CHALMERS
UNIVERSITY OF TECHNOLOGY

Predicting Vehicle Motion and Driver Intent using Deep Learning

Master's thesis in Complex Adaptive Systems

JOAKIM ANDERSSON

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:EX045

Predicting Vehicle Motion and Driver Intent using Deep Learning

Joakim Andersson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Mechatronics research group
Chalmers University of Technology
Gothenburg, Sweden 2018

Predicting Vehicle Motion and Driver Intent using Deep Learning
JOAKIM ANDERSSON

Supervisor: Martin Sanfridson, AB Volvo
Examiner: Nikolce Murgovski, Chalmers University of Technology

Master's Thesis 2018:EX045
Department of Electrical Engineering
Division of Systems and Control
Mechatronics research group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2018

Predicting Vehicle Motion and Driver Intent using Deep Learning
JOAKIM ANDERSSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis investigates the application of Deep Learning and Mixture Models on the prediction of human drivers in traffic. The chosen approach is a Mixture Density Network where the Neural Network is composed of Long-Short Term Memory units and the Mixture Model consists of univariate Gaussian distributions. The model outputs accelerations for the next four seconds in longitudinal and lateral directions as well as indicating whether the driver intends to change lane during this time or not. The results of the motion prediction is promising with a Mean Absolute Error of around 1 meter after a 4 second prediction. The intention prediction results in around 75% accuracy in identifying lane change trajectories 1.7 seconds prior to the lane change and around 95% accuracy 1 second prior to the lane-change. The conclusion of the study is that in order to accurately predict humans, the model has to make use of as many variables as possible. However it will not be possible to fully predict humans by just observing the environment since there are many unobservable variables affecting the decision-making. One way to take this into account is to make use of signals humans use to display their intentions, such as indicator lights.

Keywords: Deep Learning, LSTM, Mixture Density Network, Prediction.

Acknowledgements

I want to thank AB Volvo for the opportunity to work on this problem. I would also like to give a huge thank you to my examinator Nikolce Murgovski and my supervisor Martin Sanfrison for all their time and valuable feedback I have received during the thesis.

Joakim Andersson, Gothenburg, May 2018

Contents

List of Figures

List of Tables

1

Introduction

Humans today spend a large portion of their time travelling; the society as a whole has grown dependent on transportation, with the average American spending over 300 hours in their car each year [1]. As the time spent in cars increases so does the accidents since today's transportation system is far from perfect when it comes to safety. The World Health Organization (WHO) produced a report on the total number of fatal accidents during 2015 where they stated that around 1.25 million people lost their lives in road related accidents throughout the world [2].

During the past few years there has been a massive surge towards autonomous vehicles within the automotive industry. The hype stems from recent years success in image recognition and perception, allowing computers to, in some scenarios, achieve super-human performance [3]. However, perception of the environment is not all that is required to solve the problem of self-driving cars as the car needs to operate safely in this environment as well.

Achieving full autonomy would not only lead to a lower environmental footprint as it would allow traffic to flow more freely, but also, and foremost, it would increase the safety of driving. Some experts in the industry believe that there would be a drop of up to 80% in the rate of accidents following the introduction of autonomous vehicles [4, 5].

Self-driving vehicles is something that will engage engineers for many years to come. Today, however, many vehicles are already equipped with so called Advanced Driver-Assistance Systems (ADAS) which do not allow full autonomy but they, as the name suggests, assists human drivers in driving the car.

One of the biggest challenges with both the current ADAS systems and the full autonomy is the human factor. Self-driving cars and ADAS systems operate in close proximity to humans, meaning that they need to be able to predict how other human drivers will behave in order to ensure safe operation. This thesis will research the prediction of humans in other cars.

1.1 Purpose

Humans are difficult to model and predict since humans are irrational. Furthermore, no human is another one alike, meaning there are almost endless possible outcomes in every scenario.

When driving it is almost always necessary to know what the surrounding cars intend to do in order to ensure safe and energy optimized control of the vehicle. Take the example illustrated in figure 1.1, if the rear car in the left lane intends to take the exit on its right, it has to predict how the other cars will behave in order safely maneuver itself to the exit, if it's even possible from a safety perspective.

Because of the irrationality of humans and the recent success seen in the field of deep learning this thesis will take a data driven approach to prediction. The aim of the thesis is to see to what extent a human driver's intent and the vehicle motion in general can be predicted using deep learning. In order to give at least some measure of how certain the network is, the chosen method is a composition of deep learning and Gaussian mixture models.

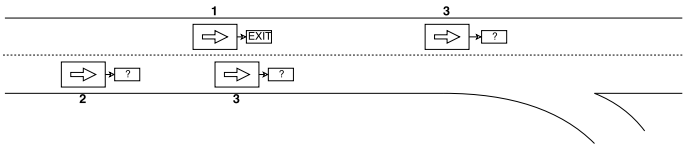


Figure 1.1: A possible scenario on a highway with a lane with two possible choices.

1.2 Research Questions

The research questions considered in this thesis are

- How well can deep learning predict vehicle motion.
- How well can deep learning predict the intent of the driver.
- How robust is this prediction to noise or incomplete perception of the world.
- Is the model uncertainty useful, that is, does the standard deviation of the mixture model increase for a bad prediction when compared to a good prediction.

1.3 Scope and Limitations

The prediction of vehicles will take place in a highway setting. The highway has multiple lanes, a merge and an exit ramp. The thesis will make use of a publicly available dataset called NGSIM [6]. The data are thus limited to the captured scenarios within that dataset. The data consist of normal driving, there are no captured collisions, no data for appearances of foreign objects such as road debris, etc. Since it is a highway, there are no pedestrians, hitchhikers or other humans without vehicles present so the predictions are done solely on humans operating vehicles.

In order to see the bottlenecks of the data driven approach it is assumed during training that the data can be collected flawlessly, all involved metrics can be collected.

2

Theory

2.1 Time-Series

A time series T is defined as an ordered sequence of n real-valued variables

$$T = \{t_1, t_2, \ldots, t_n\}, t_i \in \mathbb{R}.$$

By observing an underlying process over time one is usually collecting ordered, uniformly spaced in time, real valued observations thus identified as a time series. A time series can by definition be *univariate* or *multivariate*; in the latter case there are multiple dimensions observed simultaneously [7].

Given a time series T, a *subsequence* S of T is defined as a series of length $m \leq n$ consisting of contiguous observations from T

$$S = \{t_k, t_{k+1}, \ldots, t_{k+m-1}\}$$

where $1 \leq k \leq n - m + 1$.

2.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computing systems inspired by the biological neural networks. One of the most basic ANN structures is the *feed-forward network* (FFN). The building block of a modern FFN is a neuron, see Figure 2.1, which takes several real valued inputs x_i, multiplies by some weights w_i adds a bias term b and produces an output as follows

$$output = g\left(\sum w_i x_i + b\right) \tag{2.1}$$

where g is an activation function. [8]

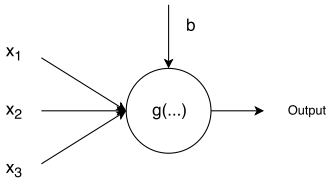


Figure 2.1: A neuron, which takes inputs x_i and bias b and produces an output according to equation (2.1).

The neuron units can be stacked in one or multiple layers; a *multilayer perceptron* MLP can be seen in the Figure 2.2, where a FFN with 2 hidden layers is on display.

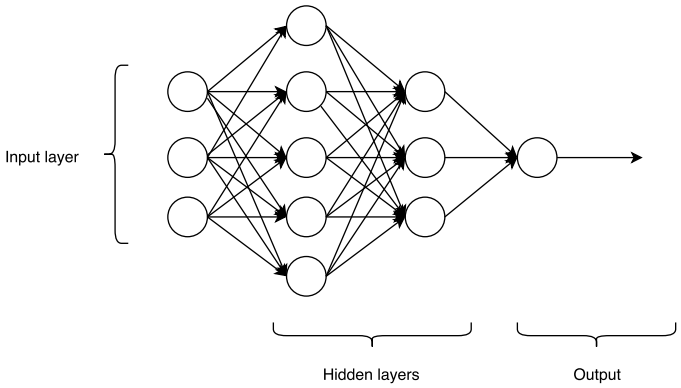


Figure 2.2: A schematic view of a multilayer perceptron with 3 inputs, 2 hidden layers and 1 output.

2.2.1 Supervised Learning

In machine learning there are different types of learning paradigms. In this section, the paradigm of supervised learning (SL) will be introduced.

SL is based on labeled data. For each set of inputs there is a desired, known, output. The training procedure for this type of supervised learning is known as an error-correction learning algorithm. In the case of MLPs and ANNs in general this is implemented as a two step procedure, the forward and the backward passes.

1. **Forward pass:** The network is fed inputs in the training sample and carries out the feed forward propagation, neuron for neuron until an output is produced. This output is then used to find the error signal as defined by some arbitrary loss function.

2. **Backward pass:** From the error signal each individual parameter's contribution to this error is calculated by differentiating the loss function with regard to each parameter. The error in the output of the network is thus back-

propagated throughout the network so that the parameters can be adjusted accordingly.

2.2.2 Recurrent Neural Networks

A recurrent neural network (RNN) is a type of ANN that differs from the basic FFN in how the output of a neuron is handled. In RNNs the output is passed forward in the network simultaneously as it is passed as input to the same neuron again in the next iteration, see Figure 2.3. The introduction of so called recurrent edges that connect adjacent time-steps by introducing cycles in a neuron gives the network a sense of time [9]. This is useful when there is a strong correlation between samples, such as in sequential time-series of sensor data.

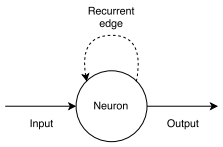


Figure 2.3: A schematic overview of a simple recurrent network unit. The output is fed as input in the next time-step again.

A more formal view of the simple recurrent network (SRN) unit is that at time t, Nodes with recurrent edges receive input in two forms, sample input $\boldsymbol{x}^t$ and hidden values $\boldsymbol{h}^{t-1}$ from the network's previous state. The output $\hat{\boldsymbol{y}}^t$ at time t can be calculated by the hidden state $\boldsymbol{h}^t$ at time t. Specifically the two calculations can be described by the equations

$$\boldsymbol{h}^t = g(\boldsymbol{W}_{hx}\boldsymbol{x}^t + \boldsymbol{W}_{hh}\boldsymbol{h}^{t-1} + \boldsymbol{b}_h) \tag{2.2}$$

$$\boldsymbol{y}^t = g'(\boldsymbol{W}_{yh}\boldsymbol{h}^t + \boldsymbol{b}_y) \tag{2.3}$$

where $g(x)$ and $g'(x)$ are activation functions, $\boldsymbol{W}_{hx}$, $\boldsymbol{W}_{hh}$ and $\boldsymbol{W}_{yh}$ are weights associated with the input, hidden state and output respectively. The two parameters $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$ are called biases.

2.2.2.1 Back Propagation Through Time

The recurrent network can be "unfolded" to a structure looking like a deep feed-forward network with shared weights between layers, see Figure 2.4. It is now possible to apply the back-propagation algorithm used in regular FFNs in order to optimize the network parameters [10].

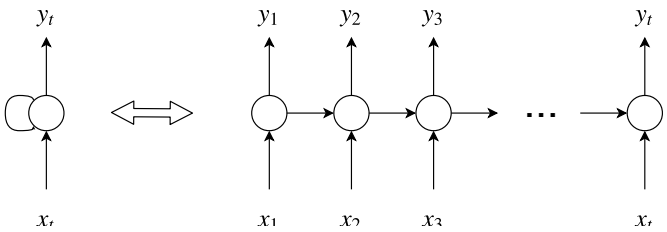


Figure 2.4: A recurrent unit can be unfolded in time.

2.2.2.2 Long Short-Term Memory networks

A problem that arises during training of basic RNNs is the exploding or vanishing gradient [11][12]. The Long Short-Term Memory (LSTM) architecture was designed as to avoid these problems [13]. This structure has since been further enhanced to what is called the vanilla LSTM architecture visualized in Figure 2.5 [14]. In the figure it is clear that the LSTM is far more complex than the basic SRN unit. The building blocks of the LSTM are three gates (input, forget, output), block input, a single cell (the Constant Error Carousel) and an output activation function. Mathematically the LSTM layer with N units and M inputs can be described as follows:

- Input weights: $\boldsymbol{W_z}, \boldsymbol{W_i}, \boldsymbol{W_f}, \boldsymbol{W_o} \in \mathbb{R}^{N \times M}$
- Recurrent weights: $\boldsymbol{R_z}, \boldsymbol{R_i}, \boldsymbol{R_f}, \boldsymbol{R_o} \in \mathbb{R}^{N \times N}$
- Bias weights: $\boldsymbol{b_z}, \boldsymbol{b_i}, \boldsymbol{b_f}, \boldsymbol{b_o} \in \mathbb{R}^{N}$

with the update rules

$$
\begin{aligned}
\boldsymbol{z}^t &= g(\boldsymbol{W}_z \boldsymbol{x}^t + \boldsymbol{R}_z \boldsymbol{y}^{t-1} + \boldsymbol{b}_z) \\
\boldsymbol{i}^t &= \sigma(\boldsymbol{W}_i \boldsymbol{x}^t + \boldsymbol{R}_i \boldsymbol{y}^{t-1} + \boldsymbol{b}_i) \\
\boldsymbol{f}^t &= \sigma(\boldsymbol{W}_f \boldsymbol{x}^t + \boldsymbol{R}_f \boldsymbol{y}^{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{c}^t &= \boldsymbol{z}^t \odot \boldsymbol{i}^t + \boldsymbol{c}^{t-1} \odot \boldsymbol{f}^t \\
\boldsymbol{o}^t &= \sigma(\boldsymbol{W}_o \boldsymbol{x}^t + \boldsymbol{R}_o \boldsymbol{y}^{t-1} + \boldsymbol{b}_o) \\
\boldsymbol{y}^t &= h(\boldsymbol{c}^t) \odot \boldsymbol{o}^t
\end{aligned}
\tag{2.4}
$$

where $\odot$ defines pointwise multiplication, σ, g and h are point-wise non-linear activation functions. The sigmoid function, σ, is used as gate activation, and the hyperbolic tangent is usually used as the block input and output activation functions, h and g.

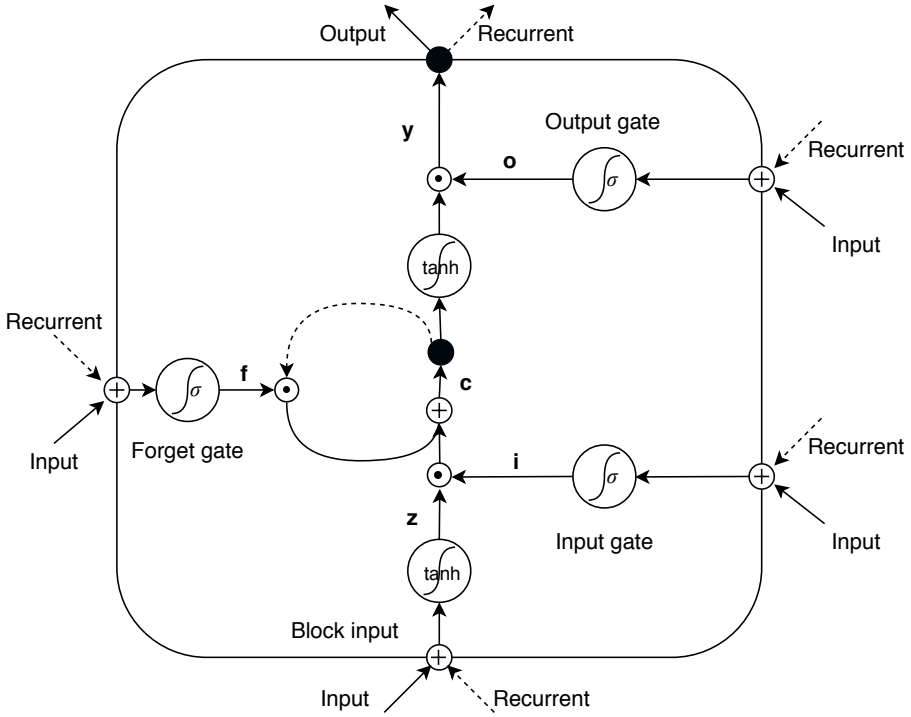


Figure 2.5: The inner workings of an LSTM block. The dashed lines represent time delayed connections, recurrent edges. σ and tanh are the sigmoid and hyperbolic tangent activation functions respectively.

2.2.3 Regularization

One of the big challenges in machine learning is how to make a trained model perform better on unseen samples. Many strategies used in machine learning have therefore been developed with the intention of making the model perform better on the test set, sometimes at the expense of increased training error. These strategies are collectively known as regularization [15].

2.2.3.1 Bootstrap aggregating

Bootstrap aggregating, known as bagging, is a form of regularization where multiple models are generated from the data. The different versions are generated by making bootstrap replicates of the dataset and train each model on one of the new datasets. The models are then aggregated to allow the average of the models to become the final predicted result [16]. This is the principle behind some of the more successful machine learning algorithms like random forests [17].

2.2.3.2 Dropout

Dropout is a form of regularization that is very common to use in combination with neural networks. Neural networks are usually expensive to train, which is why regular bagging is not practical. Dropout can be thought of as an inexpensive way of combining bagging with neural networks. During the training phase of the network, with the introduction of dropout, every unit has a probability p of being "dropped" from this training round, meaning that the unit and connecting edges is temporarily removed from the network, creating a new model [18]. Refer to Figure 2.6 for an example of three submodels created by dropout.

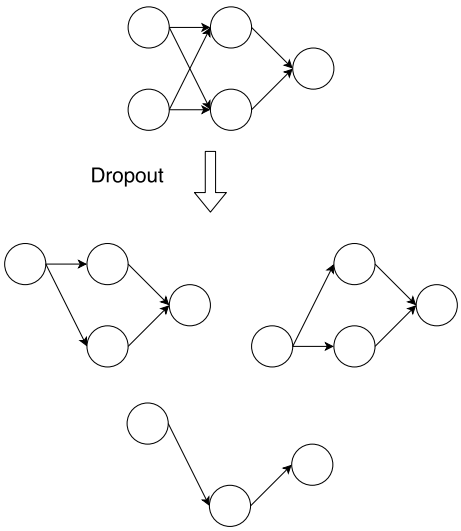


Figure 2.6: Example of three submodels created by dropout from a fully connected network.

2.2.3.3 Multi-Task Learning

Multi-Task Learning (MTL) is a sub-field in machine learning that aims to do more than one thing for the same input. By sharing representation between tasks one enables the model to generalize better, with the limitation that the tasks usually have to be related to some extent [19]. One example of this could be allowing one neural network to predict both longitudinal and lateral acceleration from the same input, where the first few layers in the network are shared between the two tasks, and then separated into two different layers producing the final outputs.

2.3 Mixture Gaussian Distribution

One problem with a regular neural network is that the output does not contain any measurement of uncertainty. Statistical distributions, however, are designed to show the probability of the likeliness of one measurement. The Gaussian mixture distribution is a mixture of Gaussian distributions defined as follows:

$$p(y|\mu_1, \ldots, \mu_k, \sigma_1, \ldots, \sigma_k, \pi_1, \ldots, \pi_k) = \sum_{i=1}^{k} \pi_i \mathcal{N}(\mu_i, \sigma_j^2) \tag{2.5}$$

where μ_i are the means, σ_i the variances, π_i the mixing proportions (which must be positive and sum to one) and $\mathcal{N}$ a normalized Gaussian distribution [20].

2.3.1 Mixture Density Networks

Mixture density networks (MDNs) integrate the idea of a mixture distribution into neural networks, creating a network that is less error-prone in cases where the conditional average is too limited to fully map the input-output space [21].

The architecture, displayed in Figure 2.7, works as follows: An input vector is fed into a neural network. The neural network, outputs parameters of a mixture distribution which in turn are used to construct a mixture model. The mixture model gives the conditional probability density for the possible outcomes given the input vector.

Assuming the conditional probability density can be represented by a mixture model, the density can be written as

$$p(\boldsymbol{t}|\boldsymbol{x}) = \sum_{j=1}^{k} \alpha_j(\boldsymbol{x}) \phi_j(\boldsymbol{t}|\boldsymbol{x}) \tag{2.6}$$

where $\alpha_j(\boldsymbol{x})$ are called mixing coefficients, and can be regarded as prior probabilities (conditioned on $\boldsymbol{x}$) of the target vector $\boldsymbol{t}$ generated from the j^{th} component of the mixture. The $\phi_j(\boldsymbol{t}|\boldsymbol{x})$ represent the conditional density of the target vector of the j^{th} kernel, where the kernel may be any probability distribution [22].

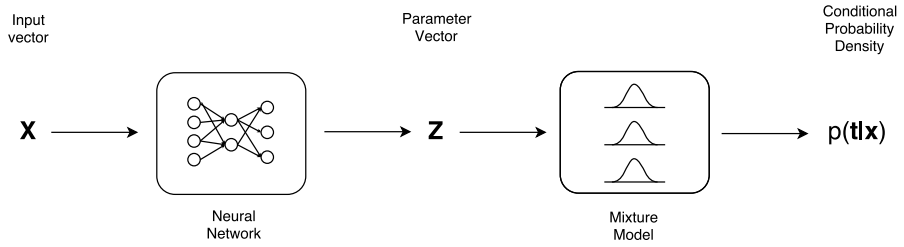


Figure 2.7: The architecture of a mixture density network. The input vector is fed into a neural network, which outputs parameters of a mixture distribution. The parameters are then used to construct a probability distribution over the possible outcomes.

2.3.1.1 Loss function

A loss function can be constructed by maximizing the likelihood that the mixture model gave rise to the target values, the likelihood can be written as

$$\mathcal{L} = \prod_{q=1}^{n} p(\boldsymbol{t}^q, \boldsymbol{x}^q) \tag{2.7}$$

$$= \prod_{q=1}^{n} p(\boldsymbol{t}^q|\boldsymbol{x}^q)p(\boldsymbol{x}^q) \tag{2.8}$$

$$\tag{2.9}$$

where q is a training pattern. It is convenient to instead minimize the negative logarithm of $\mathcal{L}$. Defining the error function

$$\boldsymbol{E} = -\ln(\mathcal{L})$$

one may write

$$\boldsymbol{E} = \sum_{q} \boldsymbol{E}^q \tag{2.10}$$

$$\boldsymbol{E}^q \propto_\theta -\ln\left[\sum_{j=1}^{k} \alpha_j(\boldsymbol{x}^q)\phi_j(\boldsymbol{t}^q|\boldsymbol{x}^q)\right] \tag{2.11}$$

where the last step is discarding constant terms and keeping the terms that are dependent on the model parameters θ. This loss function can now be used to optimize the parameters in the neural network model.

2.4 Data Analysis

2.4.1 Kalman Filter

Kalman filters are based on discretized linear dynamical systems. A Kalman filter assumes that the state at time t evolved from the state at time t-1 according to the equation

$$\boldsymbol{x_t} = \boldsymbol{F}_t\boldsymbol{x}_{t-1} + \boldsymbol{B}_t\boldsymbol{u}_t + \boldsymbol{w}_t \tag{2.12}$$

where $\boldsymbol{x}_t$ is the state vector, $\boldsymbol{u}_t$ is the control signal, $\boldsymbol{F}_t$ is the state transition matrix, $\boldsymbol{B}_t$ is the control input matrix and $\boldsymbol{w}_t$ is the process noise vector, assumed to be Gaussian with mean zero and covariance matrix $\boldsymbol{Q}_t$. The system may also allow measurements to be performed according to

$$\boldsymbol{z}_t = \boldsymbol{H}_t\boldsymbol{x}_t + \boldsymbol{v}_t \tag{2.13}$$

where $\boldsymbol{H}_t$ is the transformation matrix, $\boldsymbol{z}_t$ is the vector of measurements and $\boldsymbol{v}_t$ is the vector containing measurement noise, assumed to be Gaussian with mean zero with covariance matrix $\boldsymbol{R}_t$. [23] [24]

The Kalman filter is often conceptualized as having two distinct phases, the predict and the update phases. In the predict phase it uses the state estimate from the previous iteration to estimate the current state, the *a priori* state.

$$\begin{aligned}\hat{\boldsymbol{x}}_{t|t-1} &= \boldsymbol{F}_t\hat{\boldsymbol{x}}_{t-1|t-1} + \boldsymbol{B}_t\boldsymbol{u}_t \\ \boldsymbol{P}_{t|t-1} &= \boldsymbol{F}_t\boldsymbol{P}_{t-1|t-1}\boldsymbol{F}_t^T + \boldsymbol{Q}_t\end{aligned} \tag{2.14}$$

After the predict phase the *a priori*, the algorithm updates the *a priori* state estimate with the measurement provided for the current time step, this new state is called the *a posteriori* state. The update phase performs the following updates.

$$\begin{aligned}\hat{\boldsymbol{y}}_t &= \boldsymbol{z}_t - \boldsymbol{H}_t\hat{\boldsymbol{x}}_{t|t-1} \\ \boldsymbol{S}_t &= \boldsymbol{R}_t + \boldsymbol{H}_t\boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^T \\ \boldsymbol{K}_t &= \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^T\boldsymbol{S}_t^{-1} \\ \hat{\boldsymbol{x}}_{t|t} &= \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\hat{\boldsymbol{y}}_t \\ \boldsymbol{P}_{t|t} &= (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)\boldsymbol{P}_{t|t-1}(\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)^T + \boldsymbol{K}_t\boldsymbol{R}_t\boldsymbol{K}_t^T \\ \hat{\boldsymbol{y}}_{t|t} &= \boldsymbol{z}_t - \boldsymbol{H}_t\hat{\boldsymbol{x}}_{t|t}\end{aligned} \tag{2.15}$$

2.4.2 Metrics

2.4.2.1 Mean Absolute Error

The Mean Absolute Error (MAE) is a measure of difference between two continuous variables often used to evaluate forecasts of time-series. It is a natural measurement as it is straightforward and its interpretation is clear, as opposed to the root mean squared error. MAE is defined mathematically as:

$$\text{MAE} = \frac{\sum_i^n |y_i - \hat{y}_i|}{n} \tag{2.16}$$

where y_i is the true value, $\hat{y}_i$ is the estimated value and n the total number of variables.

2.4.2.2 Kullback-Leibler Divergence

Kullback-Leibler divergence (KL-Divergence) or relative entropy is a non-symmetric measurement metric of how much a distribution diverges from another expected distribution [25]. Specifically the measurement KL(q||p) measures how much information would be lost by approximating q with p [26].

$$KL(q||p) = -\int q(\boldsymbol{Z}) \ln\left(\frac{p(\boldsymbol{Z})}{q(\boldsymbol{Z})}\right) d\boldsymbol{Z} \tag{2.17}$$

$$= -\int q(\boldsymbol{Z}) \ln(p(\boldsymbol{Z}))d\boldsymbol{Z} + \int q(\boldsymbol{Z})q(\boldsymbol{Z})d\boldsymbol{Z}. \tag{2.18}$$

This is thus useful for evaluating a model consisting of several distributions since a low divergence indicates that it most likely would suffice with lower number of distributions.

When the KL-divergence is between two Gaussian distributions the above formula simplifies to

$$KL(q||p) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \tag{2.19}$$

Using this formula, it is possible to evaluate how similar the different distributions in a mixture model are. If the value is low, close to zero, one may assume that the distribution is unnecessary, as the variable could be simulated from the other distribution alone and by the principle of Occam's razor, the lower number of distributions may be preferred [27].

2.4.3 Precision, Recall and F-Score

Precision is defined as the ratio of relevant or true positive samples to the total number of predicted positives, mathematically formulated as

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.20}$$

Thus a precision value of 1 indicated that the algorithm didn't produce any false positives, meaning that the results from the prediction is very trustworthy.

Recall is a measurement describing the fraction of relevant samples that were retrieved, mathematically

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.21}$$

A recall value of 1 means that the algorithm was able to correctly identify all relevant samples in the data set.

The F1-score of a classification is the harmonic mean of the precision and recall, specifically it can be described as

$$\text{F1-Score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.22}$$

An F1-score of 1 means that the classifier has perfect precision and recall for the measured quantity.

3

Prediction Approach

3.1 Data

The data used in this thesis comes from the New Generation Simulation (NGSIM) dataset. NGSIM is publicly available and consists of recorded data from several highways in the US. Specifically the recording of US-101 in California will be used. The data is captured by several cameras located above the highway recording information of the vehicles on the highway. Each objects coordinate is recorded at each sample. The sampling rate of the data is 10Hz and the whole recording spans 45 minutes.

3.1.1 Filtering

To extract the accelerations in longitudinal, x, and lateral, y direction at each time step, the positions will be differentiated in relation to time according to

$$\dot{x} = \frac{d}{dt}x \tag{3.1}$$

$$\dot{y} = \frac{d}{dt}y \tag{3.2}$$

$$\ddot{x} = \frac{d}{dt}\dot{x} \tag{3.3}$$

$$\ddot{y} = \frac{d}{dt}\dot{y} \tag{3.4}$$

$$\tag{3.5}$$

The data was captured by sensors sitting high above the ground overlooking the highway, so the positional data have some degree of noise. In order to avoid unrealistic acceleration values the positions must be filtered. The underlying process is a dynamic linear system, which means that a Kalman filter can be applied.

To construct the filter the longitudinal and lateral data will be handled separately to simplify the constructed filters to one dimension.

In section 2.4.1 the different components of a Kalman filter were defined, specifically $\boldsymbol{F}, \boldsymbol{H}, \boldsymbol{B}, \boldsymbol{R}, \boldsymbol{Q}$, however, there is no control signal in our case, since the acceleration is unknown, instead, one may assume that the next state is given by

$$\boldsymbol{x_t} = \boldsymbol{F}\boldsymbol{x}_{t-1} + \boldsymbol{G}a_t \tag{3.6}$$

$$\boldsymbol{x_t} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{3.7}$$

where a_t is an unknown input (normally distributed with mean 0 and standard deviation σ_a) and $\boldsymbol{G}$ applies this effect to the state vector with

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{3.8}$$

$$\boldsymbol{G} = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \tag{3.9}$$

$$\tag{3.10}$$

thus, the update function, Equation (2.12) may be written as

$$\boldsymbol{x}_t = \boldsymbol{F}\boldsymbol{x}_{t-1} + \boldsymbol{w}_t \tag{3.11}$$

$$\boldsymbol{w}_t \sim \mathcal{N}(0, \boldsymbol{Q}) \tag{3.12}$$

$$\boldsymbol{Q} = \boldsymbol{G}\boldsymbol{G}^T\sigma_a^2 = \begin{bmatrix} \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 \\ \frac{1}{2}\Delta t^3 & \Delta t^2 \end{bmatrix} \sigma_a^2. \tag{3.13}$$

The measurements are given by Equation (2.13), with

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

and

$$\boldsymbol{R} = \sigma_z^2$$

where σ_z is 2ft and 4ft for lateral and longitudinal positions respectively, as specified in the dataset. The algorithm is given adequate values for σ_a, $\hat{\boldsymbol{x}}_{0|0}$ and $\boldsymbol{P}_{0|0}$ after which the prediction and update formulas in Equations (2.14) (2.12) can be applied iteratively.

3.2 Network Architecture

From the previous section it is clear that the data is a time series. The correlation between the samples is well suited for a RNN consisting of LSTM neurons. LSTM was preferred over other RNN units such as the gated recurrent unit based on empirical studies on the task at hand. The specific type of LSTM used was the vanilla LSTM described in Figure 2.5.

The number of layers of LSTMs was three with 512 units in each layer, chosen to balance performance with computational effort. The performance gained from adding further layers did not outweigh the increased training time.

3.2.1 Input Features

In order to allow the network to capture the most relevant correlations and features the network will be fed as much information as possible. However, since the estimation is intended to operate in real-time, on-board a vehicle, the features have to be observable by the on-board sensors. The input features are as follows:

- Target vehicle
  - Longitudinal acceleration
  - Lateral acceleration
  - Longitudinal velocity
  - Lateral velocity
  - Vehicle type
  - Has right lane
  - Has left lane
- Surrounding vehicles
  - Delta longitudinal position
  - Delta lateral position
  - Delta longitudinal velocity

– Delta lateral velocity

The situation is displayed in Figure 3.1. If there is no car in one of the specified places the corresponding inputs will be filled with artificial data.

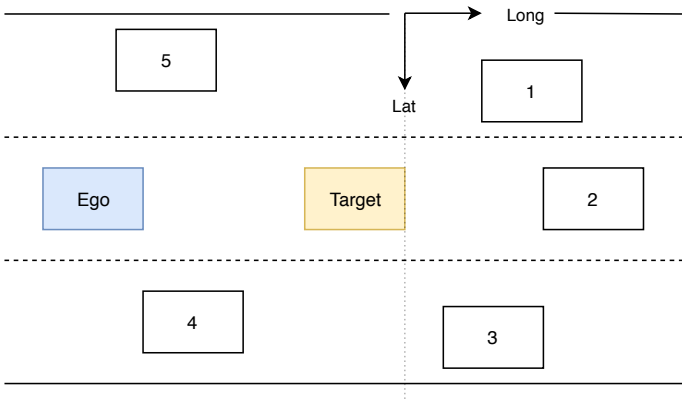


Figure 3.1: The considered vehicles when the ego vehicle, the vehicle where the sensors are located, wants to predict the vehicle in-front.

3.2.2 Motion prediction

The motion of vehicles can be described by Newtonian physics. If the network outputs acceleration along one dimension, $\ddot{x}$, the motion along that axis can be calculated according to the equations

$$\dot{x}_{t+1} = \dot{x}_t + \ddot{x}_t \Delta t \tag{3.14}$$

$$x_{t+1} = x_t + \dot{x}_t \Delta t + \frac{\ddot{x}_t \Delta t^2}{2} \tag{3.15}$$

Driving is not black and white, there are many valid decisions at any point in time, and different humans may choose differently given the same scenario. With this in mind it is not justifiable outputting a scalar value for the acceleration in the next time steps. However, there should be a value that is more likely to occur than others, given a specific driver. This leads us to believe it would be best to output some sort of Gaussian distribution given a situation. However, there may be differences between different humans, some may drive more aggressively than others, which should be captured in the output. The selected approach is a mixture model, composed by Gaussian distributions. The whole method of motion prediction is thus an MDN as described in Figure 2.7.

This thesis will evaluate a mixture of univariate Gaussians. The neural network should output the parameters to these distributions, so in the following section the MDN layer will be described in more detail.

3.2.2.1 Univariate Gaussian

The equations describing the univariate Gaussian mixtures are shown in equation (2.5). The weights π_i will have a softmax function as they should sum to one, the means μ_i will have no activation function as they could theoretically take on any value and the standard deviation σ_i will have an exponential activation function to make sure they are positive. Thus, there are three parameters for each mixture that the network has to find.

3.2.3 Intention prediction

To capture the intention of the driver there is a need for another layer in parallel to the motion prediction layer after the LSTM layers. This layer will indicate if the driver intends to switch lane (and remain there) within the next four seconds. This task is closely related to the lateral motion prediction, but the complexity of the output is much lower. The output from this layer should thus be a simple indicator if it intends to change lane and to the left or right. This is easily represented by a three state vector with probabilities for each choice. The layer is thus a simple, fully connected layer with three neurons with the softmax activation function. The motivation behind this is according to the paradigm of multi-task learning, it may be possible for the network to generalize better in the case of lane changes if there is some abstract representation for lane change embedded in the LSTM layers.

3.2.4 Overview of the models

For the one-dimensional prediction, the single-task network, the overview of the network architecture is shown in Figure 3.2. For the multi-task model, motion and intent prediction on the same network, the overview can be found in Figure 3.3.

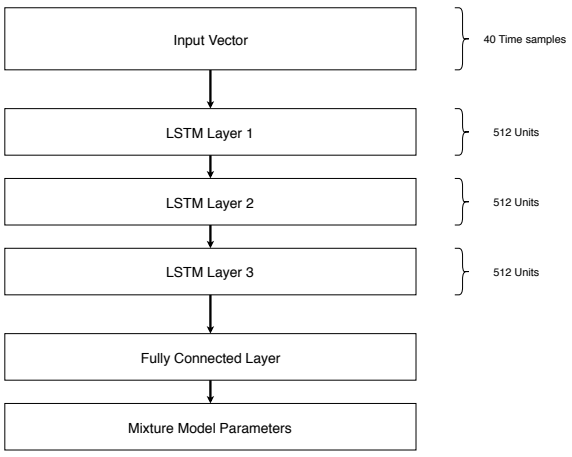


Figure 3.2: Architecture for the single-task network.

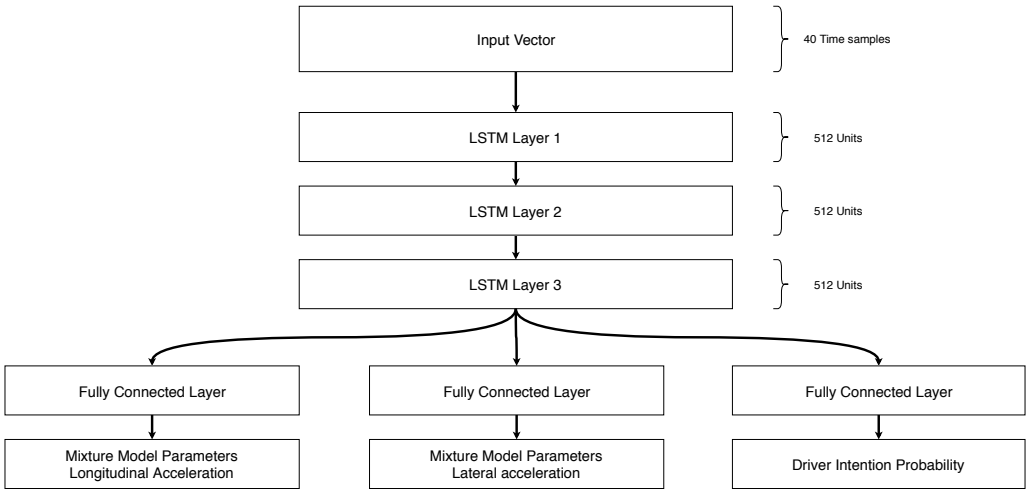


Figure 3.3: Architecture for the multi-task network.

3.3 Training and Evaluation

In order to evaluate the models, the data will be split into three different sets, training, validation and test set. Training and validation set will be used in the training process with the training set providing information for the error signal. The validation set is then used after each training epoch (one epoch is one pass over all training data) to evaluate whether the network got better in general or if it is starting to learn the characteristics of the specific training samples, so called over-fitting. The last set, the test set, is used to evaluate the model performance.

The evaluation metric used is the MAE where the trajectories are generated from the test set.

For a relative performance measure of the model results will be compared to some baseline results. The baseline results will be generated using two simple methods, prediction using constant velocity and constant acceleration for all the trajectories in the test set. For these cases the predicted trajectory will follow the trajectory generated from either keeping the last observed velocity or acceleration constant throughout the prediction horizon.

3.3.1 Robustness

It is not possible to have a full history of every vehicle at all times, e.g. a car approaching from behind or a car merging onto the highway. Even without full history one would like to be able to predict that vehicle's movement in some way. For this reason it is of interest to look at how the network handles the situation

where the full history is only partially observable. The chosen approach for this is to approximate the history by allowing the unobserved samples to be equal to the first observed measurement. The motivation behind this is that it is unlikely that the velocity increased too much during the unobserved time, and also it allows for an easy comparison with the constant velocity/acceleration prediction baselines.

Furthermore, in reality perfect recollection of the surroundings is not always possible, sometimes some vehicles will be unobservable due to it being hidden from the sensors. The training of the network assumed perfect observability, but it is interesting to look at how this transfers into situations without perfect data. For this reason the models will be evaluated with some data being replaced by the artificial values indicating no vehicle present. More specifically the model will be evaluated for the following scenarios, refer to Figure 3.1:

I Vehicle 1 missing

II Vehicle 2 missing

III Vehicle 3 missing

IV Vehicle 4 missing

V Vehicle 5 missing

The evaluation of these cases will also give some information on how valuable the obfuscated information is to the network.

4

Results

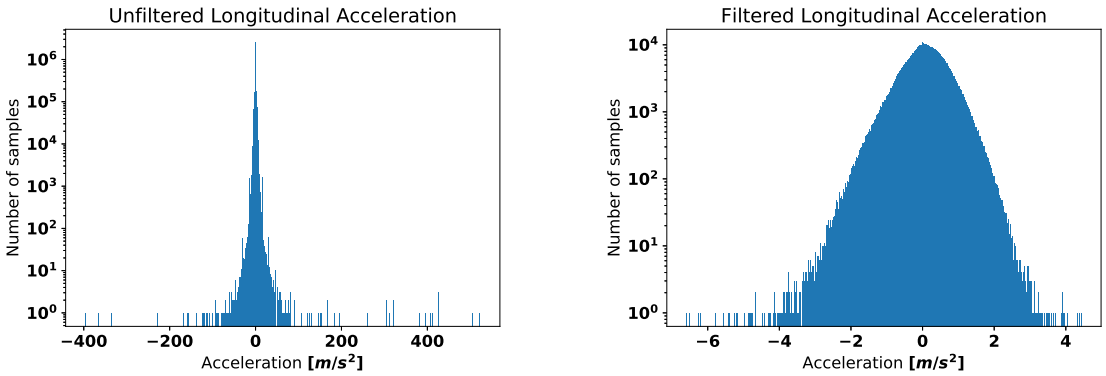
4.1 Data Analysis

4.1.1 Acceleration

During the data exploration in the beginning of the thesis it was noted that the position data was subject to a relatively large amount of noise. It was not apparent from visual inspection of the trajectories, but extracting the acceleration between contiguous time samples made it clear. The raw acceleration data obtained by differentiation of the postitional data in the data set can be seen in the left plot in Figure 4.1. As described in section 3.1.1 a Kalman filter was applied to the raw data with the parameters of the Kalman filter chosen as per the specifications of the recording cameras. The result is shown in the right plot in Figure 4.1. In the histogram it is clear that the acceleration data is more reasonable after filtering. The accelerations are in the interval $[-6.5, 4.3] m/s^2$. However, the majority of the acceleration lies centered around 0 and it follows a normal distribution.

The most important aspect of the filtration is to make sure that it doesn't completely change the trajectory of the vehicle, so the parameter controlling the standard deviation of the applied acceleration had to be tuned manually. The tuning was controlled by a measurement indicating the deviation of the filtered trajectory from the raw data. This measurement needed to be minimized with the added constraint of keeping the resulting acceleration values reasonable.

The result of the filtering successfully matches what it intended to do. The data is much more reasonable without losing the characteristics of the raw data.



(a) Unfiltered acceleration

(b) Kalman-filtered acceleration

Figure 4.1: Acceleration values extracted from the unfiltered data, the left plot, and the Kalman-filtered data. The y-axis has a log-scale to visualize the outliers in both cases.

4.1.2 Correlation

The correlation between the first target acceleration and the past acceleration values is plotted in Figure 4.2. The correlation between past values for the longitudinal acceleration is much more persistent than for the lateral acceleration.

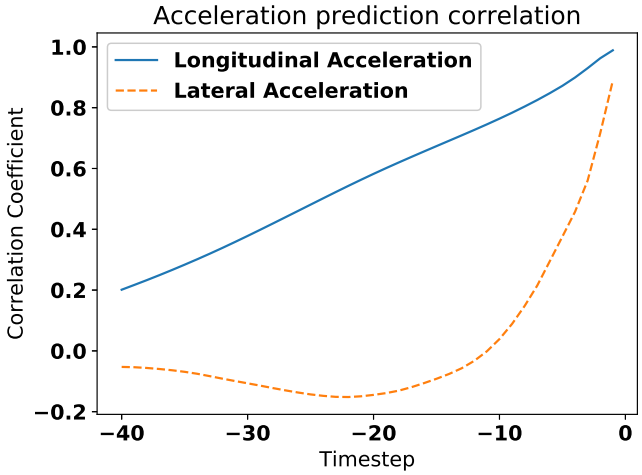


Figure 4.2: Plot showing the correlation between the first target acceleration and the previous acceleration values.

4.1.3 Effect of dropout

To increase model efficiency and prevent over-fitting, dropout was added at each LSTM layer. To evaluate what the dropout coefficient should be a micro benchmark was done for a subset of the data. Three values of dropout were tested, none, 20% and 50%. The result is shown in Figure 4.3. It is clear that 20% performs best and no dropout performing the worst.

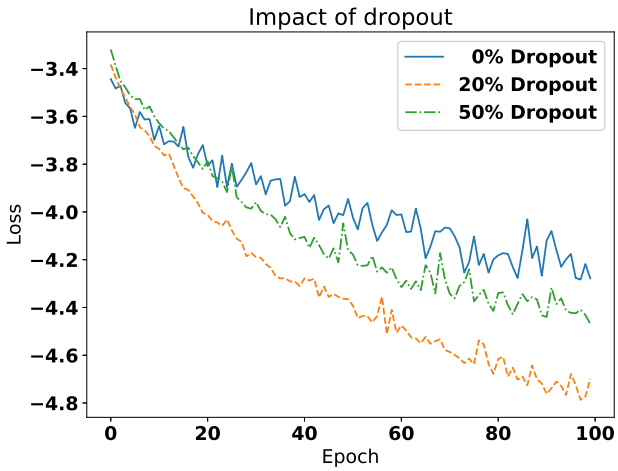


Figure 4.3: Impact of the choice of dropout during the training phase. In the figure three different values of dropout were chosen and run on subset of the training data. The loss function for the mixture density network is displayed on the y-axis. It is clear that 20% dropout has a competitive advantage over the other two values.

4.1.4 Number of distributions in the mixture

Using the KL-divergence for Gaussian distributions, Equation (2.19), the number of distributions in the mixture was evaluated. The result can be seen in the table below.

Table 4.1: The average lowest KL-divergence was calculated on the test set for three different number of distributions in the mixture model.

Distributions	Average lowest KL-divergence
2	2.61
3	0.2055
4	0.0626

Almost no information is lost if the Gaussian with the least impact were to be removed in the case of a model of four Gaussians and very little in the case of three

Gaussians compared to the model with two Gaussians.

Based on the results in the table, the risk of overfitting and the increased computational effort with multiple distributions, it was decided that the optimal number of distributions was two.

4.2 Longitudinal prediction

To compare the network's performance to a baseline the two methods described previously are evaluated against the test set along with the network. The result of the evaluation are shown in Table 4.2 below

Table 4.2: Baseline results compared with the single-task model for longitudinal prediction.

Method	Mean absolute error in m at time horizon [s]			
	1	2	3	4
Constant velocity	0.281 ± 0.232	1.00 ± 0.823	2.13 ± 1.74	3.62 ± 2.95
Constant acceleration	0.164 ± 0.137	0.610 ± 0.508	1.35 ± 1.13	2.41 ± 2.00
Single-task model	0.0434 ± 0.0616	0.223 ± 0.257	0.549 ± 0.575	1.05 ± 1.05

as the table indicates, the constant acceleration baseline performs better than the constant velocity approach. This indicates that the majority of the data seen in the test set is not smooth free-flow traffic but rather a dense traffic situation with a lot of variation in velocities. This makes sense, since the data was recorded during the morning commute in Los Angeles, notorious for highly congested rush hour traffic.

Relating the performance of the baselines to the correlations displayed in Figure 4.2 one can see the impact of correlation. The first two seconds of prediction with the constant acceleration model is almost twice as good as the constant velocity method. This highlights the importance of correlation between the target values and its history and solidifies why it is a good idea to make use of recurrent connections when working with time series.

As can be seen in Table 4.2 the baselines are far outperformed by the single-task architecture. The mean absolute error at the prediction horizon 4 seconds is more than twice as good for the single-task architecture compared to the constant acceleration baseline and more than three times as good than the constant velocity baseline.

There is a relatively high uncertainty in the MAE value; this comes from the fact that there are some cases where the networks prediction diverges quite a bit from the true trajectory. This will be discussed in more detail later, here it will suffice

observing that the uncertainty is much lower in the case of the single-task prediction compared to the baselines, indicating that there are more extreme outliers in the baseline predictions.

In Figure 4.4 four cases of prediction is displayed, where the two top plots successfully predicts the vehicle acceleration, the bottom left plot shows a prediction that is good up until half of the prediction horizon and the bottom right shows a prediction that fails to correctly predict the characteristics of the true acceleration trajectory.

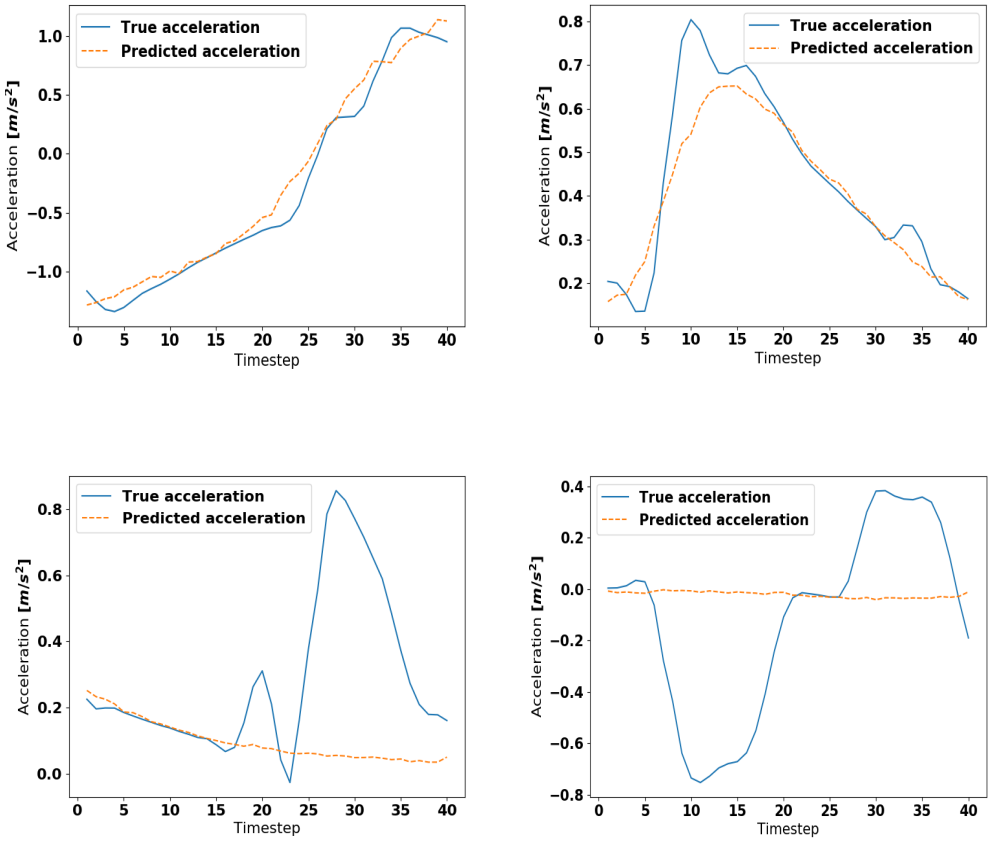


Figure 4.4: In the four plots above the networks predicted longitudinal accelerations is displayed alongside the true accelerations of the vehicle. The two top plots shows two instances where the acceleration has been successfully predicted. In the bottom left the predicted acceleration follows the true accelerations closely to start of with but diverges after about half of the prediction horizon. In the bottom right plot the predicted acceleration fails to follow the true acceleration.

4.2.1 Feature impact and prediction error

To see how the prediction changes with added features Figure 4.5 and Table 4.3 were generated. Due to the time it took for every training epoch this evaluation was only done for some of the features (for the relative importance of the surrounding vehicles see Section 4.2.2). The tested feature sets were the following:

- Set 1
  - Acceleration
- Set 2
  - Acceleration
  - Velocity
- Set 3
  - Acceleration
  - Velocity
  - Headway distance

Table 4.3: The evaluation of the longitudinal prediction for a different set of features. The displayed values are the mean absolute errors of the predicted trajectory from the true trajectory at four different prediction horizons.

Feature Set	Mean absolute error in m at time horizon			
	1s	2s	3s	4s
1	0.0619 ± 0.0796	0.347 ± 0.567	0.933 ± 0.863	1.84 ± 1.62
2	0.0600 ± 0.0771	0.338 ± 0.347	0.911 ± 0.838	1.80 ± 1.57
3	0.0561 ± 0.0720	0.292 ± 0.301	0.719 ± 0.668	1.35 ± 1.20

From the features that were analyzed in this exploration it is clear that the most influential one, apart from past acceleration values, is headway distance. Which makes sense from a human perspective. The likelihood that a person starts accelerating is very small if the driver in front is not accelerating, i.e the headway distance is not increasing.

The velocity feature had a minor but still noticeable impact on the prediction outcome. This means that the acceleration of a vehicle depends on its velocity. More explicitly it was found that the vehicle is more likely to accelerate more when the

velocity is low. This is mainly because busy traffic situations has a higher probability of allowing a scenario where a driver accelerates from 0 to 20 kph than a scenario where the driver goes from 70 to 90 kph. Also, lower velocity usually tends to result in more varying flow of traffic with short increases in velocity whereas a higher velocity usually has a more stable flow of traffic.

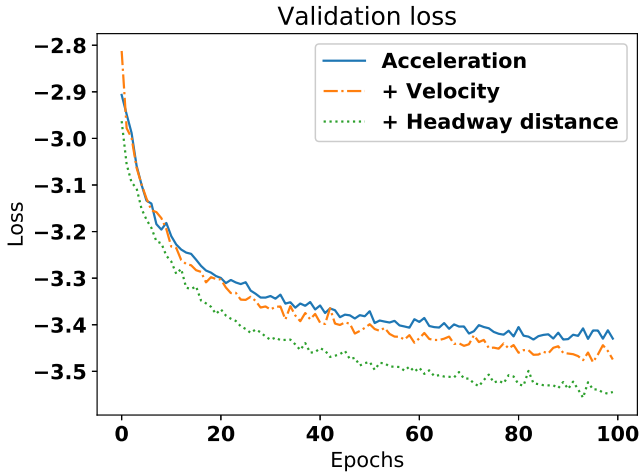


Figure 4.5: The validation loss during training for different feature sets in the longitudinal prediction model.

4.2.2 Robustness

To test the robustness of the network, the prediction model was evaluated on the test set where some history was hidden and thus approximated as the first seen sample. The importance of the full history was evaluated in Figure 4.6 where the MAE is calculated as a function of the number of hidden samples.

In the figure there is a clear decrease in accuracy as more and more samples get obfuscated. Looking at how the error increases with more hidden samples one sees that there is a significant increase in steepness towards the last 5 samples, 35-40 on the x-axis. This means that these last 5 samples (the most recent ones) have a much more significant importance in the prediction and not having these will result in a quick deterioration of the prediction. This relative importance can be explained by looking at the correlation graph in Figure 4.2. In this graph we see that the correlation for the last 5 samples of the longitudinal acceleration is all above 0.9, which is very correlated and thus provides very much information to the network.

We can also note that even in the case where the network is only able to observe one sample, it beats both the baselines. This highlights the power of a data driven approach where many variables are observed simultaneously and each contributes towards some objective that may appear uncorrelated.

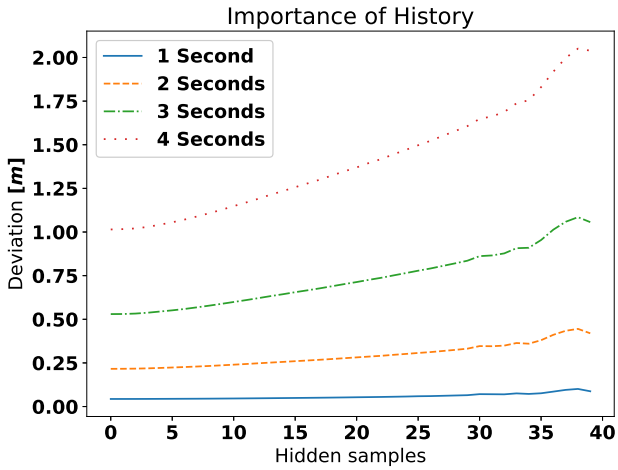


Figure 4.6: Figure showing the importance of the history of the vehicles considered in the prediction. There is a sudden loss in prediction when the network loses the information in the most recent 5 samples.

To continue the feature impact evaluation and discussion, we use the result presented in Table 4.4. The values presented here is of no real use other than as a relative measurement of importance of the features obscured in the scenario, because the network was trained on cases where all information was available and didn't have any obscured data in the training process, meaning that the result is very bad for the average case when obscuring one or multiple vehicles in the surrounding.

Table 4.4: Evaluation of the scenarios defined in section 3.3.1

Scenario	Mean Absolute Error $[m]$
I	3.44
II	12.9
III	5.70
IV	4.47
V	6.40

If we look at the table, we see that the largest MAE is achieved in scenario II, which is when the vehicle in-front is obscured. This ties back into our initial evaluation of the individual features, where the headway distance had a large contribution to the end result. So this is in accordance with our expectations.

The second most important vehicle, according to this evaluation, is the car to the back left. This car is the most important car when merging onto the highway or passing the vehicle in front on their left as we need to make sure we have a gap that we can squeeze in by controlling our longitudinal motion in accordance to that vehicle before making any lateral controls.

The third most important vehicle is the front right vehicle, which is the vehicle you would most likely consider the most important one when making a merge to the right and taking the exit lane. People in the right lane tend to have a lower speed than you, which would cause you to base your speed on the car on your front right over the back right car.

The remaining two vehicles do have an impact, but it's relatively small compared to the vehicle in-front.

4.3 Lateral Prediction

The evaluation of the lateral predictions was done slightly differently from the longitudinal model. Since the coordinate system used in this thesis is the one displayed in Figure 3.1 which follows the curvature of the road, the most interesting trajectories are the ones that involves lane changes. The result is presented in Figure 4.7. In the left plot the vehicle's current position in time before the lane change is shown on the x-axis, the y-axis shows the final error of the predicted trajectory. The right plot shows the overview of a vehicle positioned 2 seconds before the lane change with the prediction spanning to 2 seconds after the lane change.

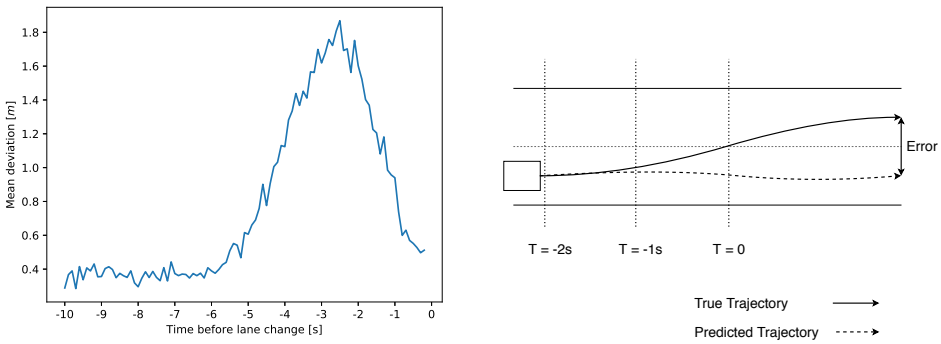


Figure 4.7: Prediction of lateral trajectories involving lane changes. In the left plot the deviation of the 4 second prediction horizon is plotted against each timestep leading up to the lane change. The figure to the right illustrated a situation that starts at -2s and plans until 2s. The error shown between the two trajectories is the corresponding y-value for the x-value of -2 in the left plot. The lane change is defined as the first instance the car passes over into the other lane with the intention of remaining there.

We see that as the time increases towards 0 (the instance at which the vehicle starts crossing over into the next lane) the error starts to increase. Specifically the

increase starts at 6 seconds prior to the lane-change (-6 on the x-axis). This means that there are some vehicles that will start to move towards the adjacent lane 2 seconds prior to the lane-change (since the prediction lasts 4 seconds) and that the network fails to predict these instances. From this one may conclude that predicting a lane-change more than two seconds before it happens is difficult, as there are no explicit movement indicating a lane-change before this point in time.

The point at which the increasing trend is reversed happens around 2.5 seconds prior to the lane-change (-2.5 in the figure), where the error starts decreasing with increased time. What this means is that more than 50% of the lane-changing trajectories have been identified at this point and the network predicts the lateral movement of the vehicles correctly in the majority of the cases.

As time increases from -2.5 towards 0, the error decreases, indicating that more and more lane change trajectories are identified by the network between these points in time.

In the Figure 4.8 there are four different cases of prediction of lane-changing trajectories on display. In the top two and the bottom left plots, the network successfully predicts that there will be lateral movement towards the adjacent lane. In the bottom right plot the network does not identify that the vehicle will change lane. The difference between the successful and the failed predictions is the speed at which the lane-change takes place. This was a common denominator between most failed predictions, the lane-change was an aggressive transition towards the adjacent lane.

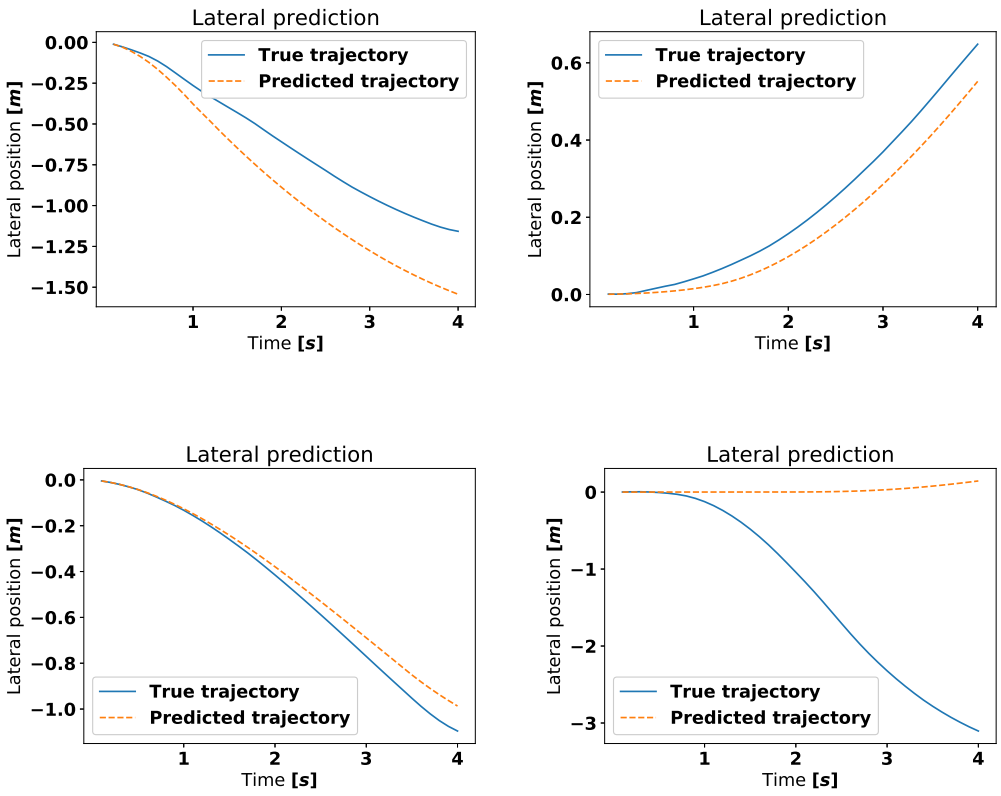


Figure 4.8: In the four plots above the networks predicted lateral trajectory is displayed alongside the true lateral trajectory of the car. The two top left and the bottom left has identified that a lane change is taking place in the near future, whereas the network fails to predict the lane change in the bottom right plot.

4.4 Multi-task learning

The longitudinal result of the multi-task architecture is similar to the single-task architecture and can be seen in Table 4.5. There is a slight advantage to the multi-task model when compared to the results of the single-task model presented in Table 4.3.

Table 4.5: The prediction error over time for longitudinal acceleration prediction

Architecture	Mean absolute error in m at time horizon [s]			
	1	2	3	4
Multi-Task	0.0418 ± 0.0599	0.216 ± 0.245	0.530 ± 0.548	1.03 ± 1.02

The lateral prediction is evaluated on the lane changing trajectories in the same manner as before, the result is plotted in Figure 4.9. The structure is very similar to what was shown in Figure 4.7 and the conclusion is that the single-task and multi-task models are equivalent in terms of lateral prediction.

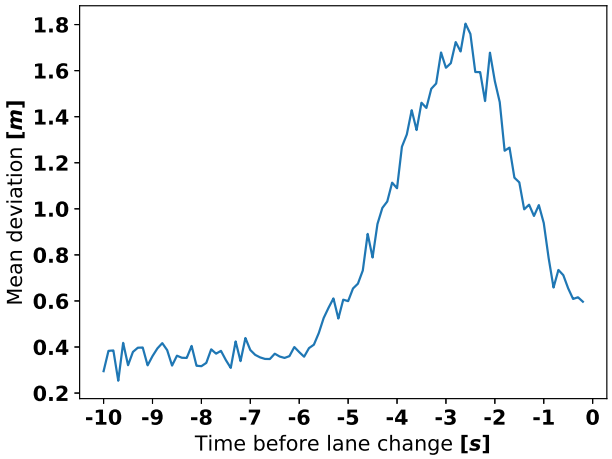


Figure 4.9: Deviation of the lateral prediction trajectory for trajectories involving a lane change

4.4.1 Driver Intent

To evaluate the performance of the intent classifier, it was evaluated on the lane-changing trajectories in the test set. The accuracy of the performance is shown in Figure 4.10, the predictions are matched with the classification metrics shown in Figure 4.11.

The result of the intent-prediction follows what we could read from the lateral evaluation. From Figure 4.7 it was concluded that the network had identified roughly 50% of the lane-changing trajectories at 2.5 seconds prior to the lane-change. If we compare this conclusion to the result of the intention prediction in Figure 4.10 we see that at 2.5 seconds before the lane-change the accuracy is around 50%. The accuracy increases until roughly 1 second prior the lane-change, where the accuracy stabilizes at around 95%.

In Figure 4.11 more detailed information is displayed on the predictions. For example, we can see that the precision of the predictions is high at all time, even at low accuracy, meaning that the network is conservative with giving out a label indicating intent of lane-change. However, there is a significant difference between left and right lane changes, where the right lane-change seem to be more difficult to identify compared to the left lane-changes. This may be explained by the fact that the data

include a merge lane, meaning that all vehicles observed in this lane, identifiable by the features *Has left lane* and *Has right lane* both being zero, will eventually make a left lane-change. Another reason why lane-changes to the right may be more difficult is because there is an exit lane as well. Since the drivers are human, it is likely that the exit lane comes as a surprise to some of the drivers, meaning that they make very sudden lane-changes to the right, without much planning.

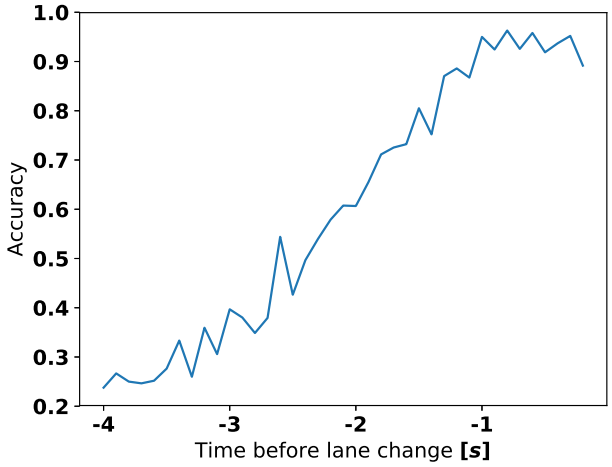


Figure 4.10: The classification accuracy of trajectories involving a lane change.

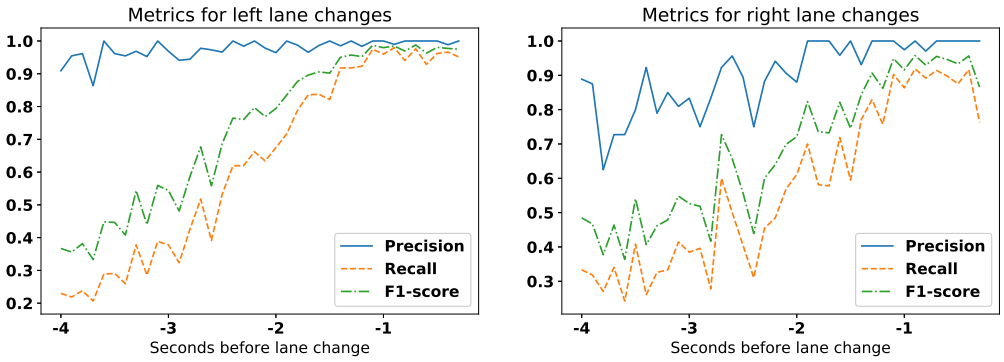


Figure 4.11: Precision, recall and f1-scores for the intent classifier during classification of the lane changing trajectories in the test set.

4.4.2 Introduction of indicator lights

The result from including a simulated indicator light in the prediction model is shown in the lateral lane change trajectories in Figure 4.12. In the figure it is clear

that the indicator light has an effect on the prediction. There is a noticeable shift of about 0.5 seconds between when the increasing error trend is reversed, from 2.5 seconds prior the lane-change without indicator lights to about 3 seconds with the lights.

This result follows what we expect. The indicator signals were derived from a normal distribution centered around 3 seconds before the lane-change, this means, from the definition of the normal distribution, that about 50% of the trajectories should have have turned on their indicator lights at this point in time. This was what we concluded from the figure and is further reinforced in Figure 4.13. In this figure we see that the number of indicator lights reaches 50% just after three seconds prior to the lane-change, just as expected.

As we can see in Figure 4.13 the prediction accuracy follows the curve of the distribution of indicator lights. This means that the network has found the correlation between the indicator light and the intent. More importantly, the network has almost always a higher accuracy than the indicator light distribution, meaning that the network does not solely rely on the indicators, it complements its basis for prediction with other information it receives from the environment.

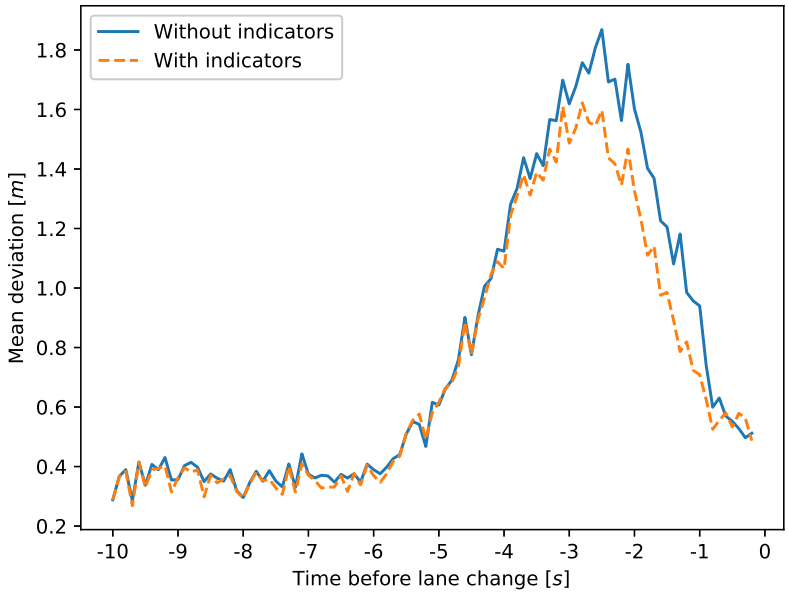


Figure 4.12: Comparison of the predicted trajectories with and without indicator lights in the multi-task model. The deviation is affected by the indicator lights feature as can be seen by the shift in about 0.5 seconds when the error starts to decline with increased time.

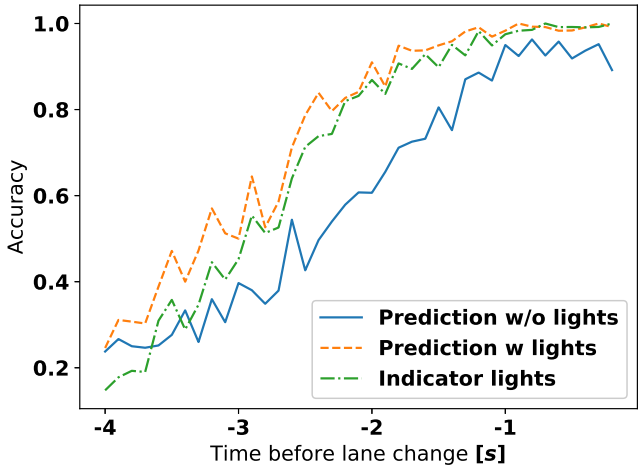


Figure 4.13: The accuracy of the driver intent prediction with indicator lights as a feature of the target vehicle.

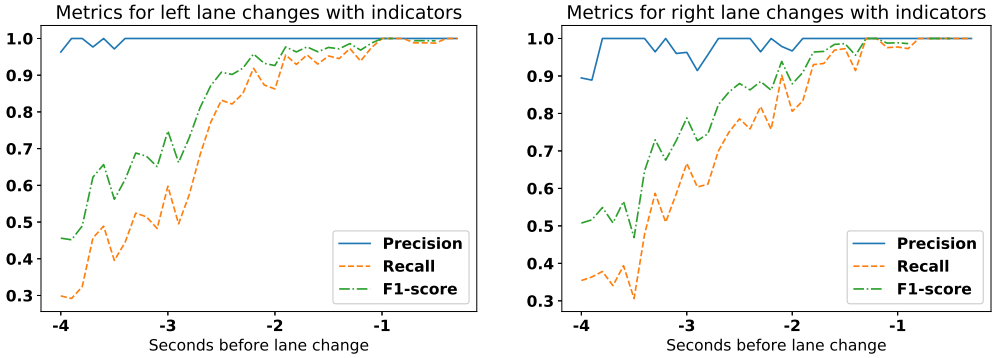


Figure 4.14: Precision, recall and f1-scores for the intent classifier during classification of the lane changing trajectories when the indicator light of the target vehicle was used as an additional input.

The introduction of indicator lights also affected the longitudinal predictions, as shown in the table below.

Table 4.6: The prediction error over time for longitudinal acceleration prediction using the multi-task architecture with artificial indicator lights.

Architecture	Mean absolute error in m at time horizon [s]			
	1	2	3	4
MTL Indicators	0.0404 ± 0.0585	0.207 ± 0.244	0.509 ± 0.544	0.977 ± 0.986

When compared to the result of both the single-task and the multi-task architec-

tures, there is a noticeable increase in accuracy in the longitudinal accuracy. If the intentions of a driver are easier to predict, the motion prediction will also have a positive impact.

4.5 Worst case model comparisons

The two models for longitudinal prediction is compared for three different types of trajectories. The upper 5 percentile for the trajectories yielding the highest deviation, highest acceleration/deceleration, and the highest jerk. These different subsets are then compared in the table below

Table 4.7: The worst case prediction error for the models at the full prediction horizon, 4s.

Trajectories	Metric	Constant Accel	Single-task	Multi-task
Deviation	Mean	7.90	4.15	**4.04**
	25%	6.77	3.34	**3.27**
	50%	7.43	3.76	**3.66**
	75%	8.54	4.53	**4.39**
	max	26.8	**21.5**	23.0
Acceleration	Mean	7.20	2.07	**2.05**
	25%	5.81	**0.68**	0.70
	50%	7.01	**1.51**	1.52
	75%	8.40	2.93	**2.88**
	max	26.8	**21.5**	23.0
Jerk	Mean	2.95	1.71	**1.70**
	25%	1.08	**0.59**	**0.59**
	50%	2.26	**1.26**	1.27
	75%	4.02	2.29	**2.28**
	max	26.8	**21.5**	23.0
Total		0	7	10

as we can see, the result of the single and multi-task architectures are very close, with a slight advantage to the multi-task architecture. More importantly, the baseline results of the constant acceleration are outperformed in all cases evaluated.

4.5.1 Correlation between uncertainty and deviation

To correlate the prediction with the model's own uncertainty measurement the scatter plots in Figure 4.15 were produced. In the plots the prediction error is displayed on the x-axis and the sum of the standard deviation of the model is shown on the

y-axis. It is hard to establish that there is a correlation between the total sum of the standard deviations and the actual error in the prediction. However, we can see the increasing trend if we specifically choose to look at the minimum uncertainty for each value along the x-axis. From this we can conclude that there probably is a correlation, but it is drowned in noise. One possible explanation for this is that the network is trained to predict the acceleration whereas the results are evaluated on positions. It is possible that the uncertainty measurement gets drowned in noise during this transition.

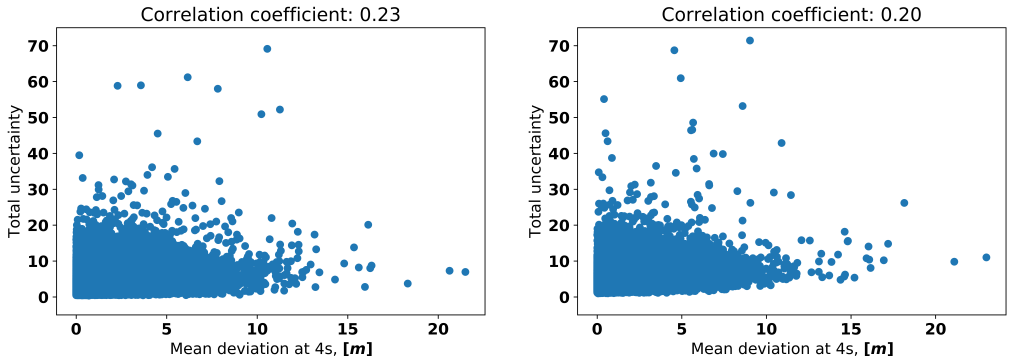


Figure 4.15: Scatter plots of the predicted mean deviation error and it's corresponding total uncertainty parameters in the mixture model. The left plot contains the data from the single-task architecture, and the right plot is from the multi-task architecture. In both figures the uncertainty tends to increase slightly as the deviation increases.

5

Discussion and Conclusion

5.1 Limitation of the Kalman filter

The Kalman filter is in itself a predictor, and one might wonder why there is a need for a neural network. The reason behind this is because although it is possible to do multi-step prediction with a Kalman filter there is a limit to how far this prediction extends with reasonable accuracy. A multi-step prediction done solely by a Kalman filter was found to be accurate up until 10-15 timesteps after which the predictions diverged and the full 40 timesteps prediction ended up being worse than constant velocity inference. This is where the data driven approach comes into play as it is possible to observe almost an infinite number of variables which all may affect the driver, whereas the Kalman filter relies solely on the physical movement of the vehicle.

A Kalman filter is still useful during the inference as the input to the network is eventually captured on the fly by on-board sensors, which all have some degree of noise. A Kalman filter can thus be used to filtrate this data according to the one-step prediction of the system to allow realistic acceleration and velocity values to be extracted to the prediction unit just as in this thesis.

5.2 Limitation of intention prediction

The intention prediction was quite accurate and it was possible to identify most vehicles before they switched lane, however, it was not possible to identify all vehicles without indicator lights. This is one of the limitations of predicting humans. Humans are not possible to predict in all cases, there will always be edge cases that will not be covered in the data or unobservable variables influencing the decision. This is why a 95 % percent accuracy is very good.

One of the potential variables that could be observed and have a noticeable effect on the accuracy is the indicator light of the vehicle in question. This was also shown by

introducing artificial lights with the outcome that we could identify close to 100% of the vehicle one second prior to the lane change. In the real world, this won't really be the case, as not everyone is going to use the indicator lights before changing lane, however, we should be able to see a noticeable increase in accuracy with this addition in real scenarios.

5.3 Single-task vs Multi-task

The result of the two models turned out to be fairly equal, with the multi-task model having a slightly better performance in the longitudinal evaluation.

One interesting difference between the two architectures was the speed of convergence. The results for the longitudinal single-task predictions were generated on a model that had been trained for 138 epochs compared to 104 epochs for the multi-task model. The reason for this speed difference may be that by introducing three different tasks the network learns a more useful abstract representation quicker as a more task-specific representation, a local minimum in the optimization, gets punished by the other tasks.

From the evaluation of the two architectures, there are two main reasons why the multi-task architecture is preferred over a single task one. The first reason is because the multi-task architecture allowed for a faster convergence in the training. The second reason is because it allows for a great reduction in total compute needed in the inference step as one forward pass allows for prediction of three different variables as opposed to only one with the single-task.

5.4 Conclusion

This thesis studied the application of deep learning on the problem of predicting irrational behaviours in human drivers. From the result one may conclude that achieving perfect prediction is near impossible, as capturing all parameters that form the basis for a human decision is not feasible. However, it was shown that a data driven approach where reasonable data, such as environmental observations of surrounding vehicles, outperforms simplistic prediction models, i.e. the constant velocity/acceleration, in the evaluated scenarios.

The longitudinal prediction is performing well in the average case where the simplistic models fail, i.e. in congested and dense traffic conditions. The average prediction error was around one meter after a full four second prediction horizon. Even in the worst cases there is valuable information to be learned from the prediction, since compared to the simplistic models the error of the network is less, meaning that the

network captures the tendency of most trajectories but to a varying degree.

The lateral prediction gets complemented by the intention prediction, where the intention does not necessarily improve the lateral performance, however the intention prediction is easier to interpret from a human perspective. The accuracy of the intention prediction increases rather linearly from 35% at three seconds before the lane change to around 95% at one second prior to the lane change.

The two architectures evaluated in this thesis turned out to be fairly equal in terms of performance, with a slightly better performance seen in the multi-task model. The real gain from using the multi-task model is the gain in saved compute power, both during training and during inference.

One of the reasons for choosing the mixture density model was the uncertainty parameters in the mixture model. As it turned out, it is not easy to make the neural network learn to correctly output a value for the uncertainty measure that accurately describes if the network is certain about the predicted trajectory in the evaluation of the network. It was difficult finding a correlation between the standard deviation of the distributions in the mixture model and the mean absolute error. This is partly due to the task at hand and partly due to the neural network. The task is so complex, there is no way to define what specific actions are correct at any point in time, as there are multiple correct actions, which is why the uncertainty might be large at any time and not just in the cases where the prediction is bad. The data are also part of the problem, as the network just learns what is present in the data, hopefully it generalizes pretty good from the available data, but there will always be situations where the human does not behave according to previous observations. This is one drawback of using neural networks.

As a final conclusion, even if producing a self-aware, in terms of uncertainty of its output, turned out to be difficult, the advantages of using a data driven approach outperforms the negatives. I see that the future of this type of prediction, where the target is inherently irrational and difficult to predict, will have some sort of data driven element to it. The field of machine learning and deep learning in particular is quickly evolving with new state-of-the-art results and methods being developed every week. This in combination with more and more data being recorded means that data driven approaches will become better and better, when and if the limit will be reached, only time will tell.

Bibliography

[1] Traffic Safety, AAA Foundation for: *American driving survey, 2015 – 2016*. 2017.

[2] Organisation, World Health: *Global status report on road safety 2015*. 2015.

[3] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun: *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. CoRR, abs/1502.01852, 2015. http://arxiv.org/abs/1502.01852.

[4] Keeney, Tasha: *Mobility-as-a-service: Why self-driving cars could change everything*. October 2017. http://research.ark-invest.com/self-driving-cars-white-paper.

[5] Schwarting, Wilko, Javier Alonso-Mora, and Daniela Rus: *Planning and decision-making for autonomous vehicles*. Annual Review of Control, Robotics, and Autonomous Systems, 1(1), 2018. https://doi.org/10.1146/annurev-control-060117-105157.

[6] U.S Department of Transportation: *Next Generation Simulation (NGSIM) Vehicle Trajectories*. https://data.transportation.gov/Automobiles/Next-Generation-Simulation-NGSIM-Vehicle-Trajector/8ect-6jqj, 2017. Online; accessed 30 January 2018.

[7] Esling, Philippe and Carlos Agon: *Time-series data mining*. 45:12, November 2012.

[8] Nielsen, Michael A.: *Neural Networks and Deep Learning*. Determination Press, 2015.

[9] Lipton, Zachary Chase: *A critical review of recurrent neural networks for sequence learning*. CoRR, abs/1506.00019, 2015. http://arxiv.org/abs/1506.00019.

[10] Werbos, P. J.: *Backpropagation through time: what it does and how to do it*. Proceedings of the IEEE, 78(10):1550–1560, Oct 1990, ISSN 0018-9219.

[11] Bengio, Y., P. Simard, and P. Frasconi: *Learning long-term dependencies with gradient descent is difficult.* IEEE Transactions on Neural Networks, 5(2):157–166, Mar 1994, ISSN 1045-9227.

[12] Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio: *On the difficulty of training recurrent neural networks.* CoRR, abs/1211.5063, 2012. http://arxiv.org/abs/1211.5063.

[13] Hochreiter, Sepp and Jürgen Schmidhuber: *Long short-term memory.* Neural Comput., 9(8):1735–1780, November 1997, ISSN 0899-7667. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[14] Greff, Klaus, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber: *LSTM: A search space odyssey.* CoRR, abs/1503.04069, 2015. http://arxiv.org/abs/1503.04069.

[15] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville: *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[16] Breiman, Leo: *Bagging predictors.* Mach. Learn., 24(2):123–140, August 1996, ISSN 0885-6125. http://dx.doi.org/10.1023/A:1018054314350.

[17] Statistics, Leo Breiman and Leo Breiman: *Random forests.* In *Machine Learning*, pages 5–32, 2001.

[18] Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov: *Dropout: A simple way to prevent neural networks from overfitting.* Journal of Machine Learning Research, 15:1929–1958, 2014. http://jmlr.org/papers/v15/srivastava14a.html.

[19] Ruder, Sebastian: *An overview of multi-task learning in deep neural networks.* CoRR, abs/1706.05098, 2017. http://arxiv.org/abs/1706.05098.

[20] Rasmussen, Carl Edward: *The infinite gaussian mixture model.* In Solla, S. A., T. K. Leen, and K. Müller (editors): *Advances in Neural Information Processing Systems 12*, pages 554–560. MIT Press, 2000. http://papers.nips.cc/paper/1745-the-infinite-gaussian-mixture-model.pdf.

[21] Zen, Heiga and Andrew Senior: *Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis.* In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3872–3876, 2014.

[22] M. Bishop, Christopher: *Mixture density networks.* U.K., Tech. Rep. NCRG/4288, January 1994.

[23] Kalman, Rudolf: *A new approach to linear filtering and prediction problems.* 82:35–45, January 1960.

[24] Faragher, R.: *Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes].* IEEE Signal Processing Magazine, 29(5):128–132, Sept 2012, ISSN 1053-5888.

[25] Kullback, S. and R. A. Leibler: *On information and sufficiency.* Ann. Math. Statist., 22(1):79–86, March 1951. https://doi.org/10.1214/aoms/1177729694.

[26] Bishop, Christopher M.: *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006, ISBN 0387310738.

[27] Allison, L.: *Normal, gaussian.* http://allisons.org/ll/MML/KL/Normal/.