prediction fluctuation

To investigate the behaviour of the predictions in regards to time, the change between each subsequent sample is calculated. This to see if the predictions tend to fluctuate a lot and to be able to tell something about the robustness of the predictions. The fluctuation distance is calculated from the last point in the 5 second prediction, by the method presented in section 3.3.3.

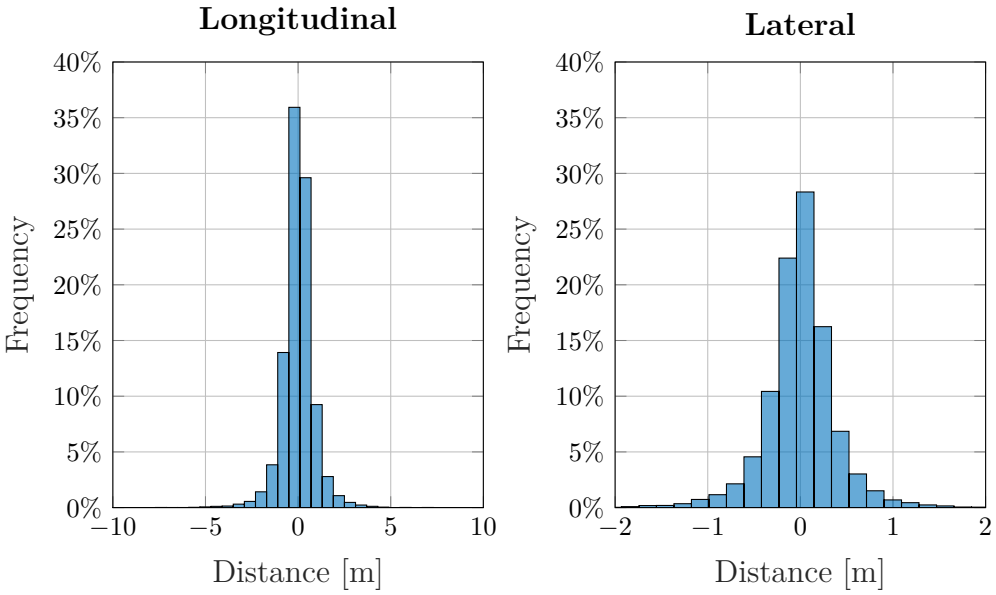


Figure 4.13: The fluctuation of the predictions, given both longitudinally and laterally. The distance of the fluctuations are here presented in relation to the frequency of the distances represented when predicting throughout the validation set.

The fluctuations shown in Fig. 4.13 is an interesting part of the behavioural outcome of the network model. If a large fluctuation in the predictions is a downside or not is not obvious to answer. As previously seen in the error distribution, there are a few sequences that produce a large error compared to others. The conclusion from these sequences was that the large errors were caused by other vehicles on the road acting with high unpredictability. Even though there was a large error for a small period of time, the network model was then able to properly assess the traffic situation and predict accordingly. What stems from this observation is that a high fluctuation in the predictions is not explicitly a bad property. It could also mean

that the model is able to quickly adapt to an unpredictable situation, just like a normal driver would.

5

Conclusion

This study presents a data-driven approach using neural networks for modelling normal driving behaviour on a multi-lane highway environment. This driver model was evaluated on decided performance measures, to conclude that the model reflected the innate behaviours that are present in human driving. In other words, mimic the driving in data logs gathered on expeditions in various places over the world.

The chosen network model made use of Long Short-Term Memory (LSTM) cells, structured as a sequence-to-sequence model, in order to make a trajectory prediction of the future data points. To make these predictions, the model used surrounding objects' relative positions, ego velocity, and road lane marker positions. With the help of these data features, the network model was able to make predictions about the future trajectory mostly with high precision. Further, the behaviour of the network was analysed to verify the intended performance. For instance, the model was tested to investigate the object importance in the predictions. It was shown that the network model made use of surrounding objects when these were present and that it never predicted the trajectory into a collision. A comprehensive study of correlations between prediction error and data features was made to investigate any deficiencies or cases that the network may have trouble dealing with.

Analysing the network model's ability to predict the driver's intention in traffic stretched to evaluate if the network model was able to predict a lane change before it was initiated. The conclusion was that, when comparing lane change performance with performance in straightway driving, the network model could properly reflect the driver's intention of making future actions.

The conclusion is thus that the proposed network model can predict the future trajectory of a human-driven vehicle with a low error range, and that it can to some extent also predict the driver's intention regarding lane changes.

6

Future work

To progress towards the goal of modelling the normal driver, this chapter highlights ideas and further extensions to aim this purpose.

Dataset

Future works should expand the dataset to include other kinds of events and environments. This could be driving in city traffic, country roads or including a wider range of velocities of the ego-vehicle. Also, expanding the dataset to 360-degree vision would be beneficial for capturing the whole vision of the driver.

Network architecture

This research used an LSTM-approach to find features in the dataset. Future work could investigate another kind of network structures or any combination of other methods, like the ones mentioned in related works.

To train networks of this complexity and with this amount of data, takes a long time. Thus, future work could investigate how to train LSTM-networks in a more efficient way. This could be data structure analysis as well as optimising the input structure to the network.

Also, since general driving is not just one correct answer, an interesting extension of the network model would be a statistical approach to the network output. The network output would then reflect the various choices the driver is faced with when driving.

Closed-loop analysis

So far, the network has been tested and evaluated open-loop, meaning that the network has always been updated with the ground truth after each prediction. An interesting future work would be to implement the network in a closed-loop environment. The driver model could then further analysed.

Bibliography

[1] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks." [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[2] X. B. Peng and G. Berseth, "DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning," *ACM Trans. Graph*, vol. 36, p. 41, 2017. [Online]. Available: https://doi.org/http://dx.doi.org/10.1145/3072959.3073602.

[3] I. H. Kim, J. H. Bong, J. Park, and S. Park, "Prediction of drivers intention of lane change by augmenting sensor information using machine learning techniques," *Sensors (Switzerland)*, 2017.

[4] M. Manoj krishna, M. Neelima, M. Harshali, and M. Venu Gopala Rao, "Image classification using Deep learning," *International Journal of Engineering & Technology*, vol. 7, no. 2.7, p. 614, 3 2018. [Online]. Available: https://www.sciencepubco.com/index.php/ijet/article/view/10892

[5] Z. Wang, J. Merel, S. Reed, G. Wayne, N. De Freitas, and N. H. Deepmind, "Robust Imitation of Diverse Behaviors," Tech. Rep. [Online]. Available: https://arxiv.org/pdf/1707.02747.pdf

[6] K. M. Kumar, H. Kandala, and N. S. Reddy, "Synthesizing and Imitating Handwriting Using Deep Recurrent Neural Networks and Mixture Density Networks," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 7 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8493843/

[7] Y. Lee, T. Kim, and S.-Y. Lee, "Voice Imitating Text-to-Speech Neural Networks," Tech. Rep. [Online]. Available: https://arxiv.org/pdf/1806.00927.pdf

[8] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 4363–4369.

[9] N. Ye, Y. Zhang, and R. Wang, "Vehicle trajectory prediction based on hidden Markov model," *KSII Transactions on Internet and Information Systems*, vol. 10, no. 7, pp. 3150–3170, 7 2016.

[10] M. Wöllmer, B. Schuller, F. Eyben, and G. Rigoll, "Combining long short-term memory and dynamic bayesian networks for incremental emotion-sensitive ar-

tificial listening," *IEEE Journal on Selected Topics in Signal Processing*, vol. 4, no. 5, pp. 867–881, 10 2010.

[11] F. Altché and A. de La Fortelle, "An LSTM Network for Highway Trajectory Prediction," 1 2018. [Online]. Available: http://arxiv.org/abs/1801.07962

[12] A. Bükk and R. Johansson, "Private Communication," 2019.

[13] B. Hanin, "Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations," 8 2017. [Online]. Available: http://arxiv.org/abs/1708.02691

[14] I. Goodfellow, Y. Begio, and A. Courville, "Deep Learning." [Online]. Available: http://www.deeplearningbook.org/

[15] C. Olah, "Understanding LSTM Networks." [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[16] B. Mehlig, "Artificial Neural Networks," Tech. Rep., 2019. [Online]. Available: https://arxiv.org/pdf/1901.05639.pdf

[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Tech. Rep. 8, 1997. [Online]. Available: http://www7.informatik.tu-muenchen.de/~hochreithttp://www.idsia.ch/~juergen

[18] G. Li, S. E. Li, L. Jia, W. Wang, B. Cheng, and F. Chen, "Driving Maneuvers Analysis Using Naturalistic Highway Driving Data," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2015-October. Institute of Electrical and Electronics Engineers Inc., 10 2015, pp. 1761–1766.