Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Statistics and Machine Learning
2020 | LIU-IDA/STAT-A--20/012--SE

Vehicle Path Prediction Using Recurrent Neural Networks

Mim Kemal Tekin

Supervisor : Amanda Olmin
Examiner : Jose M. Peña

External supervisor : Erik Wernholt and Gustav Persson

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

Vehicle Path Prediction can be used to support Advanced Driver Assistance Systems (ADAS) that covers different technologies like Autonomous Braking System, Adaptive Cruise Control, etc. In this thesis, the vehicle's future path, parameterized as 5 coordinates along the path, is predicted by using only visual data collected by a front vision sensor. This approach provides cheaper application opportunities without using different sensors. The predictions are done by deep convolutional neural networks (CNN) and the goal of the project is to use recurrent neural networks (RNN) and to investigate the benefits of using reccurence to the task.

Two different approaches are used for the models. The first approach is a single-frame approach that makes predictions by using only one image frame as input and predicts the future location points of the car. The single-frame approach is the baseline model. The second approach is a sequential approach that enables the network the usage of historical information of previous image frames in order to predict the vehicle's future path for the current frame. With this approach, the effect of using recurrence is investigated. Moreover, uncertainty is important for the model reliability. Having a small uncertainty in most of the predictions or having a high uncertainty in unfamiliar situations for the model will increase success of the model. In this project, the uncertainty estimation approach is based on capturing the uncertainty by following a method that allows to work on deep learning models. The uncertainty approach uses the same models that are defined by the first two approaches.

Finally, the evaluation of the approaches are done by the mean absolute error and defining two different reasonable tolerance levels for the distance between the prediction path and the ground truth path. The difference between two tolerance levels is that the first one is a strict tolerance level and the the second one is a more relaxed tolerance level. When using strict tolerance level based on distances on test data, 36% of the predictions are accepted for single-frame model, 48% for the sequential model, 27% and 13% are accepted for single-frame and sequential models of uncertainty models. When using relaxed tolerance level on test data, 60% of the predictions are accepted by single-frame model, 67% for the sequential model, 65% and 53% are accepted for single-frame and sequential models of uncertainty models. Furthermore, by using stored information for each sequence, the methods are evaluated for different conditions such as day/night, road type and road cover. As a result, the sequential model outperforms in the majority of the evaluation results.

Acknowledgments

I would like to express my deepest appreciation to my supervisor at Linköping University, Amanda Olmin, for all guidance and feedback in every detail throughout the thesis. Thanks for the time you dedicated to supervising the project.

I would also like to express my gratitude to my supervisors at Veoneer, Erik Wernholt and Gustav Persson, for their support and all their experience that they shared with me. Additionally, thanks to Veoneer for providing me the data that is used in this project.

Many thanks to the advice and comments given by my examiner, Jose M. Peña, and my opponent classmate, Thijs Quast, during revision meeting. They have been a great help in the improvement of the thesis report.

I owe my gratitude to Andreas Stasinakis and Stefano Toffol for their great collaboration and true friendship during whole master program. Thanks for always being there in my best and toughest moments.

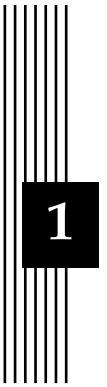
I would like to thank to my girlfriend, Büşra, for the continuous support during whole time and being patient even when I am so stressed and grumpy. Thank you for standing by me with all of your love.

Finally, I would like to express my greatest gratitude to my parents for their encouragement and all support. This experience would not have been possible without their sacrifices and trust. Therefore, I dedicate this thesis to you.

Contents

List of Figures

List of Tables



1 Introduction

1.1 Motivation

In recent years, the majority of car manufacturing and technology providing companies invest a significant amount of resources and time in the development of self-driving vehicles. Autonomous vehicles have potential benefits when it comes to maintaining road safety and reducing traffic accidents caused by human mistakes. Another advantage is that they can provide extra mobility for people with disabilities [22]. The biggest challenge of designing autonomous vehicles is satisfying the safety for the driver as well as other people. Therefore, safety systems for vehicles are important topics to consider when designing autonomous vehicles.

Veoneer is a technology supplier that designs, manufactures and sell state-of-the-art software, hardware and systems for occupant protection, advanced driving assistance systems (ADAS), and collaborative and automated driving to vehicle manufacturers globally. These systems are based on a number of sensors, such as vision, radar, LiDAR and thermal sensing. Veoneer also develop deep learning-based state-of-the-art methods and combine these with their hardware [1]. Another focused field of Veoneer is ADAS that assist the driver to provide safer traveling by the systems such as Autonomous Emergency Braking, Adaptive Cruise Control, and Lane Keep Assist, etc. In order to make ADAS be able to control these systems, it is necessary to have reliable supportive modules. Lane detection and object detection are examples of methods that can be used to help ADAS.

Vehicle path prediction can also be used to improve ADAS since it provides better information about the location of the vehicle in the near future. This system can foresee dangerous scenarios such as an object which is about to enter the vehicle's path, high speeds or inappropriate steering on bendy roads. Moreover, the system can provide warnings in these situations or if it is necessary, apply proper intervention. Therefore, all of these improvements in better handling with risky situations make this topic an interesting research field.

In this project, we propose a sequential deep learning approach to make vehicle path predictions with higher accuracy and precision. The sequential approach can be compared with a single-frame approach [23]. In the single-frame approach, the model is built in a structure that can be trained with an input that is only one image collected from the front vision sensor

[1] https://www.veoneer.com/en/who-we-are

of the vehicle and the corresponding true path prediction that is mentioned as ground truth in next chapters. This means that the model learns from single-frames and it does not have information about historical directions, which is not typically the case in real-life scenarios. This approach might perform well in the case of straight roads or roads with clear curves, but when the car is going to make a turn or change lanes, the lack of historical information may hurt the performance. As the vehicle's movement is dependent on previous locations and the directions, this information can improve the accuracy and provide a better certainty. Another factor that may harm the performance is having unclear visual features like invisible lanes, night conditions, current light directions, etc. In these cases, having a historical memory of captured visual feature can improve the performance.

1.2 Aim

The goal of this project is to build and evaluate a sequential model that learns the patterns between the frames that are collected by vision sensors and predicts the vehicle path. Recurrent Neural Networks (RNN) are suitable for this task, since they have the ability of dealing with sequential data. Moreover they can learn dependencies in a sequence by using the historical information through its memory. Vehicle path prediction task can be achieved with a single-frame approach. The single-frame approach is implemented by Veoneer [23], but this project also includes reproducing the single-frame approach in order to compare with the sequential approach fairly in the sense of accuracy and uncertainty. Moreover, this project aims at estimating uncertainty in the predictions, in order to give a better understanding of the reliability of the point estimator model. The evaluation and comparison of these methods are done by metrics that are described in detail in Chapter 4.

Therefore, the research questions of this thesis are:

1. How does the historical information that is captured by RNNs affect the performance compared to single-frame approach?
2. Can the model be built to reliably estimate uncertainty in the path predictions?

1.3 Related Work

In this project, the goal is to make predictions by learning historical movements from sequential image data and investigating whether this approach increases the performance compared to the approach that uses only single-frame inputs. Vehicle path prediction has been studied by using different machine learning and deep learning models with different types of data structures before. Previous studies with sequence learning approaches are done to predict car trajectories with a data that is collected from Lidar scanner that is mounted on a fixed point and the approaches follow a bird's-eye view. Since only one scanner is used, this dataset contains only the location information of the vehicles. However, the data used in this project is collected from the front vision sensor of the vehicle. The use of only visual data for this task gives the opportunity to apply this system cheaper and simpler. Using directly visual data helps to solve some security tasks like detecting objects that are about to enter the vehicle path or better perception of the physical conditions of the road and this solutions are achieved by only one model.

Zyner et al. [26] use Recurrent Neural Networks (RNN) with a location data that is collected and transformed from a traverse on roundabout to predict the vehicle paths and driver intention. The data is collected from Lidar scanners and the whole collection includes x/y positioning, velocity, heading angle, size and a classification for the vehicle type with a confidence [2]. Therefore, the data that is used is not a visual data as this project deals and vehicle

[2] http://its.acfr.usyd.edu.au/datasets/five-roundabouts-dataset/

prediction on a roundabout is a more specific purpose than this thesis. Additionally, Long Short-Term Memory (LSTM) is used with same data in order to predict driver intention by Zyner et al. [25]. These researches do not share same aims with this thesis, however they show that models based on RNN and LSTM can learn historical dependencies and predict a future state.

A good example of learning historical dependencies with LSTMs from visual input sequences is the image and video captioning task [2]. Donahue et al. [2] propose a structure called long-term recurrent convolutional neural network (LRCN) that extracts visual features from the sequential frames in the video and trains these features with an LSTM to generate some textual captions for the video.

2 Theory

2.1 Feedforward Neural Networks

Feedforward Neural Networks (FNN) are one of the essential concepts in deep learning models. In the simplest form, the goal is to learn a mapping between input x and output y by an approximation function. This mapping can be represented as $y = f(x;\theta)$ and the learning is achieved by finding best fitting θ parameters which are known as weights of the network. The most characteristic property of FNN is as in its name, the information flows from input to output without any feeding backward [5].

An FNN consists of layers that contain many units as illustrated in Figure 2.1. Each layer applies a function to learn the mapping. The FNN in Figure 2.1 can be formulated as Eq. 2.1 where $f^{(i)}$ is the layer operations, x is the first layer called the Input layer, $f^{(1)}$ is the Hidden layer and $f^{(2)}$ is the Output layer that generates y values. For each units in Hidden layer, calculation can be done as where $W^{(1)}$ is the weights of Hidden layer, $b^{(1)}$ is the bias values for Hidden layer. With this formulation in the layers, a FNN is solely able to learn linear problems. In cases of nonlinear problems, these layers should be activated with an activation function that is denoted as $\Phi(.)$. To be more specific, some of the possible activation functions are shown in the Eq. 2.5 and 2.7. Finally, the formula defined previously can be rewritten as Eq. 2.3, when dealing with a non-linear problem. These functions provide the ability to the FNN to learn nonlinear and more complex tasks. Activation functions can be applied even in the Output layer, if necessary. The figure and the formula can be extended for an increased amount of Hidden layers, so that the neural network can learn to solve more complex problems.

$$y = f^{(L)}(...f^{(2)}(f^{(1)}(x))) \tag{2.1}$$

$$z_n = W^{(n)}x + b^{(n)} \tag{2.2}$$

$$z_n = \Phi(W^{(n)}x + b^{(n)}) \tag{2.3}$$

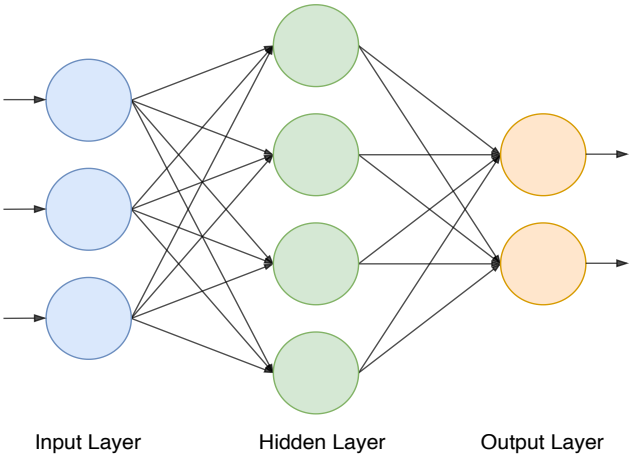


Figure 2.1: Simple feedforward neural network.

The previously mentioned operation is one forward pass of a FNN. In this way, FNN calculates the outputs with the current parameters and weights. This is not enough to approximate the best parameters and Rumelhart et al. [18] suggested a learning algorithm called backpropagation for optimizing the parameters. Backpropagation is done by calculating error gradients with respect to network parameters and these gradients are used to update the parameters. The first step of obtaining error gradients is calculating the error between predicted values from the forward pass and true output values. This error function is called the loss function and it can be Mean Absolute Error (MAE) for a regression task. Finally, the error gradients can be calculated with respect to parameters in each layer. After obtaining the gradients for each parameters, parameters can be updated by Stochastic Gradient Descent (SGD) algorithm [17] as formulated in Eq. 2.4 where t is the iteration count, i is a numerator for the training samples or sets of training samples and α is the learning rate. The learning rate (α) controls the step size between gradients. In this project, ADAM [8] algorithm is the optimization algorithm that updates the weights, and Cosine Learning Rate scheduler is used to control learning rate over iteration counts. Cosine Learning Rate scheduler is reviewed in detail in following sub-sections.

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial L_t(y^i, f(x^i)))}{d\theta_t} \tag{2.4}$$

Activation Functions

Activation functions are used to enable solution of solving non-linear problems. Without activation functions, neural networks can only learn linear problems as mentioned before, therefore they are necessary to solve more complex problems. Some of activation functions are defined as follows:

Sigmoid Function

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{2.5}$$

Hyperbolic Tangent Function

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.6}$$

ReLU Function [13]

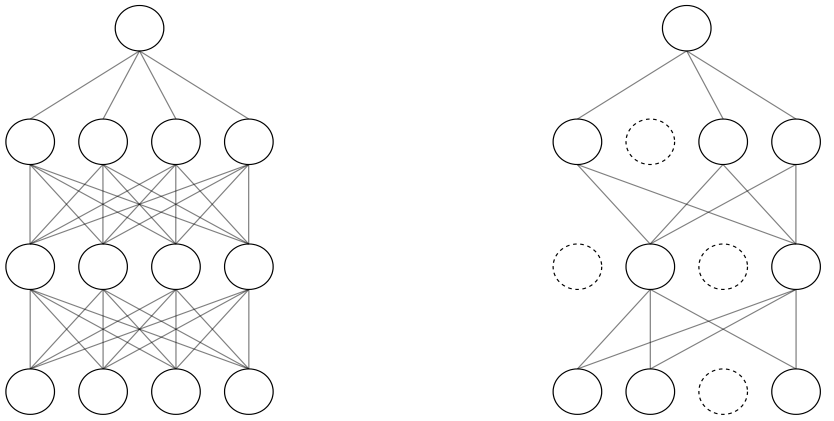
$$ReLU(x) = \max(0, x) \tag{2.7}$$

Regularization Techniques

There are some techniques that can be applied to optimize the learning and increase the generalization of the deep learning models. In this subsection, the techniques that are used in this project are reviewed. Specifically, regularization techniques support the learning systems in the task of generalization on unseen data. Depending on the amount of the training data, models can suffer from overfitting or underfitting. These problems cause a bad performance on test data, since the model does not generalize well. In order to overcome this issue, it is common to use different regularization techniques.

Dropout

Dropout technique [21] is an approach that provides solving a problem by passing the inputs through different combinations of units in the current architecture of the neural network. It evaluates the network with some excluded units for specific forward passes. The units that are removed are selected by a probability p. This approach is visualized in Figure 2.2. As can be seen, the complete neural network has all of the units fully connected, however after applying dropout some of the units are dropped with the probability p. After applying this method during training, the model has better generalization on unseen data, since the model is forced to learn the task by different sub-structures of the neural network.



(a) Complete neural network. (b) After applying dropout.

Figure 2.2: Dropout technique.

Learning Rate Scheduling: Cosine Annealing

Learning rate scheduling is a technique that optimizes the learning process. During a standard neural network training, usually a constant learning rate is used. This provides a constant learning speed and gradient steps that may cause unstable learning after some epochs and affect the learning curve visibly. A learning rate scheduler can adjust the learning rate during the training so that the later epochs can learn more efficiently by avoiding taking long steps in the gradient space. In this project, Cosine Annealing [11] is used to regulate the learning rate in all models after making experiments with a constant learning rate and Co-

sine Annealing. The learning rate α_t at epoch t is calculated by Eq. 2.8 where T is the total number of epochs.

$$\alpha_t = \alpha_{t-1} \cos\left(\frac{\pi t}{2T}\right) \tag{2.8}$$

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) [10] are a special type of Feed Forward Networks that can process grid-like data structures more efficiently and effectively. Images can be an example of grid structured data, therefore CNNs are often used in image processing tasks. An image data can be defined as a 3D-matrix of shape $(H\text{x}W\text{x}C)$ where H is height, W is width and Cs are channels. Each value can be either a color value in $[0, 255]$ or a scaled version in $[0, 1]$ for the corresponding color channel. Later, these values are processed by computer and this is called image processing. Image processing can be computationally expensive when fully connected layers are used as in Figure 2.1, since the number of parameters will be large. CNNs handle this problem by reducing the number of parameters that will be trained with shared parameters for the layers. These parameters are called kernel that is a matrix used in convolution operation over the input.

CNN is commonly used in visual feature extraction tasks since they have the ability to capture edges or common characteristics in the data. In this way, a neural network can be trained to extract the crucial features to solve a task. For example, this can be ear and nose shapes in a cat and dog detection task. Besides, in the best practice, CNN is used with activation functions and pooling operations that is reviewed in Section 2.2 in more detail.

Convolution Operation

The convolution operation is achieved by moving the kernel along horizontal and vertical directions of the grid structured data/image. While the kernel is moving, all pixels that are covered by the kernel in the image are multiplied element-wise with the kernel pixels and the resulting values are summed. In this way, one pixel of the resulting matrix is obtained. The step length of the kernel is controlled by the stride parameter. For instance, stride=1 means that moving the kernel by 1 pixel. This operation is visualizes in Figure 2.3. Finally, the calculated output is a filtered version of the input and the type of visual feature that is captured depends on the values of the kernel elements. The training is done to learn extracting useful visual features to solve the target task.

Pooling Operation

Pooling functions apply a summary statistics function to a rectangular neighborhood [5]. One of the common pooling functions, which is also used in this project is the max pooling [24] function that returns the maximum output that is within the rectangular neighborhood. Figure 2.4 shows how pooling is applied with a 2x2 filter and stride=2. As can be seen in the figure, the size of the image is reduced. By the help of the stride, the input is downsized with respect to height and width. This is a very common method to apply downsampling in order to capture meaningful and more complex features in a CNN.

2.3 Recurrent Neural Networks

A Recurrent Neural Network (RNN) model [18] is another type of neural network that is specialized in sequential data. The main difference compared to an FNN is the ability of dealing with sequential inputs, enabling processing of multiple inputs in order to make a prediction. A RNN has the ability to use historical information to predict future states. RNNs

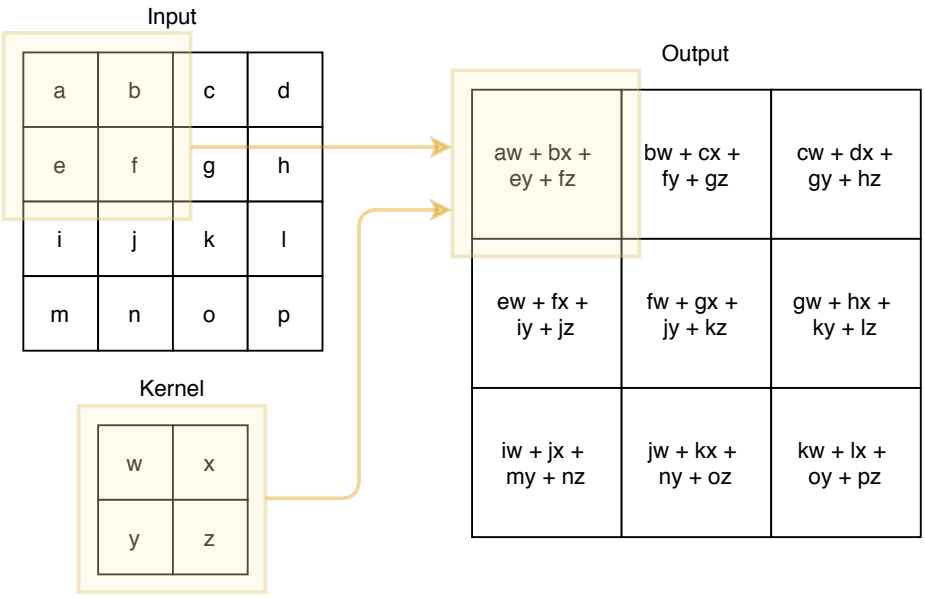


Figure 2.3: Convolution operations.

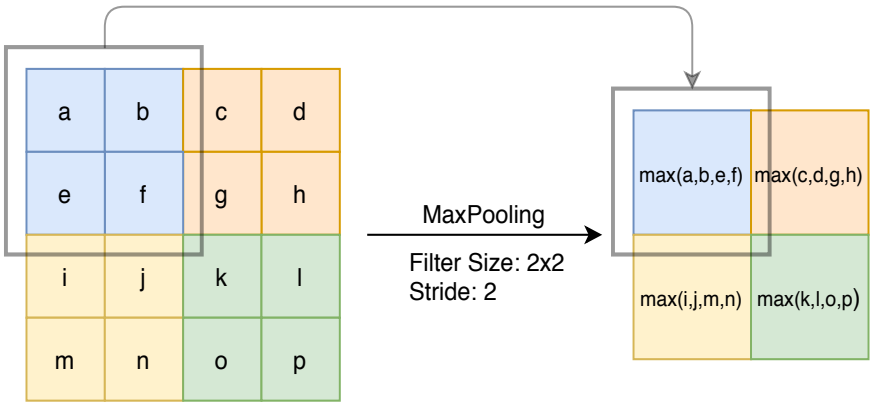


Figure 2.4: Pooling operations.

consist of computational elements that are called RNN cells and each cell takes one input in the sequence and the historical memory. In this way, the model can capture the relations or patterns between timesteps. RNNs can also deal with different sizes of inputs since they use a shared parameter strategy, so that the input size does not change the parameter count. In Figure 2.5, $x^{<t>}$ represents each input in the sequential data where t is the timestep, $a^{<t>}$ is the activation that is transferred and processed over the cells. Activation carries the historical information that is captured in every cell.

In the RNN cell, there are weights for input, activation, and output. These parameters are shared for each cell as mentioned before, so that the network learns only these parameters and uses it for each input element in the sequence. The transferred memory between cells is initialized as a zero vector as default and it is calculated by Eq. 2.9.

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{2.9}$$

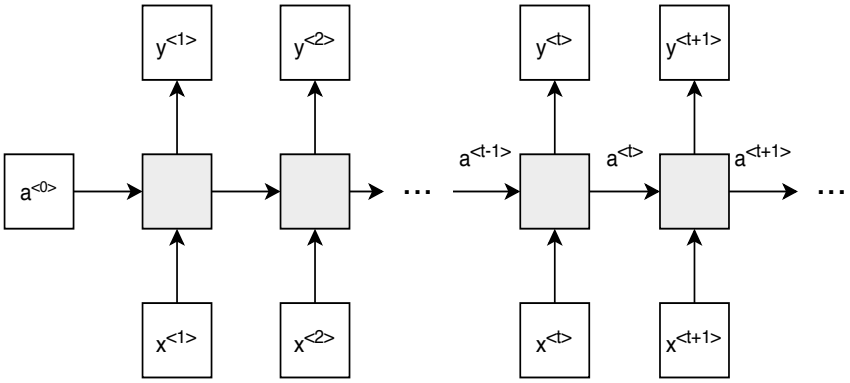


Figure 2.5: Recurrent Neural Network.

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \tag{2.10}$$

Finally, the output of the cell is calculated as in Eq. 2.10. In both Eq. 2.9 and 2.10, $g_{1,2}$ are activation functions, $b_{a,y}$ are biases and $W_{aa,ax,ya}$ are weights where a stands for activation, x for input and y for the output.

A significant drawback of RNNs is that they suffer from vanishing and exploding gradients and this problem is analyzed by Bengio et al. [1] in detail. Hochreiter et al. [6] proposed Long Short-Term Memory RNNs in order to deal with long term dependencies in the data by splitting the transferred activation into two-components as described in the next section.

2.4 Long Short-Term Memory RNN

Long Short-Term Memory (LSTM) [6] overcomes the gradient vanishing problem that RNN has. The gated structure of LSTM splits the short term and long term dependencies, so that the model has a better ability to deal with longer input data sequences. LSTM also has cells and in each cell, it has 4 gates that deal with different operations: (1) Forget gate, (2) Input gate, (3) Cell state, and (4) Output gate. The sequential architecture of LSTMs are shown in Figure 2.6. The learned historical information is transferred with cell state and during this process, it will be modified by other gates. There are some operations that provide learning and forgetting effects in the gates. These gates and the operations are annotated in Figure 2.7 and it is also visible to see that one LSTM cell has 2 inputs and 2 outputs in horizontal direction except for the data input x_t where t detonates time step. The upper input-output is called long term memory (LTM) and represented as C_{t-1} and C_t. The lower input-output is called short term momory (STM) and represented as h_{t-1}, h_t in the formulas. In the next sections, the operations in the gates of an LSTM cell are discussed.

Forget gate

The forget gate decides which information from LTM will be remembered or forgotten. Input x_t and previous hidden state h_{t-1} are concatenated and passed through the sigmoid function to decide which values will be updated. Since the Sigmoid activation function (σ) maps the input values into a (0, 1) space, it can be used to determine which values will be updated. More specifically, the values that are close to 0 are forgotten and the values are close to 1 are remembered value. In this way, the network forgets several values according to the information that is obtained from the STM and current input. In other words, the network forgets

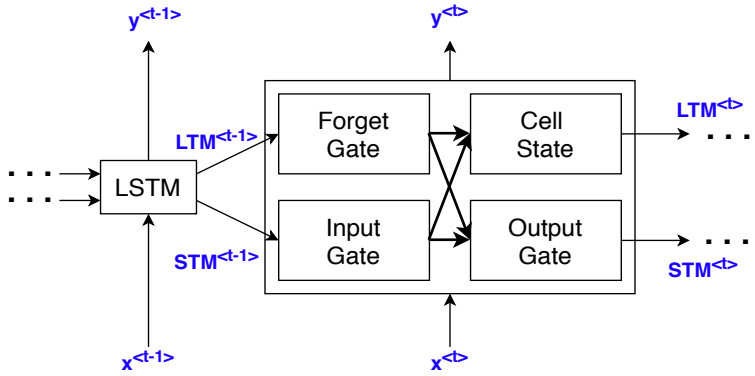


Figure 2.6: Long Short-Term Memory RNN where LTM is long-term memory, STM is short-term memory, x is input, y is output and t is the number of timestep

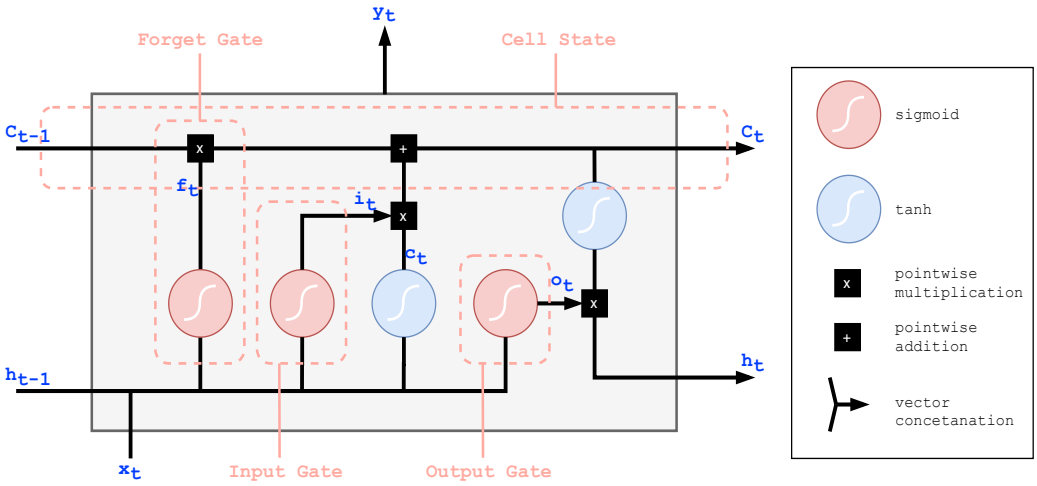


Figure 2.7: A Long Short-Term Memory Cell in detail.

irrelevant history with short memory and current input. This gate is formulated also in Eq. 2.11.

$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f) \tag{2.11}$$

Input gate

In the input gate, similarly to the forget gate, the concatenated x_t and h_{t-1} vector is passed through the sigmoid function in order to decide which values will be kept to learn from the STM and the current input. The values that will be learned is calculated as Eq. 2.12 where W_i and b_i are the weights and the bias values for the input, x_t is the current time step input and h_{t-1} is the previous time STM. After this operation, the same concatenated vector passes through a tanh function to map the values into a (-1, 1) space and helps the network to regulate. This operation is formulated as Eq. 2.13 where W_c and b_c are the weights and the bias values for this operation. The resulting i_t and c_t values are stored and used in the Cell state in order to apply the learning to LTM. c_t can be interpreted as a vector that influences

the LTM (C_t) and i_t can be interpreted as the vector that chooses which and how much the values of c_t will be learned.

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i) \tag{2.12}$$

$$c_t = \tanh\left(W_c[x_t, h_{t-1}] + b_c\right) \tag{2.13}$$

Cell state

The cell state can also be called Remember state since it generates the new LTM from the output of forgotten values (Forget Gate) and learned values (Input Gate). In this way, operations of forgetting some information from long term memory and learning from the new input with short memory operations are combined and used to generate the new LTM (C_t) based on the previous value C_{t-1}. This objective is achieved by Eq. 2.14 where f_t is the output of Forget gate, C_{t-1} is previous long term memory, i_t and c_t are the outputs of Input gate.

$$C_t = f_t * C_{t-1} + i_t * c_t \tag{2.14}$$

Output gate

Finally, the network has all of the information that is necessary for the output calculation. Firstly, the standard output gate value is calculated by Eq. 2.15 where W_{xo} and b_o are the weights and the bias values for the output. Secondly, the resulting vector (o_t) is multiplied element-wise with new LTM ($\tanh(C_t)$) in order to take the long term memory into account for the final output of the cell as formulated at Eq. 2.16.

$$o_t = \sigma(W_{xo}[x_t, h_{t-1}] + b_o) \tag{2.15}$$

$$h_t = o_t * \tanh(C_t) \tag{2.16}$$

2.5 Modeling Uncertainty in Deep Learning

Reliability in predictions is an important property of the models that deals with safety systems. One purpose of building a vehicle path prediction model is supporting the current ADAS systems which can be directly related to safety. Therefore, uncertainty estimation is an interesting investigation that shows whether the models can also report how much the predictions are certain or uncertain. In this task, since these models can be critical systems, the desired result is high accuracy and low uncertainty in the predictions. However, also it is important that detecting the predictions that are not reliable due to the uncertainty, so that the system does not take action under uncertain conditions.

Deep learning regression models that use a point estimation approach typically do not capture uncertainty. Uncertainty can be divided into two subtypes: (1) Epistemic Uncertainty and (2) Aleatoric Uncertainty [7]. While epistemic uncertainty captures the uncertainty in the model, aleatoric uncertainty captures uncertainty in the observations. Moreover, aleatoric uncertainy can be modelled in two different ways as (i) homoscedastic uncertainty that is defined as a constant uncertainty for different inputs and (ii) heteroscedastic uncertainty that

has the ability to capture variable uncertainty for different inputs. In this task, heteroscedastic uncertainty is more important, since it is able to estimate the uncertainty for each input.

Epistemic uncertainty can be estimated by using Bayesian Neural Networks (BNN) [15, 12, 14]. BNNs aim to fit a distribution over the weights instead of having a point estimation. In this way, all weights have uncertainty that can affect the output and be used to estimate the model uncertainty. This approach can be very computationally expensive in deep neural network models that contain many parameters. This computational requirement makes it very hard to achieve in practice.

Gal et al. [4] shows that applying a dropout layer before every weight layer gives a Bayesian approximation of Gaussian Processes (GP). The method is referred to Monte-Carlo Dropout and it runs more efficient than BNN. This method is based on the dropout regularization technique and Monte-Carlo sampling method. Dropout is reviewed in Section 2.1 under Regularization Techniques. In Monte-Carlo Dropout method, inputs are passed through the network several times with different dropout masks and the final prediction is made by averaging these samples. Additionally these samples can be used to estimate uncertainty in the model that is explained by variational inference technique [3]. In the implementation, Monte-Carlo Dropout can be formulated as Eq. 2.17 where T is the sample count and $f(x, p)$ denotes one forward pass with a fixed p dropout rate during the test run.

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f(x, p) \tag{2.17}$$

In order to capture aleatoric uncertainty, the noise parameter σ should be approximated. As mentioned before, the uncertainty is constant for homoscedastic uncertainty which means σ is fixed for every input point. On the other hand, for heteroscedastic uncertainty, σ should be learned for every input since this type of uncertainty assumes that noise is variable for each data observation [16, 9]. Under the assumption of having the output as Gaussian distributed, in order to achieve the learning of standard deviation and estimate the aleatoric uncertainty, the loss function of the neural network can be replaced with a Gaussian negative log-likelihood. As a result, by this loss function the model has the ability of adjusting the variance (σ^2) by predicting the mean of the Gaussian distribution. The loss function is defined in Eq. 2.18. Each location point is an unary Gaussian distributed component and D in the equation represents the count of these components. Additionally, i is an indicator of the observations in the dataset.

$$L_{nn}(\theta) = \frac{1}{D} \sum_{i} \frac{1}{2} \hat{\sigma}_i^{-2} ||y_i - \hat{y}_i||^2 + \frac{1}{2} \log \hat{\sigma}_i^2 \tag{2.18}$$

Moreover, epistemic and aleatoric uncertainty models can be combined to capture both of them. In this case first aleatoric uncertainty model should be built. After training the aleatoric uncertainty model, Monte-Carlo Dropout method should be applied by sampling T predictions from the network that can be described as $\hat{y}_t, \hat{\sigma}_t = f^{\hat{W}_t}(x)$ where t is the sample number and $\hat{W}_t$ denotes dropout mask usage, so that sampling runs over networks that have different weight combinations. Finally, the predictive uncertainty can be formulated approximately as Eq. 2.19.

$$\mathrm{Var}(y) \approx \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t^2 - \left(\frac{1}{T} \sum_{t=1}^{T} \hat{y}_t \right)^2 + \frac{1}{T} \sum_{t=1}^{T} \hat{\sigma}_t^2 \tag{2.19}$$

As these uncertainties can be combined, they can be modelled individually as well. The affects of these modelling choices are shown in Kendall et al. [7]. Epistemic uncertainty is

more suitable to model the uncertainty on a small dataset, but this can be affected by the observations that are not seen during training. Moreover, epistemic uncertainty is more computationally heavy since it is based on sampling and it requires multiple runs of the network for just one output. In the cases of having a big dataset, aleatoric uncertainty can explain most of the uncertainty and it can be used for real-time applications since it captures the noise in the data with only one model [7]. The choice of modelling depends on the task and the system constraints and when only one of them is modelled properly, previous research has indicated that the modelled uncertainty tries to compensate and maybe capture the other type of uncertainty too [7].

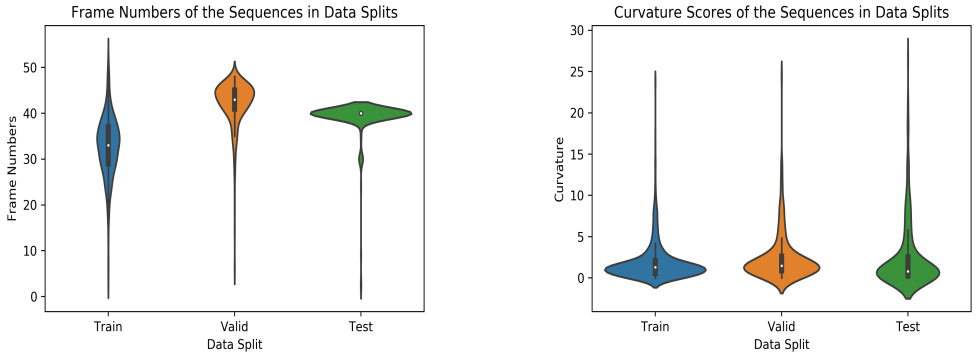
3 Data

This chapter describes the data that is used in the project. The data is provided by Veoneer and it has been collected in the field by a fleet of data collection vehicles. The data can be considered as video files that contain 600 frames (images) for each on average. A recording typically consists of several consecutive video clips. Each image frame in a video is sized as 400x1024 with 3 colour channels. The target data (ground truth) constitutes 5 (x, y) location points in Cartesian coordinate system every 10 meters starting from 15 meters ahead of the vehicle for one frame in the video. Additionally, there are some anonymous variables that are not evaluated in this thesis, but are involved in the training processes. The ground truth data is generated and supplied by Veoneer.

The data is split into three different subsets by sampling from the very large data collection. First, the videos are sampled for training, validation, and test data splits. Important to consider is that while sampling, videos recorded on the same day should not be present in more than one subset. In this way, the resulting splits do not have many similar observations during training and the model can be validated and tested on the data that has not been seen before.

The distributions of the frame counts and the curvature scores for data splits are visualized in Figure 3.1. The curvature scores are calculated as the average curvature for each video, and this score is described in Section 4.5 in more detail. As it can be observed in Figure 3.1b, the curvature score distributions of data splits are almost the same. Additionally, Figure 3.1a shows that Train data has more variety in frame counts in the videos while Validation and Test data has almost same amount of frames in the sequences.

In this project, two different model structure approaches are trained by using the same data. The first model is based on a single-frame approach, used as a baseline, and the second model is the sequential approach. For these specific purposes, it is necessary to define two different data structures that are described in following sections. The dataset sizes for both of the sturctures are shown in Table 3.1. Additionally, sequence counts by each category and label in the datasets are given. In the categorical data, *None* labels are not considered during evaluation. This does not affect the conclusion, since the models are not trained on these categories. However, the evaluation by using these categories is interesting to see the generalization performance of the models in different conditions. In the following subsections, the data structures used in this project are explained in detail.



(a) Distributions of frame counts in each sequence by data split type.

(b) Curvature distribution of all videos. This metric is reviewed in Section 5.2

Figure 3.1: Distributions of the data splits that are valid for sequential model.

Table 3.1: Dataset size and sequence counts for each metadata. None categories are not included by the metadata evaluation since they are the minority in category distributions.

Metadata / Data Split		**Train Dataset**	**Validation Dataset**	**Test Dataset**
Time Of Day	**Day**	555	129	207
	Night	511	134	100
	None	56	19	30
Road Type	**City**	364	108	124
	Highway	99	34	92
	Rural	659	140	121
Road Cover	**Dry**	690	184	163
	Water	385	92	154
	None	47	6	20
Total Videos		1122	282	337
Total Single-frame Observations		188485	11898	13210
Total Sequential Observations		36938	11890	13210

3.1 Single-Frame Data Structure

Single-Frame data structure consists of one image frame as input and 5 location points in this frame as ground truth. This means that the model has only single image input as traditional computer vision and image processing tasks. The data for one training batch is randomly chosen from the data set and they do not have a specific relation among each other. In this way, the model learns the mapping between the road characteristics and the ground truth. There are some frames that are used in training visualized in Figure 3.2 where the ground truth path is projected into the frames.

3.2 Sequential Data Structure

Sequential data is similar to a video stream. A video stream consists of images that are called frames. In this data structure, the target frame is always the last frame in the sequence and the training has been done by the ground truth of the last frame, and the ground truth of first five frames are not used. Figure 3.3 shows one complete sequential observation that is used in the training. The data used in sequential experiments has 6 frames per observation and the frame-rate is 30 which means that the dataset has 1 frame in every 30 frames. Additionally,

(a) Video 1, frame 517

(b) Video 2, frame 217

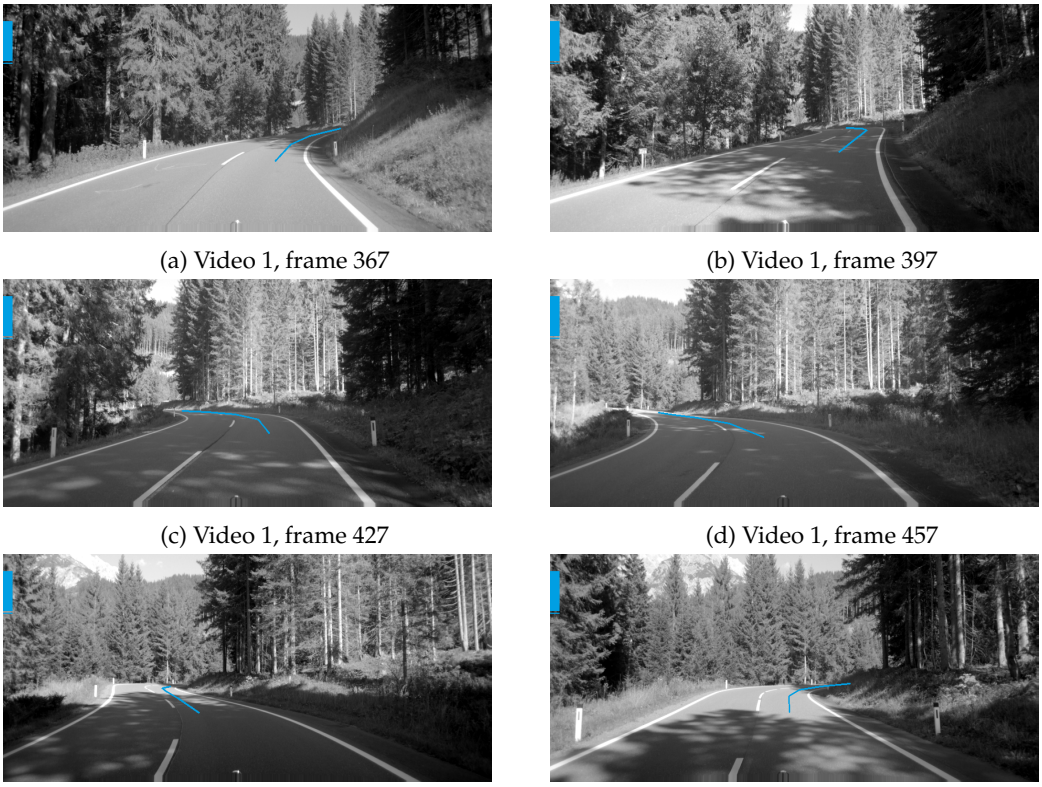
(c) Video 3, frame 375

(d) Video 4, frame 338

Figure 3.2: 4 sample frames from the single frame dataset.

the videos are recorded with 22 frames per second and the data that is used for sequential model takes 1 frame per 1.35 second. The observation in Figure 3.3 has the frame in Figure 3.3f as the target frame (the last frame) and additionally it provides access to five earlier frames as inputs to the model. According to this example, the training for this sequence is done by processing the Figures 3.3a-3.3f, but the loss is calculated by using only the ground truth of Figure 3.3f. The reason for this is that the goal is to predict the path for the target frame by using the historical data instead of predicting paths for historical frames.

In Table 3.1, the total frame count of single-frame model is much greater than sequential model in order to compare the models more fairly. Satisfying this fairness is very hard, since the sequential model will be trained by using only the ground truth path of the last frame, but previous 5 frames are also processed. On the other hand, the single-frame model deals with only one frame. If the single-frame model is trained with only the target frames (6th frame in the sequence) of sequential model, then single-frame will have a disadvantage, since it processes fewer images than sequential model. The final decided approach is training the single-frame model with all the frames that are processed by sequential model, but in this case sequential model has the disadvantage because it only processes some frames as historical frames in the sequence and does not train with the ground truth paths of these frames. However, this is a more fair way to train both of the models. Lastly, there is another visible fact in Table 3.1 that the sequential model does not have exactly 1/6 of the frame count of single-frame. The reason for this is that there may be some overlapping historical frames in the sequential dataset. However, all target frames are unique.



(a) Video 1, frame 367

(b) Video 1, frame 397

(c) Video 1, frame 427

(d) Video 1, frame 457

(e) Video 1, frame 487

(f) Video 1, frame 517

Figure 3.3: 1 sequential observation in the sequential dataset that contains 6 frames.

4 Method

This chapter explains the methods and strategies applied to solve the tasks in the project. The project includes 3 different main models that are investigated. The baseline model is (1) single-frame structured model, the main model is (2) sequential structured model, and (3) uncertainty estimation model that uses both of the model structures (1, 2). These models and their structures are reviewed in this chapter in detail.

4.1 Single-Frame Model

The single-frame model consists of one Visual Feature Extractor (Sub-section 4.3) structure, four Downsampling blocks (Sub-section 4.3), and one output layer that predicts the outputs. This model takes one frame as an input and it predicts the (x, y) coordinates of the next 5 location points for that input frame. Two different output layers are investigated in this project. One of them is Convolutional layer and the other one is a fully-connected layer. Single-Frame model is considered as a baseline in this project since the goal of the project includes investigating the advantages of using historical frames for the prediction of the target frame over using frames alone without learning this historical information. This architecture is described in Figure 4.1 with a flow-chart.

The data that is used for training includes all of the frames that sequential model processes. In this way, the resulting target sequential model can be compared with the single-frame model fairly.

4.2 Sequential Model

A sequential model is a model that has the ability to use historical data to predict the target variable. The sequential model that is used in this project is a LSTM that is a gated structure version of a RNN. This model is based on the same visual feature extractor and downsampling blocks as a single-frame in order to keep the extracted features in a similar structure. The difference is having the sequence learning with LSTM after obtaining the visual features. After running LSTM for each input, there is a fully-connected layer in order to generate the final output for the model. This architecture is shown in Figure 4.2 with a flow-chart.

The defined model takes 6 frames as input and the visual features are extracted by the visual extractor block as shown in Figure 4.2. The parameters in this block are shared for

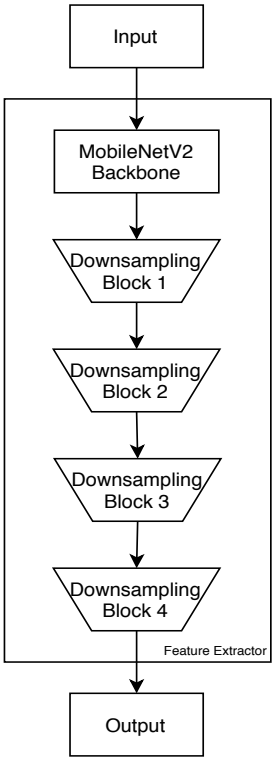


Figure 4.1: Single-Frame model structure.

each input frame, thus each input is processed by the same extractor. After this process, the LSTM layer runs over the desired layer. Since the length of input sequence is 6 frames, the LSTM has 6 cells for this project. At the end of the neural network, the paths are predicted with a fully-connected layer. Mean Absolute Error (MAE) is used as the loss function for the training of the model.

LSTM over different layers can give different performances due to the amount and the quality of the information learned between the features. In this project, three of the possible layers are tested: (1) LSTM over the extracted visual features is shown in Figure 4.3a, (2) LSTM over the smaller visual features in Figure 4.3b and as a last experiment (3) LSTM over both of the extracted visual features in Figure 4.3c. These different head structures are interesting to investigate since it has effect on the dependencies that can be captured from different features sizes. (1)st structure investigates the effects of the dependencies between full size visual features, (2)nd structure investigates what happens if the visual structures are downsampled to the output size, and finally (3)rd structure investigates the effect of using 2 layers of LSTM.

One disadvantage of the sequential model compared to single-frame model is that it processes all of the frames but it is trained to predict only the target frame (the last frame in the sequence) while single-frame is trained to predict all frames in the sequence, therefore the single-frame model is trained by more ground-truth path data.

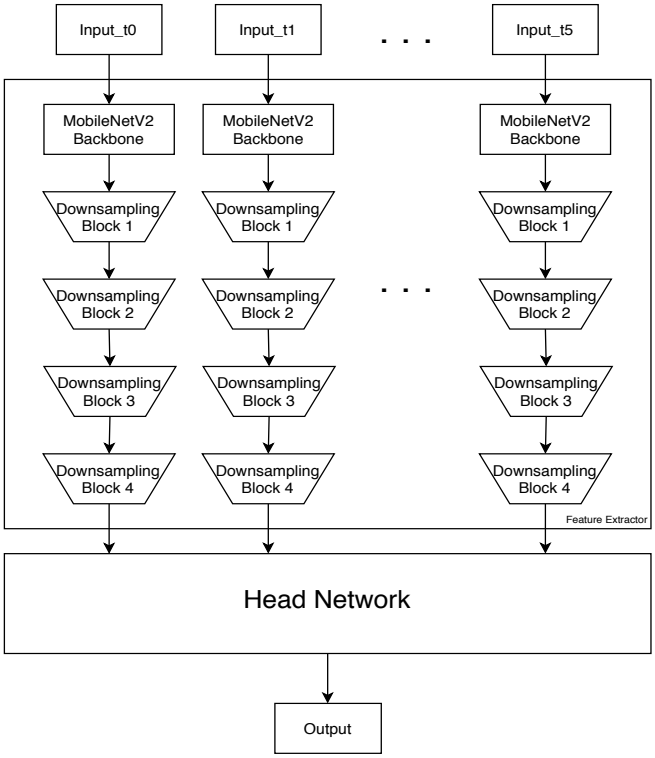


Figure 4.2: Sequential model structure that is used in this project.

4.3 Visual Feature Extraction

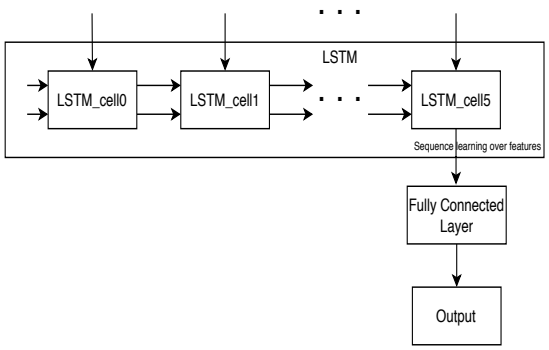
The Visual Feature Extraction component consists of two different sub-components. The first of them is called backbone that is based on a commonly used deep learning architecture for extracting features. The second sub-component is the downsampling block that applies downsizing operations by running CNN and other types of layers in order to reduce the spatial size of the images like width and height.

Backbone: MobileNetV2

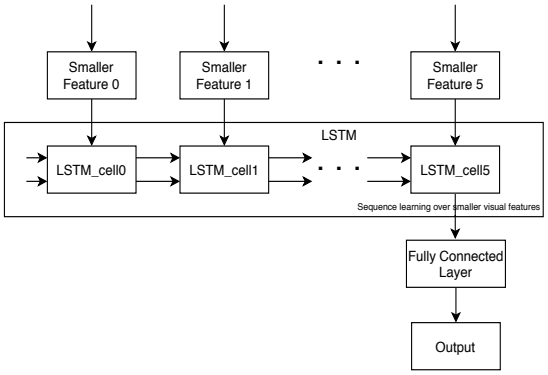
In this project, MobileNetV2 [20] architecture is used as the backbone. Usually the backbone component of the deep learning models requires most of the computation and memory because of their complexity when they extract the features. MobileNetV2 decreases these requirements significantly and still can produce the same accuracy as its predecessors according to Sandler et al. [20]. The MobileNetV2 architecture that is used in this project includes the first 6 bottlenecks in the original implementation and it has 707.168 trainable parameters. After processing the frames with this backbone, the resulting features have the dimensionality of $(14x96x160)$.

Downsampling Block

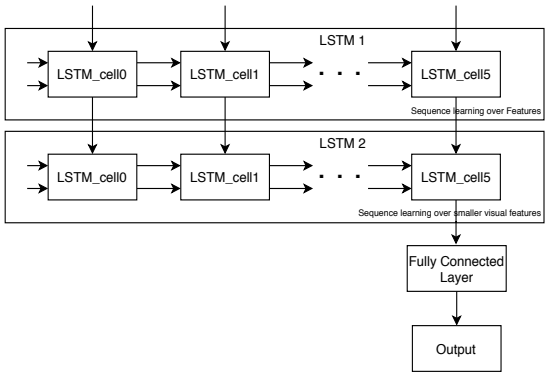
The downsampling blocks take the output of the backbone as input and they are used in order to obtain smaller sizes of feature maps as mentioned previously. In order to achieve this, Pooling or CNN layers with strides can be used. One downsampling block used in this project consists of sequentially one CNN layer, spatial dropout with dropout rate as a parameter, optionally added batch normalization, one Relu activation function, and finally



(a) LSTM head network with extracted features.



(b) LSTM head network with smaller extracted features.



(c) LSTM head network over both features that are normal sized and smaller.

Figure 4.3: LSTM head networks with different size of features.

one MaxPooling with pool size as a parameter. The optional layer may not exist in some of the blocks. By following this structure the inputs can be mapped into smaller dimensionality, so that a feature vector can be obtained in order to use it as input of the LSTM. The general structure of the downsampling block is shown in Figure 4.4.

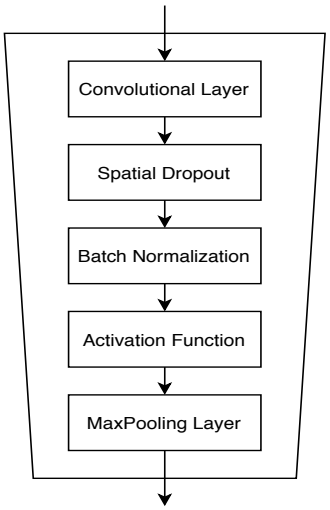


Figure 4.4: Downsampling Block.

4.4 Uncertainty Estimation

Estimation of the uncertainty of the predictions in deep learning models is reviewed in Section 2.5. The suggested uncertainty estimations are epistemic and aleatoric uncertainties. These uncertainties can be combined in order to capture the total uncertainty. However, Kendall et al. [7] states that combining both of the uncertainties does not improve the results significantly in some special cases, since they try to compensate each other when one of them does not exist. Therefore, as a modelling choice, in this project only aleatoric uncertainty is chosen for investigation. This method is selected due to its computational cost, which is significantly lower than Monte-Carlo Dropout method that is used for epistemic uncertainty during evaluation. However, this method requires a separate training unlike Monte-Carlo Dropout method. This method can be applied by assuming that the output is distributed as Gaussian distribution and the main goal of this is to learn an input-dependent variance parameter.

In order to achieve this task, the optimization task of minimizing the Gaussian negative log-likelihood is defined under the assumption that the output is Gaussian distributed. The target distributions can be defined as $\mathcal{N}_{x_j}(\mu_{x_j}, \sigma^2_{x_j})$ and $\mathcal{N}_{y_j}(\mu_{y_j}, \sigma^2_{y_j})$ where $j \in [1, 5]$ and represents the index of location point pairs.

The negative log-likelihood of Gaussian distribution is defined in Eq. 2.18. However, as recommended by Kendall et al. [7], the model should be trained to predict the log variance $s_i = \log \hat{\sigma}_i^2$ in order to prevent the negative or zero variance predictions. The output notation is defined in Eq. 4.1. As is visible, the first values are (x, y) mean predictions as in single-frame and sequential model and additionally the output contains variance predictions. Finally Eq. 2.18 can modified as 4.2.

$$\left(\mu_{x_1}, \mu_{y_1}, \ldots, \mu_{x_5}, \mu_{y_5}, \log(\sigma^2_{x_1}), \log(\sigma^2_{y_1}), \ldots, \log(\sigma^2_{x_5}), \log(\sigma^2_{y_5}), \text{anonymous\_variables}\right) \tag{4.1}$$

$$L_{nn}(\theta) = \frac{1}{D} \sum_i \frac{1}{2} \exp\left(-s_i\right) ||y_i - \hat{y}_i||^2 + \frac{1}{2} s_i \tag{4.2}$$

During prediction time, mean (μ) values are used as path predictions (x, y) coordinates and with standard deviation (σ) the lower and upper intervals for the prediction is calculated

with the formulation of $(\mu - 1.96\sigma, \mu + 1.96\sigma)$, corresponding to a 95% confidence interval of a Gaussian distribution. Moreover, different levels of confidence intervals are investigated in order to evaluate the reliability of the models' uncertainty estimations.

4.5 Evaluation Methods

In this section, the evaluation methods are described in order to interpret the results in Chapter 5.

Mean Absolute Error

Mean Absolute Error (MAE) [19] is a model evaluation metric that is often used for regression tasks. This method measures how close the predictions are to real values (ground truth). By using MAE as a loss function, the model can be trained to solve a regression task. MAE can be calculated by using Eq. 4.3.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i}^{N} |y_i - \hat{y}_i| \tag{4.3}$$

Curvature Score

The Curvature Score measures how a road bends relative to a straight path. This metric is easily calculated by MAE between the path and the straight version of the path. In the point pairs (x, y), x represents the vertical path and y represents the horizontal location. Therefore, in order to calculate the curvature score, first a straight version of the path should be generated by keeping x_t values as same and replace y_t values with zero. After getting a straight path, the curvature can be calculated with the mean absolute error between the path and the straight path. This is formulated as Eq. 4.4. Additionally, Figure 4.5 shows how the straight path and the target (ground-truth) path are located, and the distance between location points.

$$\begin{aligned} \text{straight} &= (x_1, 0, x_2, 0, x_3, 0, x_4, 0, x_5, 0) \\ \text{path} &= (x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5) \\ \text{curvature} &= MAE(\text{straight}, \text{path}) \end{aligned} \tag{4.4}$$

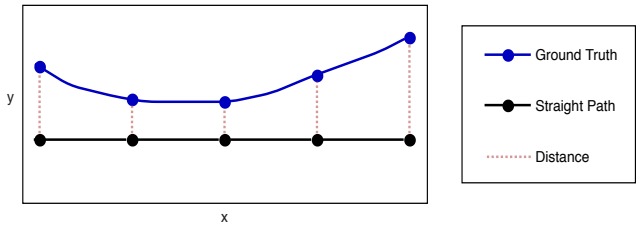


Figure 4.5: Curvature Score calculation. The score is calculated by MAE approach, therefore it is average distance of location points.

True Positive Rates With different Distance Tolerances

The True Positive (TP) rate is calculated by a maximum tolerance distance comparison. In this project, there are two maximum tolerance distance value sets where TP_1 is defined as a tight tolerance and TP_2 is a more relaxed tolerance. Evaluating the model with these tolerances are

Table 4.1: Maximum distances of tolerance levels for each data point.

Method / Max Distance	t_1	t_2	t_3	t_4	t_5
TP$_1$	0.25	0.5	0.75	1	1.25
TP$_2$	0.5	1	1.5	2	2.5

interesting, since they are related to keeping the vehicle inside the lane and having consistent predictions. The predicted points that do not exceed the tolerance distances for each point are labeled as a TP prediction.

Firstly, the Euclidean distance between each corresponding points in the ground truth path and the prediction path is calculated and subsequently these distances are checked if they exceed the maximum distance tolerances respectively that are defined in Table 4.1 where t is time step. Finally, if all points satisfy the condition, the prediction is labelled as TP$_1$ and TP$_2$. This process is visualized clearly in Figure 4.6.

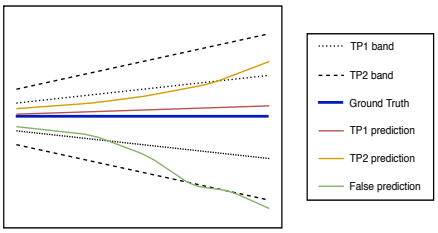


Figure 4.6: TP$_1$ and TP$_2$ visualization.

True Positive Rates With Uncertainty Estimation

This evaluation method follows a similar approach to TP rates with different distance tolerances and this metric is only used for the uncertainty models. However in this method, instead of fixed distance tolerances, there are estimated uncertainties. In order to classify an observation as TP for uncertainty estimation, the calculated confidence interval should cover the ground truth for that observation. The construction of these intervals are made by the confidence interval coverage percentage of Gaussian distribution, under the assumption of the outputs are Gaussian distributed. Firstly, lower and upper boundaries are calculated for the dimensions in each prediction as Eq. 4.5 where $\hat{\mu}$ is estimated mean, $\hat{\sigma}$ is estimated variance and z is Z-score that corresponds to the confidence interval percentage. If the confidence interval covers the ground truth, it is marked as a true positive. This process is shown in Figure 4.7.

$$CI_{\mathrm{U,L}} = \hat{\mu} \pm z\hat{\sigma} \tag{4.5}$$

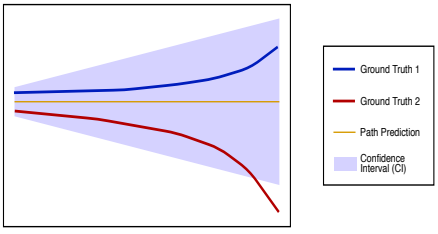


Figure 4.7: TP for uncertainty models.

Metadata

The metadata contains several labels that indicate the characteristics of the video clips used for training, evaluation and test. Therefore, this data can be used in order to compare the performance of the models on different conditions. The metadata features that are used are:

- Time: (day/night)
- Road Type: (city/highway/rural)
- Road Cover: (dry/water)

The metrics for each category is calculated by grouping the whole data according to the metadata label and averaging the metrics for that specific metadata. Only Curvy Road metadata is calculated different, since it is based on curvature score and not a categorical data. In order to average the metrics for curvy roads, the metrics of the greatest 150 curvature scores are used. So the metadata provides information how the models perform in the curviest roads.

5 Results

5.1 Model Selection

In this section, the final model selection process is explained. The selected models in this section are used in test time and all results are obtained by these models. For the hyperparameter tuning, a progressive approach is followed. The steps of the approach can be described as (1) choosing head networks for the model, (2) hyperparameter tuning.

Single-Frame Model

As mentioned in Section 4.1, the single-frame model has two different head networks. One of them uses a Convolutional Neural Network (CNN) layer and the other one uses a Fully Connected (FC) Neural Network layer in the output layer. For single-frame head network selection, different learning rates are tested and based on this, the result of the best models are presented in Table 5.1 and the learning curves for these models can be found in Appendix A.

A CNN uses kernels as parameters, while a FC uses weights between layers that are the coefficients for each neuron. Therefore, a FC has many more parameters to train than a CNN. As this situation increases the complexity of the model, it makes the training time longer and increases the risks of overfitting. Table 5.1 shows that validation errors are not differing significantly, but in the learning curves that are presented in Appendix A, it can be seen that CNN has smoother convergence and FC has a very unstable validation curve. As a result, next experiments are based on CNN head network. One more factor that can be useful is the training time. As CNN has less parameters, one epoch of the CNN takes less time during training. After considering all, CNN head network is selected for next experiments.

Table 5.1: Best validation errors of CNN and FC head network trainings.

	Validation MAE
CNN head	1.06
FC head	1.14

Table 5.2: Downsampling settings.

	Dropout	BN
Block 1	0.2	True
Block 2	0.2	True
Block 3	0	True
Block 4	0	True

Table 5.3: Single-Frame final hyper-parameters.

Parameter	Value
LR	0.001
Batch Size	20
Epoch	10

Another important component in this model is the downsampling component. This components consist of 4 blocks as it is shown in Figure 4.1. Additionally, the downsampling block is described in Figure 4.4 and it is mentioned this block has Spatial Dropout and Batch Normalization (BN) layers as optional extensions. After initial test runs, the settings shown in Table 5.2 are chosen. More specifically Dropout is applied for only the first 2 blocks with a dropout rate of 0.2 and it is observed that batch normalization slightly helps the regularization and it is applied in each block. When dropout is used for all blocks or not used at all, it is observed that the model cannot generalize on validation dataset. In addition to these settings, also different learning rates (LR) are tested and the final model settings are defined as in Table 5.3. Training a model for 10 epochs takes 24 hours and the best validation loss is 0.50. The learning curve of training of the final structure is shown in Appendix A. The reason for differences in the MAE values between head network selection in Table 5.1 and the final model results are caused by using a different subset of the data. However, the different search results can be compared fairly, since they use the same datasets for the specific parameter search.

Sequential Model

The sequential model is described with 3 options that are investigated according to Figure 4.3. All the models are trained by using the same feature extractor and the parameters of the feature extractor are loaded without freezing the parameters from the final trained single-frame model in order to reach faster convergence. These three models have differences in the feature sizes of the LSTM layers; where the first model has LSTM over larger extracted visual features (Figure 4.3a), the second model has LSTM over the smaller features (Figure 4.3b) and the last model has 2 layers of LSTM over both the visual features and the output sized features (Figure 4.3c).

The best validation values are reported in Table 5.4. As can be seen, the best model with the lowest validation MAE is the LSTM layer over features, therefore this model is chosen as the final structure of the sequential approach and will be used for the evaluation. Additionally, the learning curves of these models can be found in Appendix A.

When LSTM is combined with an image processing task, the complexity of the model increases dramatically as there is an additional visual feature extraction process. Therefore, it becomes challenging to handle the memory efficiently. The parameters that affect the memory management in this project are the input sequence length and batch size for LSTM. In this project, 32 GB GPUs are provided by Veoneer and the parameters are chosen by trying different pairs of parameters that use full memory and by evaluating the the loss on training and

Table 5.4: Best validation errors of different positions of LSTM layer. "over both features" means that 2 LSTM layers are used over features and output-sized features.

LSTM layer	Validation MAE
over features	0.43
over output-sized features	0.57
over both features	0.60

Table 5.5: Final hyper-parameters for the sequential model.

Parameter	Value
LR	0.001
Batch Size	7
Epoch	10

Table 5.6: Final hyper-parameters of uncertainty models for single-frame and sequential approaches.

Parameter	Single-Frame	Sequential
LR	0.04	0.004
Batch Size	20	7
Epoch	10	10

validation data. Finally, the input sequence length and batch size are selected to minimize validation error. After running experiments with different learning rates, the optimal initial learning rate is found to be 0.001. The final best sequential model is summarized in Table 5.5. This model is trained for 44 hours.

Uncertainty Models

The uncertainty models follow the same structure as the final single-frame and sequential structures. The only hyper-parameter that is tuned is the learning rate by running experiments with different rates. The chosen hyper-parameter values are listed in Table 5.6.

There is a loss function and a different metric for this model. The loss function is Gaussian negative log-likelihood, and the metric is MAE between the prediction path and the ground-truth path. It should be noticed that the loss function for the uncertainty models contain Mean Squared Error (MSE) term which means MSE-like loss function is minimized instead of MAE. However, MAE is chosen as a comparison metric as the goal was to compare uncertainty models and point estimation models. The results of the best models are shown in Table 5.7. According to the results, it can be said that the sequential log-likelihood model is better, as it has lower negative log-likelihood result. However, the MAE results do not differ much which means that their prediction performances are similar. Additionally, the learning curves of these experiments can be found in Appendix A.

Table 5.7: Best validation errors of different model types on uncertainty models.

Model Type	Validation Log-Likelihood	Validation MAE
Single Frame	0.81	1.08
Sequential	0.25	1.04

5.2 Model Evaluation

In this section, the models are evaluated on training, validation and test datasets. For the final model, only the test data results are compared since it is completely unseen data during the training. In addition, training and validation scores are calculated again in order to evaluate only the performance of the target variables by excluding the anonymous variables that are involved in the training process.

Metrics Evaluation

The metrics that are used in this project are Mean Absolute Error (MAE) and True Positive (TP) rate that are described in detail in Section 4.5. The evaluation results of each final model are shown in Table 5.8. Training, validation, and test evaluations are presented separately.

MAE is the main term of the loss function that is used during training for single-frame and sequential point estimator models. Therefore, MAE can provide insights about the training fitness. However, for uncertainty models of single-frame and sequential, negative log-likelihood is used as loss function and additionally instead of MAE. Gaussian negative log-likelihood contains a term similar to squared errors which is different than absolute error. Therefore, MAE is not the error that the uncertainty models minimize during training directly. However, MAE is used to compare the models since it is a better metric to understand the distance between the prediction and the ground-truth. In addition to MAE comparison, true positive rate with a tight tolerance (TP_1) and more relaxed tolerance (TP_2) give information about how large proportion of the predicted paths are close to the ground truth paths. Therefore, this metrics can be interpreted as the accuracy of the models.

According to Table 5.8, the best model on Test data is the sequential model. While it outperforms all of the other model on TP_1 rate, the differences are less evident for TP_2. This means that at the tolerance level of TP_2 other models reach similar performance. In addition, despite the fact that uncertainty models have lower or closer MAE compared to the other models, they do not have a good performance in TP_1 and TP_2 rates. Having an additional task to estimate uncertainty may cause this issue. However, TP_2 rate of the single-frame uncertainty model is better than for the single-frame model. This means that even though this model is worse in TP_1 which is a tight level of tolerance, it can predict more paths that are covered by TP_2 level of tolerance. The same relation between TP_1 and TP_2 is valid for sequential uncertainty model, however it does not perform better than the point estimator approach of the sequential model.

Another metric that is calculated for uncertainty models is the TP rate for these models. This rate is calculated in a similar approach to other TP values. If the ground truth of the path is covered by the 95% confidence interval, that prediction is a TP value. TP rates for uncertainty models are shown in Table 5.9. These values are calculated with 95% confidence interval for the Gaussian distribution. As can be seen the TP values are high, which means most of the ground-truth uncertainty is captured by the model. However, as clearly visible from results in Table 5.8, uncertainty models do not perform as good as the point estimator models. The reason for the high values of TP uncertainty is that estimated uncertainty intervals are wide and it is more likely that the interval covers the ground truth in most of the observations. In order to see this effect, all of the estimated standard deviations for each location point are averaged and summarized in Table 5.10 where each subscript of x, y denotes the time step in one path. This table shows that the uncertainty increases by the time step in the path, more specifically further points that are predicted are more uncertain. This issue is visualized clearly in Figure 5.3 where the blue area represents the uncertainty with 95% confidence interval of Gaussian distribution. It is noticeable that even in straight paths, the estimated uncertainty increases too much by the future steps that are visualized over x axis.

In addition to TP rate comparison of 95% confidence intervals, TP rates for different intervals are calculated and this is visualized in Figure 5.1. In this calibration plot, x-axis shows

Table 5.8: Evaluation metrics for each of the data splits.

Metric	**MAE**			**TP_1**			**TP_2**		
Model / Data	**Train**	**Valid**	**Test**	**Train**	**Valid**	**Test**	**Train**	**Valid**	**Test**
Single-Frame	1.763	2.200	2.142	0.20	0.15	0.36	0.41	0.35	0.60
Sequential	1.450	1.873	1.877	0.30	0.24	0.48	0.48	0.42	0.67
Single-Frame Uncertainty	1.275	1.627	1.972	0.20	0.17	0.27	0.64	0.51	0.65
Sequential Uncertainty	1.275	1.820	2.109	0.14	0.14	0.13	0.59	0.36	0.53

Table 5.9: True Positive rates for the uncertainty models.

Metric	**TP Uncertainty**		
Model / Data	**Train**	**Valid**	**Test**
Single-Frame Uncertainty	0.94	0.92	0.89
Sequential Uncertainty	0.86	0.77	0.88

Table 5.10: Average of the estimated standard deviations on test data for each location point by uncertainty models.

Model / Points	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4	x_5	y_5
Single Frame	0.45	1.15	1.28	2.48	2.27	4.10	3.27	5.82	4.32	7.68
Sequential	0.38	1.06	0.60	2.63	1.51	4.52	2.36	6.83	3.07	9.36

the real confidence interval percentages, while y-axis shows the proportion of ground truth values that is captured by uncertainty models. The optimal model is expected to follow the diagonal reference line in the plot. This plot provides more information in order to compare single-frame and sequential uncertainty models. As can be seen, both of the models capture same proportion of the uncertainty until 10%. However, in following steps, single frame is over-confident that means it underestimates the uncertainty. Additionally, sequential model is always under-confident and over-estimates the uncertainty. Both of the cases are not optimal, but over-confident state can be interpreted as that the estimated noise parameters cause too wide intervals and captures more than expected uncertainty. In the light of this, sequential uncertainty model can be a better option, because it is better to employ an under-confident model than an over-confident model. However, if the model is too under-confident, then the model is not helpful, since the model will be uncertain in most of the predictions and it will not be reliable. Finally, it can be said that both of the models do not perform good in the path prediction which is the main objective and does not make sense to use any of them instead of a point estimator model. This interpretation of this plot is mainly applicable in the case where all of the predictions are from same one distribution and in the case of this project, all predictions are from a Gaussian distribution that has different parameters. Therefore, this can affect the interpretation of the results.

Metadata Evaluation

In this section, the metadata analysis is reported. There are 3 different categories of the metadata as described in 4.5. The average performances on test data of each category is shown in Table 5.11. The table shows MAE, TP1 and TP2 performance for each category.

In all results in Table 5.11, sequential model has similar or better performance for TP_1 rate. For example, for the Road Cover-Water, Road Type-Highway, and Time of Day-Night

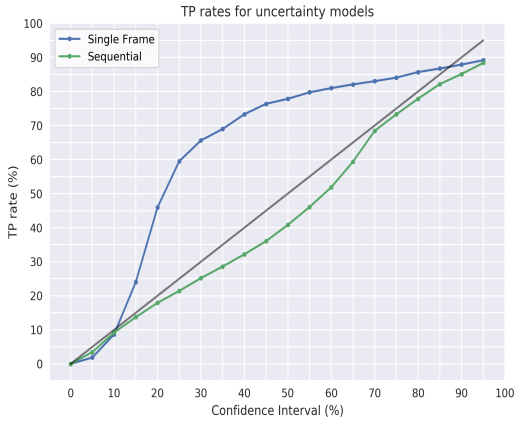


Figure 5.1: TP rates of the captured uncertainty for different confidence interval percentages (x-axis) by uncertainty models. y-axis represents what proportion of ground-truth in test data is captured by different confidence intervals for Gaussian distribution (x-axis).

Table 5.11: Metadata analysis for each model and the metrics.

Model / Metadata		**Time of Day**		**Road Type**			**Road Cover**		**Curvy Road**
		Day	**Night**	**City**	**Highway**	**Rural**	**Dry**	**Water**	
MAE	**Single-Frame**	2.12	2.05	3.11	1.13	1.92	1.97	2.15	3.97
	Sequential	1.84	1.84	2.88	0.81	1.66	1.66	1.94	3.73
	Single-Frame Uncertainty	1.97	1.81	2.87	1.03	1.77	1.74	2.04	3.83
	Sequential Uncertainty	2.11	1.95	3.26	0.96	1.80	1.87	2.20	4.00
TP_1	**Single-Frame**	0.50	0.32	0.36	0.55	0.44	0.56	0.32	0.26
	Sequential	0.52	0.43	0.35	0.68	0.47	0.56	0.42	0.26
	Single-Frame Uncertainty	0.36	0.32	0.27	0.55	0.25	0.38	0.30	0.08
	Sequential Uncertainty	0.22	0.19	0.23	0.25	0.13	0.23	0.17	0.07
TP_2	**Single-Frame**	0.67	0.53	0.52	0.78	0.60	0.73	0.52	0.39
	Sequential	0.70	0.63	0.56	0.87	0.65	0.74	0.62	0.42
	Single-Frame Uncertainty	0.69	0.60	0.57	0.87	0.58	0.73	0.59	0.33
	Sequential Uncertainty	0.67	0.59	0.55	0.86	0.56	0.71	0.58	0.32

sequential model has better results than other models. Single-frame and sequential uncertainty models are significantly inferior to the point-estimator models. Additionally, among uncertainty models, sequential uncertainty model did not perform as well as single frame uncertainty model as the case for point-estimator sequential model outperforms single-frame model. For the curvy road cases, the point estimator models outperform the uncertainty models. However, single-frame and sequential models within same type do not differ that much. In the light of Table 5.8, no model is better than other in this category. Additionally, The difference between two point estimator models is shown in Figure 5.2.

(a) (b)

(c) (d)

Figure 5.2: Single-frame and Sequential predictions with Ground truth. Blue: Ground truth, Green: Single-frame and Red: Sequential

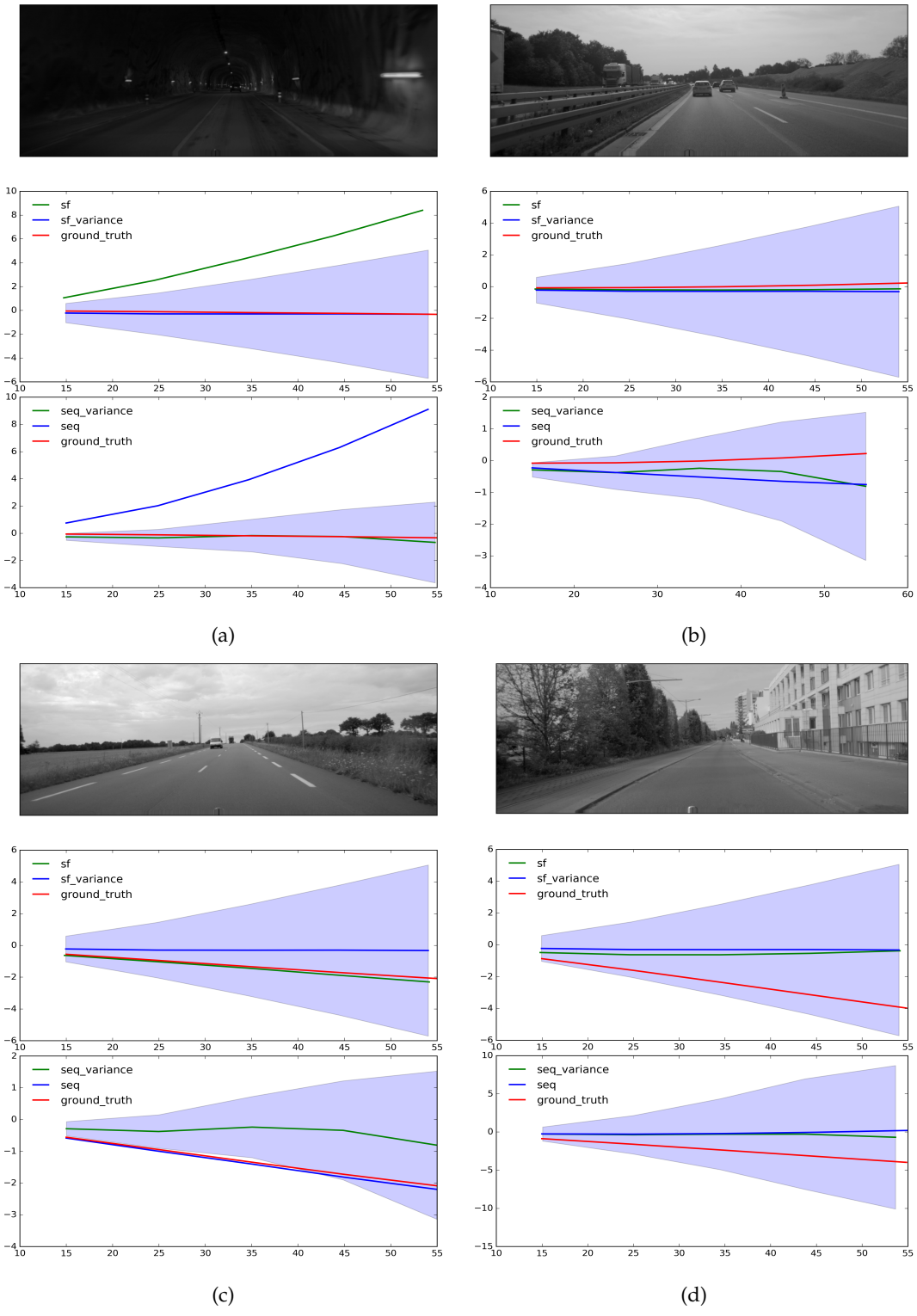


Figure 5.3: Log-likelihood predictions with Ground truth and confidence intervals, also single-frame and sequential model predictions are attached as a reference.

6 Discussion

6.1 Results

The sequential model is performing better than the single-frame model. However, there are some categories in which sequential model performs much better than the single-frame model overall. These categories are (1) Time of Day - Night, (2) Road Type - Highway and (3) Road Cover - Water. For these categories, the performance difference can be observed for both TP_1 and TP_2 and additionally MAE. Therefore, this shows that using the sequential model increases the perception in predictions during night. Because in the night, if the visual features cannot be extracted well then the single-frame model can be misled since it cannot recognize the road structure or other feature. However, even if the sequential model cannot extract features perfectly it predicts the next locations by taking into account also the historical features. Therefore, the results indicate that sequential model has the ability to memorize previous visual features. The same interpretation is also valid for the roads covered with water since the visual features can be misleading due to reflections or unclear lanes. Interpretation of the performance gain in highway is hard, but more accurate predictions on straight paths by assuming the highways are often straight can be one of the reasons. Additionally, it is observed that the sequential model is slightly better at TP_2 performance on the roads that have the highest curvature scores. The superiority of sequential model is noticeable in most of the cases and having heavier computational expenses can be acceptable, despite the fact that this model is not tested in real-time performance. Investigation of appropriateness for real-time performances of the models is not the goal of the project.

The results of uncertainty models are not as good as the point estimator models. They are comparable in path predictions and results are described in Chapter 5. While prediction results of uncertainty models are lower in TP_1 rate, they have a big improvement in TP_2 rate. Especially single-frame uncertainty model is better than single-frame point estimator model for TP_2 rate. Moreover, it is very close to the sequential point estimator model which is the best model among TP_2 rates. In the case of improvements for this model and increasing the performance of TP_1 results, it can be very good option over a sequential model since single-frame models are less computational expensive.

In addition to path prediction, uncertainty models have also uncertainty estimation. Estimated uncertainty is able to capture the big majority of the ground truths with 95% confidence interval for both of the single frame and sequential models. However, in Figure 5.1,

while single-frame model is over-confident, sequential model has better performance despite the fact that it is slightly under-confident. The goal of uncertainty estimation is differentiating the situations that are not safe to rely on the predictions. So that, the application will not be misled by the uncertain predictions. However, the TP_1 are lower than the point estimator models that makes the uncertainty models unreliable in predictions. Since it is observed that uncertainty increases for location predictions that are further away from the current position in the path, predicting shorter distances or less location points with the uncertainty estimation could be investigated in a future work. The path prediction length depends on whether it is suitable and reasonable for the application, and intended ability of the model. In a far path prediction task it is observed that the uncertainty models in this project do not perform well in the sense of path predictions and uncertainty estimation. However, if the desired application requires a shorter or closer path prediction, then this method's performance may be improved.

There can be more metadata categories that includes more traffic scenarios to extend this studies scope for different real world applications. Changing lane, exiting or entering highway, and overtaking a vehicle can be some of examples and the models can be evaluated how they perform in these kind of situations. This data can be added in future works. Another approach that can boost the performance on some of the metadata like day/night or rain/snow/sunny can be predicted by other classifier models and the models can be trained on these different conditions. By this way, in the condition specific applications, the models can be switched by recognizing the condition and there can be better predictions.

6.2 Method

The baseline model that is considered in this project is the single-frame model. This model takes a single image as input, extracts visual features with a backbone and downsampling blocks, and maps these inputs into output space. This method performs well, but the goal of this project is to investigate whether historical information can increase the performance of this method. Therefore, LSTM is investigated for this task. Moreover, in order to estimate aleatoric uncertainty, a negative log-likelihood loss function is defined under the assumption of having Gaussian distributed outputs and trained by using the same structure as the single-frame and sequential model. The results of these models are discussed in Section 6.1 and in this section, the potential reasons for the results caused by models are discussed.

Especially sequential approach requires enough memory to deal with all of these intermediate results during training. Because of this, using memory efficiently is very important to succeed the training with a reasonable sequence length and batch size. In this project, GPUs with 32 GB memory are provided by Veoneer. Additionally, sequence length and batch size parameters are selected by trying various values that can fit into the memory and picked the setting has the best convergence.

LSTM is one of the models that can be used for learning patterns from sequences and combining a recurrent neural network structure with an image processing task increases the parameter count dramatically as mentioned before. The strategy for choosing the hyperparameters to use the memory efficiently is explained in Section 5.1. However, it is observed that increasing the batch size affects the performance in a positive way. In order to set a bigger batch size, the memory should be sufficient and achieving this for LSTM structure is very challenging. In order to achieve this, one approach can be using multi-gpu processing to expand the memory size. Moreover, this can help using more frames in the input sequence and in this way the learned dependencies between frames can be increased. Another approach can be freezing the backbone, or extracting and storing the visual features, and running LSTM and head networks by using these in order to fit larger batches in the memory. However, the feature extraction is not trained in this approach and this may affect the performance bad.

For sequential model, there are many more factors that can affect the performance than single-frame model. For example, in this project, a fixed sequence length is used during training and evaluation which is not the case in reality. In real life situation, the inputs will be streamed continuously to the sequential model while the vehicle is running. In the light of this, judging overall usage of sequential model is not very correct as long as the model evaluated in practice. The current evaluations are done with 6 frames for each sequences. Another factor that can affect the usage of the model in practice is that there is no state reset in reality except the initialization at the very beginning of driving, while this approach is used in training and evaluation. The reason for this is that all of the sequences are not continuous, therefore it is not possible to run these sequences without state reset. Otherwise, the STM and LTM of the sequential model that is described in Section 2.4 will start with unrelated memory for each sequence. These are some of the follow up issues that should be investigated in the case of using this method in practice.

The data that is used for the training of sequential model is sampled by a constraint of having one frame in a frequency that is called the frame-rate. This frame-rate is different than the frame-rate that indicates the frame per second for the video. In this project, frame-rate is specified as 30 which means one sequence observation has 1 frame for each 30 frames as described in Chapter 3. This constraint is defined due to the limited hardware resources and not having the possibility to train a model with observations that have frame-rate 1 like in real world. Since this can be an important factor for the performance of the model, reconstructing the data by using a speed or distance based constraint instead of frame-rate, or evaluating the effect of using different frame-rates with the same approach can be interesting in future work.

The hyper-parameter tuning for the uncertainty models did not succeed as expected. Unfortunately, the uncertainty models do not have a good performance for further away predictions in the path and this makes the model not very reliable. This can be caused by the data inconsistency for the curvature of the road that may cause higher uncertainty in predictions. Because the uncertainty model follows the method of aleatoric uncertainty estimation that captures the uncertainty in the output for each specific input. The expected uncertainty is high when the path is curvy and low when the path is straight. However, the resulting estimated uncertainties increases by the time points of one path. As a result of this, uncertainty intervals that are obtained are too wide to be informative. Therefore, most of the ground truth data is captured by the wide intervals and this increases the TP rates for uncertainty estimation. The high TP rates can be good for the uncertainty method performance, but it makes the uncertainty estimation unreliable for the path prediction. This project investigated only aleatoric uncertainty by expecting this method could capture most of the uncertainty, but for a future work this method can be combined with Monte Carlo Dropout method that allows epistemic uncertainty estimation [7].

6.3 Ethical Considerations

The dataset that is used in the report is collected on public roads using a production camera attached to the vehicle close to the rear view mirror. The data is stored and handled internally by Veoneer according to legal requirements. In the report, example images have been selected or modified to exclude sensitive information, such as license plates or human faces. Additionally, this thesis follows the ethical guidelines provided by the American Statistical Association for statistics practitioners [1].

[1] https://www.amstat.org/ASA/Your-Career/Ethical-Guidelines-for-Statistical-Practice.aspx

7 Conclusion

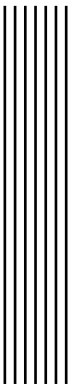
- **How does the historical information that is captured by RNNs affect the performance compared to the single-frame approach?**

  The evaluation metrics are calculated by different tolerance levels as explained in Section 4.5 in addition to MAE. The sequential model significantly outperformed by 12% on test data for the first tolerance level (TP_1) that is the most strict one. In relaxed tolerance level (TP_2), again sequential model has better results by 7%. Even though the difference is not as big as for TP_1, sequential model still outperforms which shows that using historical frames for prediction actually helped the performance.

  In this project, in order to see the performance differences of the models in different road and scene conditions, evaluation by using the metadata for the sequences that are generated previously by Veoneer is investigated. One of the major findings is that sequential model outperforms the single-frame model in some specific cases like night time, water covered road and highway roads which are discussed in Metadata Evaluation subsection of Section 5.2 in detail. As a conclusion, in addition to higher performance of sequential approach, there are specific cases discovered where the performance difference is more evident.

- **Can the model be built to reliably estimate uncertainty in the path predictions?**

  The uncertainty models are built on both of the single-frame and sequential approaches. Despite the fact that both of the uncertainty models are able to capture almost the same amount and majority of the uncertainty at 95% confidence interval, the TP rate of sequential uncertainty model is better than sequential in lower percentage of confidence intervals. However, these high TP rates are not credible since the interval bands are wider than they should be and this is discussed previously. While having the uncertainty high and covering most of the ground truth is beneficial for the TP rates, this is not the case for the path predictions. In real application, the uncertainty should be low for the cases that are easier to predict, and high for the cases that are harder to predict. Finally, it can be said that it is possible to estimate uncertainty in the predictions but the models that are trained in this project are not well-calibrated. However, the uncertainty models in this project cannot predict the vehicle paths accurately and estimate uncertainty reliably in the same time.



Bibliography

[1] Yoshua Bengio et al. "Learning Long-Term Dependencies with Gradient Descent is Difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. ISSN: 19410093. DOI: 10.1109/72.279181.

[2] Jeffrey Donahue et al. "Long-term recurrent convolutional networks for visual recognition and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2625–2634.

[3] Yarin Gal et al. "Bayesian convolutional neural networks with Bernoulli approximate variational inference". In: *arXiv preprint arXiv:1506.02158* (2015).

[4] Yarin Gal et al. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. 2016, pp. 1050–1059.

[5] Ian Goodfellow et al. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[6] Sepp Hochreiter et al. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[7] Alex Kendall et al. "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in neural information processing systems*. 2017, pp. 5574–5584.

[8] Diederik P Kingma et al. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[9] Quoc V Le et al. "Heteroscedastic Gaussian process regression". In: *Proceedings of the 22nd international conference on Machine learning*. 2005, pp. 489–496.

[10] Yann LeCun et al. "Generalization and network design strategies". In: *Connectionism in perspective* 19 (1989), pp. 143–155.

[11] Ilya Loshchilov et al. "Sgdr: Stochastic gradient descent with warm restarts". In: *arXiv preprint arXiv:1608.03983* (2016).

[12] David JC MacKay. "A practical Bayesian framework for backpropagation networks". In: *Neural computation* 4.3 (1992), pp. 448–472.

[13] Vinod Nair et al. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[14] Radford M Neal. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 2012.

[15] Radford M. Neal. "Bayesian Learning for Neural Networks". AAINN02676. PhD thesis. CAN, 1995. ISBN: 0612026760.

[16] David A Nix et al. "Estimating the mean and variance of the target probability distribution". In: *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*. Vol. 1. IEEE. 1994, pp. 55–60.

[17] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[18] David E Rumelhart et al. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[19] "Mean Absolute Error". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut et al. Boston, MA: Springer US, 2010, pp. 652–652. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_525. URL: https://doi.org/10.1007/978-0-387-30164-8_525.

[20] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[21] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[22] Salvatore Trubia et al. "Automated Vehicle: a Review of Road Safety Implications as Driver of Change Intelligent Transportation Systems View project Estimation of Road Safety using traffic microsimulation model with Traditional and Autonomous vehicles View project". In: *27th CARSP Conference* June (2017). URL: https://www.researchgate.net/publication/321110582.

[23] Veoneer. "Vehicle Path Prediction Using Convolutional Neural Networks". Internal company research that is not published publicly. 2017.

[24] Yi-Tong Zhou et al. "Computation of optical flow using a neural network". In: *IEEE International Conference on Neural Networks*. Vol. 1998. 1988, pp. 71–78.

[25] Alex Zyner et al. "A recurrent neural network solution for predicting driver intention at unsignalized intersections". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1759–1764.

[26] Alex Zyner et al. "Naturalistic driver intention and path prediction using recurrent neural networks". In: *IEEE Transactions on Intelligent Transportation Systems* (2019).

A Appendix

A.1 Model Selection

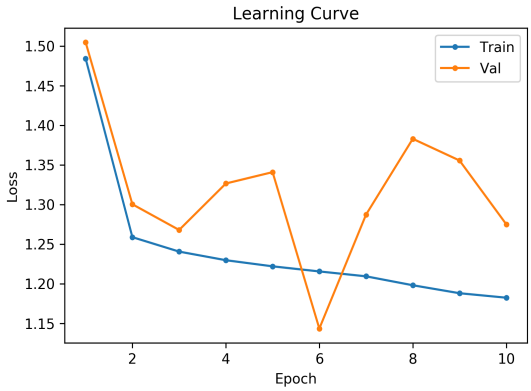


Figure A.1: Learning Curve of Single-Frame Model with Fully-Connected Head Network.

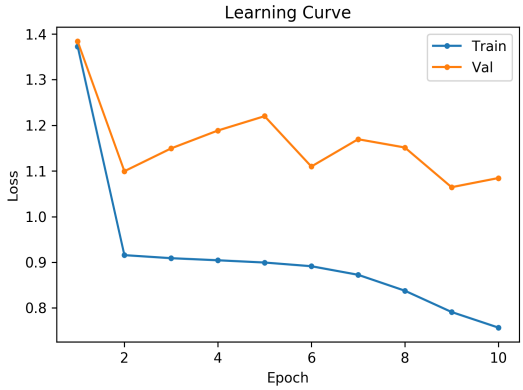


Figure A.2: Learning Curve of Single-Frame Model with Convolutional Head Network.

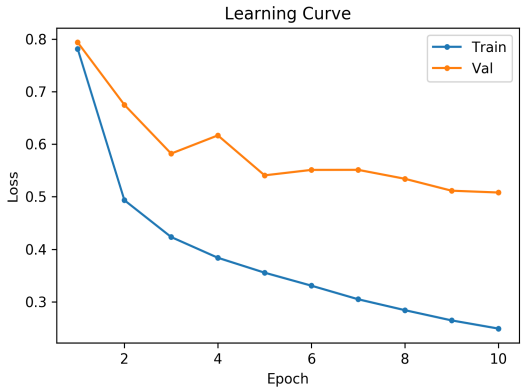


Figure A.3: Learning Curve of Final Single-Frame Model Structure with the main dataset.

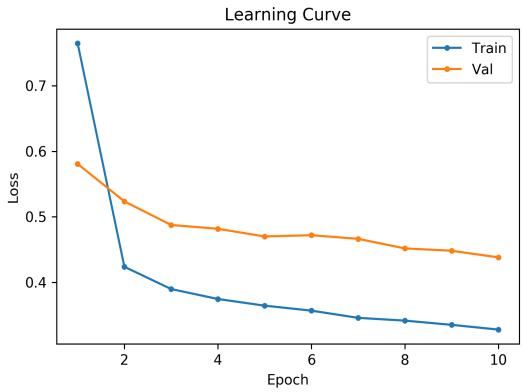


Figure A.4: Learning Curve of Sequential Model Structure with LSTM over visual features.

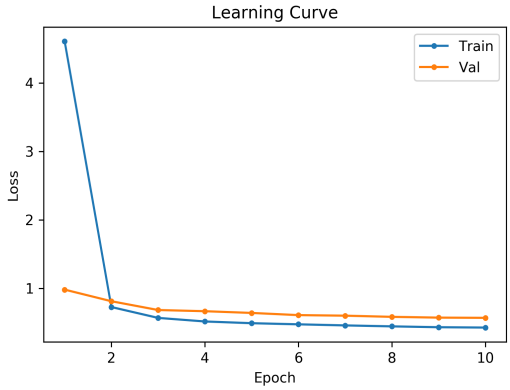


Figure A.5: Learning Curve of Sequential Model Structure with LSTM over path.

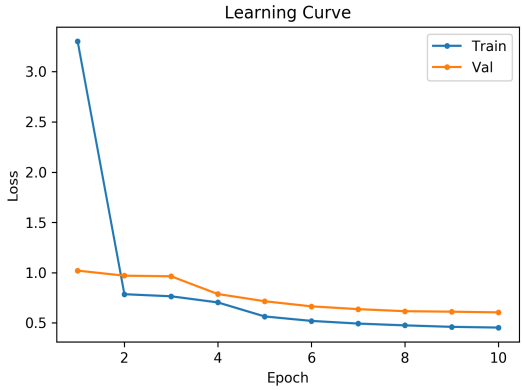


Figure A.6: Learning Curve of Sequential Model Structure with LSTM over both of the outputs and features.

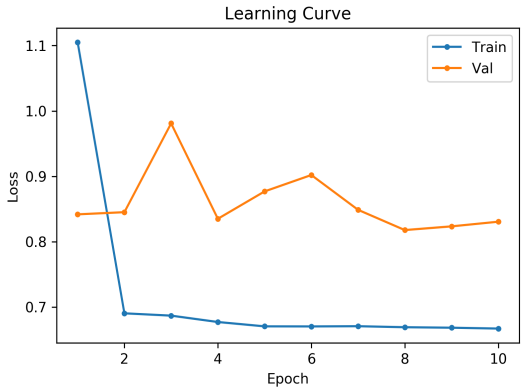


Figure A.7: Learning Curve of Log-Likelihood Single-Frame Model for negative Log-Likelihood score.

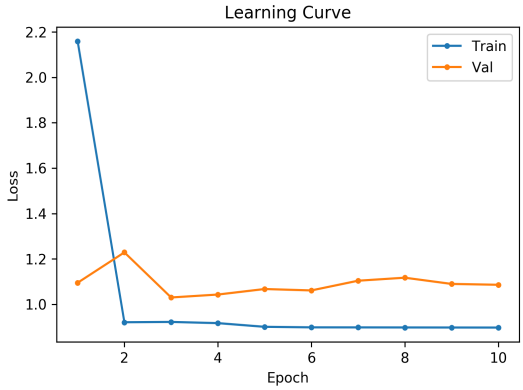


Figure A.8: Learning Curve of Log-Likelihood Single-Frame Model for MAE of the predictions.

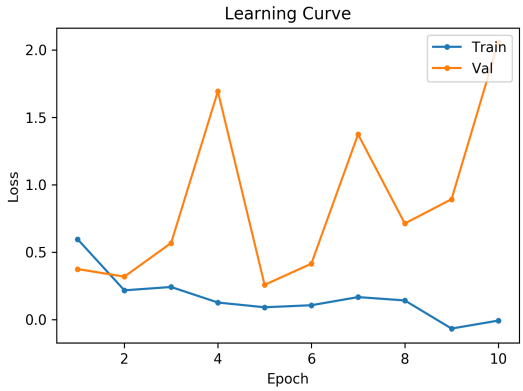


Figure A.9: Learning Curve of Log-Likelihood Sequential Model for negative Log-Likelihood score.

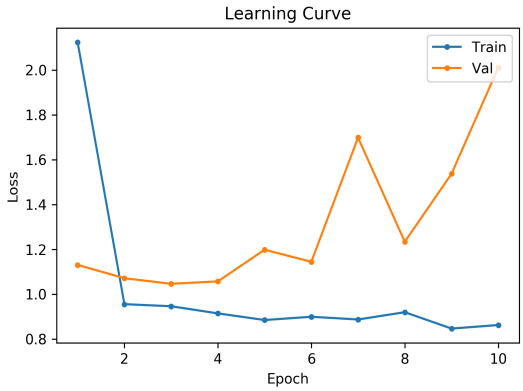


Figure A.10: Learning Curve of Log-Likelihood Sequential Model for MAE of the predictions.