# Building motion prediction models for self-driving vehicles


Imre Fodi

Layout: typeset by the author using LaTeX.
Cover illustration: Lyft

# Building motion prediction models for self-driving vehicles

## A review of the Lyft dataset and competition

Imre Fodi
11407816

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

*Supervisor*
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904 1098 XH Amsterdam

February 12th, 2021

**Abstract**

This thesis will take a look at the problem of the motion prediction of vehicles for autonomous vehicles. In order to replace human drivers in the future, autonomous vehicles need to be able to predict the movement of other traffic participants so they can act accordingly. With the release of a new large motion prediction dataset and competition by Lyft, a new influx of prediction models were presented by the participants. Earlier research on this topic has been limited by the small size of available datasets. Reviewing and validating the submission of the competitors showed a bias to CNN approaches where simple models based on pre-trained CNNs showed the best results. The importance of carefully choosing the training data was also shown to be necessary to avoid regression in the test score.

# Contents

# Chapter 1

# Introduction

The World Health Organisation reported an approximate annual traffic death rate of 1.35 million[1]. Because of concerns for the public health, the United Nations adopted the “2030 Agenda for Sustainable Development" in 2015. This agenda set a goal to reduce the number of lethal accidents by 50% by 2030 [2].

According to a study done on driving errors and accidents in 2005, more than 70% of traffic accidents are attributed to cognitive failures of drivers [3]. Autonomous driving would completely remove human error from the picture. Large companies such as Google [1], Tesla [2], Nissan [3], General Motors [4], and many others are currently developing automated technologies to enhance the safety of road vehicles.

The Society of Automotive Engineers has defined 5 levels of automation [4]. These levels range from simple driver assistance like adaptive cruise control to full automation without human intervention. See Figure 1.1.

Currently, popular commercially available technologies from Tesla [5] and General Motors [6] are considered level 2 automation. The car can control both steering and acceleration, but active human observation/intervention is still required. This level of automation also introduces new safety problems associated with human supervision being neglected. Thus, advancement to higher levels of automation becomes desirable when trying to improve road safety.

Driving automation requires the autonomous vehicle to be able to perceive its

[1] https://waymo.com/

[2] https://www.tesla.com/autopilot

[3] https://asia.nikkei.com/Business/Automobiles/Nissan-to-offer-partial-driving-automation-stand

[4] https://www.gm.com/our-stories/self-driving-cars.html

[5] https://fortune.com/2020/11/08/tesla-full-self-driving-autonomous-vehicle-safety/

[6] https://gmauthority.com/blog/gm/general-motors-technology/general-motors-autonomous-technology/gm-super-cruise/

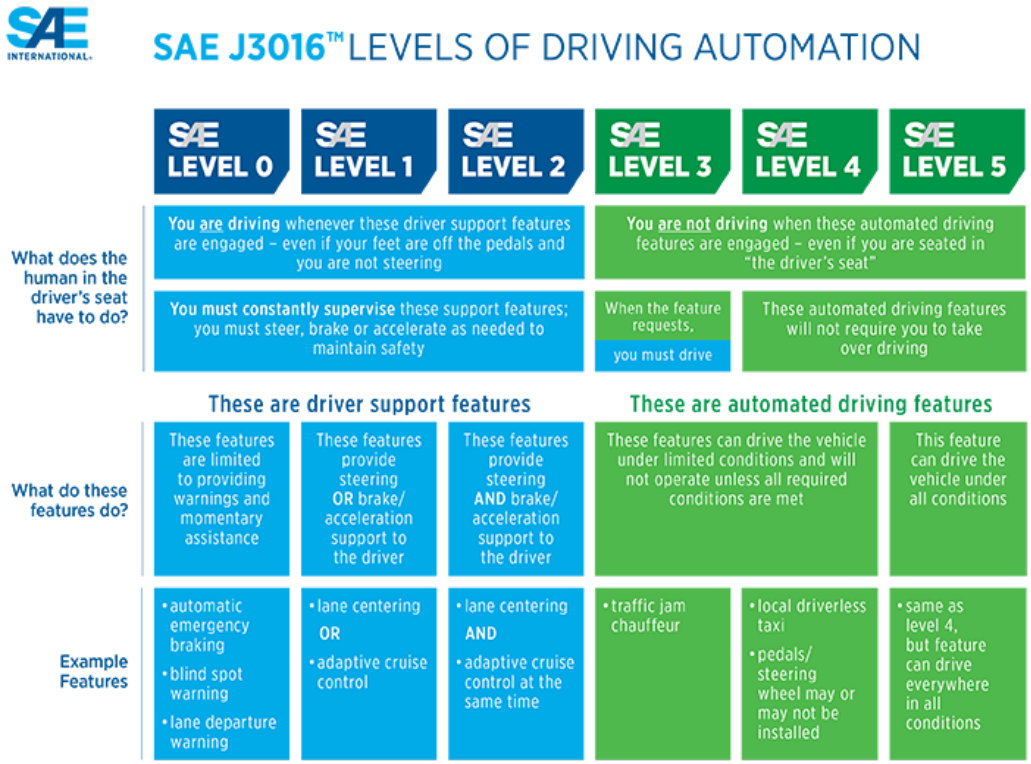



Figure 1.1: The five levels of automation as defined by the SAE
*Image from: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles [4]*

surroundings. Perception of the environment is usually achieved with an array of different sensors. Combining the information of sensors such as LIDAR, 3D cameras, regular cameras, radars, and sonar enables the autonomous vehicle to perceive the surrounding environment [5]. This information is then combined and processed to enable localisation, object detection and mapping[6].

The next step is understanding the perceived environment. Determining the right decisions based on the environment will enable the vehicle to traverse the complex situations found in everyday traffic. Traffic is dynamic so it is critical to be able to estimate the behaviour of the other traffic participants. This can be achieved by predicting the participants' behaviour and the resulting changes to the environment. With these predicted changes a path can be plotted which avoids the paths of the other traffic participants.

Many factors influence the behaviour of the drivers such as vehicle dynamics, road infrastructure, traffic laws, weather but also interactions between multiple traffic participants. The ideal solution would account for all these factors.

Inspired by the Lyft motion prediction challenge (described in Chapter 3) this thesis will try and find such a solution with predictions based on modern machine learning techniques.

## Research questions

This thesis focuses on the last step of predicting the motion of traffic participants. By reviewing the submissions of a recent competition run by Lyft in combination with Kaggle [7] and confirming their implementations, a better insight of the current approach to motion prediction is sought after.

This leads to the following research questions:

RQ1 **What training methods are the most effective in the Lyft motion prediction competition?**

The training method specifies the preparation of data, used loss function and different training parameters.

RQ2 **What specific machine learning algorithms are the most effective?**

Examples of different machine learning algorithms being Convolutional Neural Networks or Linear Regression.

[7] https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles

# Chapter 2

# Background

## 2.1 Prediction models

Many prediction models have been proposed by earlier research. To assess these different models, Lefèvre et al. [7] have organised the current different motion models into three classes with an increasing level abstraction: physics-based motion models, manoeuvre-based motion models and interaction-aware motion models. In this section, we introduce each of these models.

**Physics-based** motion models define the vehicles as dynamic entities that abide by the laws of physics. The predictions are based on the movement capabilities of the car and basic physics such as preservation of momentum. Variables considered include car weight, steering and acceleration input and the grip of the wheels on the road surface. These models are limited to short-term prediction (less than a second), making it impossible to predict a series of movements or sudden changes in the behaviour of drivers.

**Manoeuvre-based** motion models define the vehicles as independent manoeuvring entities. Which means that their movements are not influenced by other road users. These models attempt to identify the manoeuvre intention of a driver early on. When the intention is known, it can then be used to predict the manoeuvre that the driver will make. These predictions are either based on prototype trajectories or based on manoeuvre intention estimation. Both models are based on pre-existing data, either by learning from previously observed trajectories or identifying all possible motion patterns from things like satellite data. The use of pre-existing manoeuvres allows these models to be more reliable long-term compared to physics-based motion models.

**Interaction-aware** motion models define the vehicles as dependent manoeuvring entities. Here the interaction with other traffic users is considered. When basing the model on interacting entities the complexities increase quadratically with the number of entities. However, this becomes unmanageable when considering complex traffic situations. A current solution is to make the interaction model unilateral; it is then assumed that the vehicle of interest is influenced by surrounding vehicles, but not in reverse. These models are again more reliable than the manoeuvre-based models as they account for the interaction of other traffic users. They do however bring a high complexity when considering multilateral interaction between traffic users and have thus far been seen as too computationally intensive.

This thesis will be concentrating on motion models without any prior assumptions on physics or manoeuvres. Instead, predictions are made by finding patterns in previous observations based on semantic images, which allows the application of modern machine learning techniques such as Convolutional Neural Networks. These semantic images encode the locations and motion of all visible traffic agents which allows models to consider the motion of the surrounding agents when making predictions. Making them interaction-aware models.

## 2.2 Machine Learning Models

Modern machine learning techniques require a lot of data to be trained [8], so it is beneficial to start with pre-trained models. Traditionally with supervised learning, a certain model is initially trained on a problem and domain $A$. Then when a new problem and domain $B$ is introduced, new labelled data specific to the new problem is needed.

Transfer learning involves interrelating learning problems by reusing part of the earlier learned knowledge to a new domain. This is influenced by humans and our use of inductive bias, learned from other tasks while learning a yet unfamiliar task [9]. Transfer learning, when applied correctly, decreases the need for domain-specific data, increases the speed of convergence and also allows an increase of performance overall for a given machine learning model [10].

A large collection of pre-trained Convolutional Neural Networks (CNNs) trained on ImageNet[8] is available to use as backbones using transfer learning for many image-based recognition tasks. These models, which have trained on 3.2 million images, have learned to recognise certain patterns and objects which will be beneficial for our image-based motion prediction tasks.

The numbers in different architectures of these CNNs indicate the number of layers used. In this study, for instance, pre-trained CNN backbones with 50 layers

are used. Those layers are pre-trained on ImageNet, whereafter this backbone is extended with additional layers and further trained to predict the required output.

### 2.2.1 ResNet

One of the specific pre-trained models discussed in this thesis is called ResNet [11]. ResNet is a residual neural network. This is an expansion of Convolutional Neural Networks (CNN). Adding more layers to these networks has shown great benefits [12], but a large problem with deeper CNNs is the vanishing gradient problem. With CNNs every other layer is a derivative of the preceding layer. This becomes a problem with certain activation functions, such as the sigmoid function which squishes a large input space into a small input space between 0 and 1. This in turn results in a large change in the input to cause a small change in the output. The resulting derivatives therefore also become increasingly smaller every layer. These small derivatives or gradients become unusable for updating the weights and biases in the backpropagation step. Which degrades the performance of deeper networks.

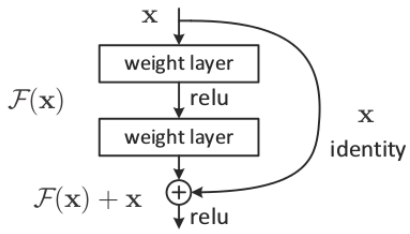



Figure 2.1: Example of a residual block and the residual shortcut connection used in residual networks [11].

Residual networks provide residual connections to earlier layers. As seen in Figure 2.1, the residual connection directly adds the value at the beginning of the block $x$, to the end of the block $(F(x) + x)$. This residual connection avoids the activation functions that "squashes" the derivatives, resulting in a higher overall derivative of the block.

For the larger ResNets (50, 101 or 152 layers) a slight variation of this residual block is used. In order to reduce the time-complexity, a bottleneck design, as seen in Figure 2.2, is used which reduces the parameters used while not reducing the performance significantly.

### 2.2.2 EfficientNet

EfficientNet is another newer pre-trained model discussed in this thesis. The size of Convolutional Neural Networks is commonly chosen arbitrarily. With the most

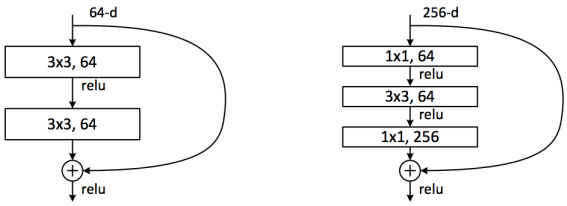
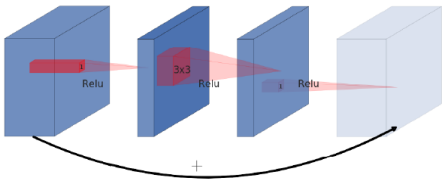


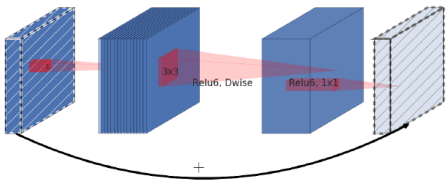

Figure 2.2: Comparison of the normal residual block and the bottleneck variant [11].

common way of scaling a CNN being to increase the depth, width or the image size. Though it is possible to scale two or three dimensions arbitrarily, arbitrary scaling requires tedious manual tuning and still often yields sub-optimal accuracy and efficiency [13].

Tan et al. [13] identified that carefully balancing network depth, width, and resolution can lead to better performance. Based on this observation, they propose a new scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. Based on their findings, they also released a collection of new CNN models named EfficientNets. These models use a similar residual building block to the bottleneck blocks used by ResNet. They use a variation called the inverted residual block. Instead of placing the residual shortcut between large layers, they connect the bottleneck layers as seen in Figure 2.3. This totals in a significant decrease in the number of parameters compared to ResNet while also showing better performance on ImageNet benchmark.




(a) Classical residual bottleneck block.




(b) Inverted residual bottleneck block.

Figure 2.3: The difference between residual block and inverted residual. Note how classical residuals connects the layers with high number of channels, whereas the inverted residuals connect the bottlenecks.

# Chapter 3

# Lyft dataset and competition

The development and availability of large-scale datasets have shown to be important for advancement in AI. In the field of motion prediction, it is particularly important to have a large number of behavioural observations and interactions [14]. The availability of these datasets continues to be limited. Some of the larger datasets freely available are the HighD dataset [15] released in 2018 (147h) and the Argoverse Forecasting (320h) dataset [16]. Lyft contributes with the largest dataset to date containing over 1000 hours of traffic scenes[14]. This chapter will discuss the Lyft dataset, their provided tools and the main challenge of the Lyft competition.

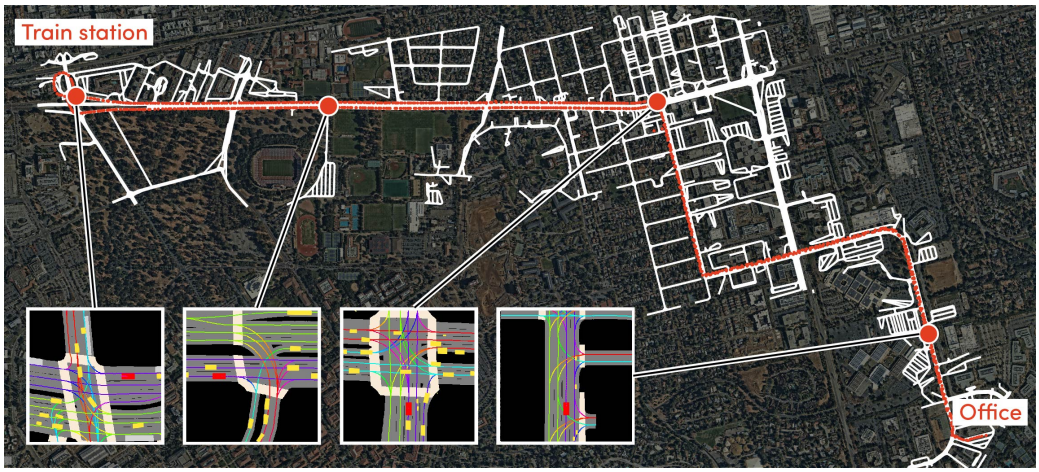



Figure 3.1: An overview of the released dataset for motion modelling on a route spanning 6.8 miles between the Palo Alto train station and the Lyft level 5 office. The examples on the bottom-left show released scenes on top of the high-definition semantic map that capture road geometries and the aerial view of the area.
*Image from:* https://www.arxiv-vanity.com/papers/2006.14480/ [14]

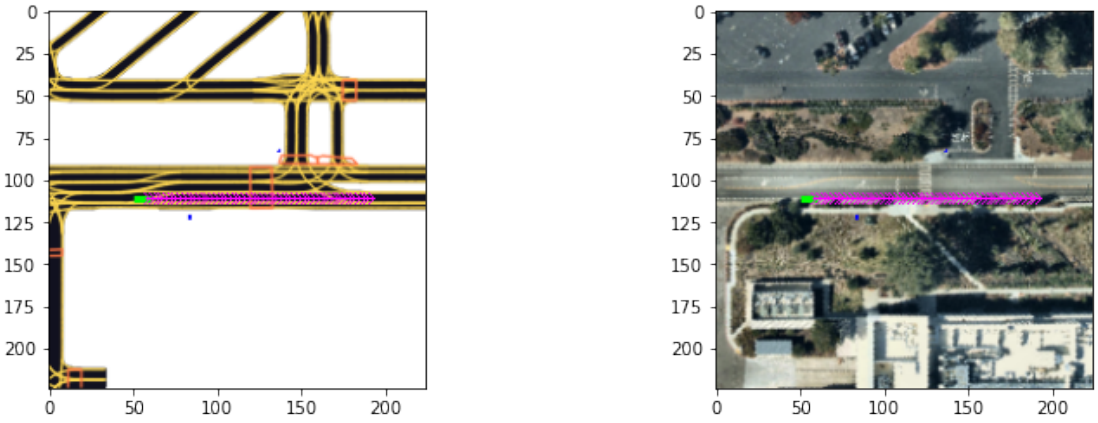
## 3.1 Lyft dataset

The Lyft dataset consists of three main components:

1. 170,000 scenes, each 25 seconds long, capturing the movement of the self-driving vehicle, traffic participants around it and traffic lights state. See Figure 3.1 for an overview.

2. A high-definition semantic map capturing the road rules, lane geometry and other traffic elements. See figure 3.2a.

3. A high-resolution aerial picture of the area that can be used to further aid in prediction. See figure 3.2b.




(a) Semantic map segment

(b) Satellite map segment

Figure 3.2: Example images of the semantic an satellite maps including an agent(green) with its ground truth trajectory (pink) at the same location.

The scenes total over 1118 hours of logs collected by a fleet of self-driving vehicles driving along a route spanning 6.8 miles between the Palo Alto train station and the Lyft level 5 office. For each scene, all visible traffic participants are represented by a 2.5D cuboid, velocity, acceleration, yaw, yaw rate, and a class label. These participants consist of vehicles, pedestrians and cyclists.

The high-definition map contains 15242 human-annotated traffic elements. These are generated by a simultaneous localisation and mapping (SLAM) system [17]. This provides centimetre-grade accuracy which allows the data to be used in the behaviour prediction models.

The aerial map captures the area of Palo Alto surrounding the route at a resolution of 6 cm per pixel. It enables the use of spatial information to aid with behaviour prediction.

The dataset comes with a predefined train, validation and test set. The train and validation set are to be used while training the model. The test set should be used to generate the final predictions. These predictions are then to be submitted to the competition to generate the final score.

## 3.2 Lyft L5kit

The dataset also includes a python toolkit called L5Kit. This provides tools for data loading, filtering, rasterisation and visualisation accompanied by a baseline motion forecasting model. Some tools and another baseline are also included for the task of motion planning but these will not be discussed in this thesis.

The main tool supplied by the toolkit used for training is the BEV scene rasteriser. This package supplies many useful functions to visualise and rasterise a sampled scene (see for instance Figure 4.1). These rasterisations provide meaningful semantic representations of the scene which can then be used as input for Convolutional Neural Network models.

## 3.3 Problem definition

The main goal set by the Lyft competition is to predict the future x and y coordinates of a given traffic participant given their surroundings and historical information thereof supplied by the Lyft dataset. The future positions must be predicted for a 5 second horizon sampled at 10hz. So the predictions submitted to the competition consist of an array of 50 x and y coordinates. The maximum amount of history steps available which are also sampled at 10Hz will be 99. However, not all samples have this amount of history steps available.

Motion prediction brings an inherent uncertainty when considering human behaviour [18]. This can be as simple as seeing someone driving a car in the right lane, from previous experience one could predict them going straight or taking a right turn on the next intersection. These predictions are all valid given the situation and useful to consider when trying to avoid an agent. In order to consider these multiple possible hypotheses, Lyft allows for the submission of 3 paths with their individual confidences per sample. The final score is then calculated using the negative log-likelihood using the ground truth data and these multi-modal predictions. The final score can range from 0 to infinity, where 0 would be a theoretical perfect score where only one path has a 100% confidence and is equal to the ground truth.

# Chapter 4

# Baseline

As mentioned earlier Lyft supplied the competitors of the competition with a baseline motion prediction model. Their baseline uses a pre-trained ResNet50 backbone. This chapter describes this approach considering their input, model and evaluation.

## 4.1 Input

The baseline implementation starts with creating birds-eye view semantic images with the included rasteriser. These images are centred on the agent under consideration (The agent of which we want to predict the future path). They chose a resolution of 224x224 for these images. These images consist of semantic representations of the road and also the chosen amount of history frames. For this baseline, 10 history frames are included per image.

This results in an image consisting of 25 channels or layers. These layers consist of 11 frames representing the surrounding agents, 11 frames representing the agent under consideration and finally 3 layers representing the surrounding environment. The number 11 for both types of agents consist of the 10 history frames and the current frame. See Figure 4.1.

## 4.2 Model

The backbone of this baseline model is a pre-trained ResNet50 model pre-trained on the ImageNet dataset. The 50 in ResNet50 indicates the number of layers this particular model uses.

The basic understanding of the inner workings of Convolutional Neural Networks is that every layer learns to distinguish different features of an image [10,

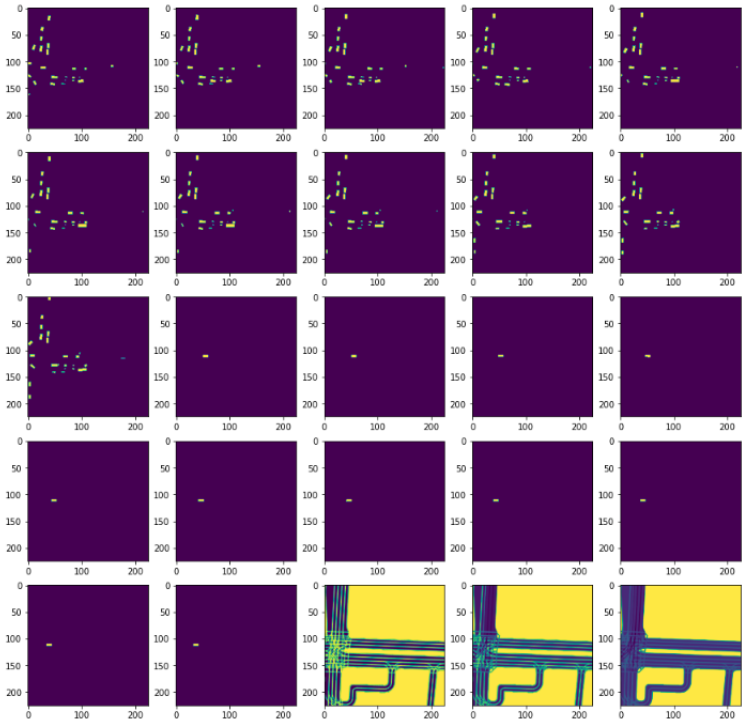


Figure 4.1: A visualisation of all 25 channels created by the rasteriser with 10 history frames and a dimension of 224x224.

19]. Where the beginning layers seem to be detecting edges and orientations of lines, going deeper they start to learn more complicated shapes like squares or circles. The later layers learn more specific shapes specific to the problem the CNN is being trained for. The aim of using this pre-trained model is to make use of the layers that can already detect edges and shapes through transfer learning.

## 4.3 Loss function

The output of the model is a single-mode prediction. This means a single path is predicted which in the case of the competition is an array of 50 x and y coordinates. Therefore they used Mean Squared Error as loss function to compare the single predicted path with the known ground truth. For the optimiser, they use the Adam algorithm [20]. The main improvement over basic stochastic gradient descent that Adam brings is the implementation of learning rate based on a running average of the magnitudes of recent gradients for that weight.

## 4.4 Performance

Running the baseline thrice for 100k iterations with a batch size of 12, resulted in a score of 125.382, 124.822 and 124.936 showing a stable score with a standard deviation of 0.24. Currently, when trying to train past 100k iterations the score was not able to increase. Chapter 5 discusses why and what needs to be improved in order to train on the whole dataset. For reference, the winning score was an 8.579 with place 10 scoring an 11.283 and place 100 scoring a 19.244. An overview of this can be seen in Table 4.1.

| method | score |
|---|---|
| #1 participant | 8.579 |
| #10 participant | 11.283 |
| #100 participant | 19.244 |
| Baseline_300k-mse | 134.875 |
| Baseline_200k-mse | 139.436 |
| Baseline_100k-mse (best) | 124.822 |

Table 4.1: Table with the baseline score trained for 100, 200 and 300k iterations including competitors for reference.

# Chapter 5

# Improvements

The competition attracted a lot of teams and ended with over 900 submitted solutions. This chapter will discuss the main steps taken by the competitors to improve their scores and look into the top-scoring implementation.

With the large number of entries, many different approaches have been tried. However when looking at the top submissions [21–26] they all used a form of ensemble strategies based on pre-trained models on the ImageNet dataset. All the improvements have been primarily tested on 100k iterations because of the limited time available for this thesis. After determining what strategies seemed to work for 100k iterations, more tests on larger numbers of iterations were performed.

## 5.1 Dataset parameters

The L5kit python package provided the participants with tools to adjust the data used for training. Most of the top submissions [21, 23, 25, 26] mention nearly the same change for the parameters `min_frame_history` and `min_frame_future`. These are set per default to 10 and 1 respectively. These parameters filter the data on the minimal frames an agent must have available in the past and future.

After looking through the test set which is used to generate the final submission file send to Lyft, it was discovered that the test set consist of samples that all have a minimal of 10 future frames and no minimum set for the frame history [27]. Whether the defaults were swapped accidentally or not has not been answered by the competition organisers but a significant performance improvement has been reported by most participants when changing the parameters to `min_frame_history=0` and `min_frame_future=10`.

The increase in performance is likely also found because the parameters force the training data to only include samples that have at least 10 future frames avail-

able. Being available involves more than just the frame itself being available in the dataset. It also checks the agents present in these frames. These checks include:

1. Is the frame available.
2. Do the agents in this frame pass the max change allowed in the area. This check ignores any frames where the agents seem to have moved too far compared to the last frame.
3. Do the agents in this frame pass the perception threshold. The perception threshold determines how certain the data is of the existence of a certain agent.

The sample points which do not pass these checks tend to be much noisier and harder to predict. Therefore by decreasing `min_frame_future`, noisy and irrelevant agents are included because these checks are then not made to filter samples with these frames. Another solution might be to create masks that would remove these agents from the frames instead of discarding the whole sample.

Applying the suggested parameters to the baseline model while training for 100k iterations gave an improvement of 15%. Improving the baseline score from 124.822 to 105.517 as seen in Table 5.1.

| method | score |
|---|---|
| Baseline_100k-hist-fut-mse | 105.517 |
| Baseline_100k-mse | 124.822 |

Table 5.1: Table including baseline score with `min_frame_history=0` and `min_frame_future=10`.

## 5.2 Multi-mode prediction

Another improvement over the supplied baseline model, which is also mentioned by Lyft [1], is expanding the model to make multi-modal predictions. Lyft proposes that having multiple behaviour predictions of surrounding vehicles increases the chance of the right behaviour being found. The model is therefore expanded to predict 3 different paths with confidences for each of them.

Applying this to the baseline also instigates the need for a different loss function which can handle these multi-modal predictions. Just like the scoring function from Lyft, negative log-likelihood (NLL) can be used to calculate the error/loss for multi-mode predictions.

To better compare the difference made by the change to multi-mode prediction we first test the negative log-likelihood function with only one path as input. This separates the possible changes made by changing the loss function and changing the prediction mode.

When comparing the single NLL model with the baseline using MSE the score changes from 105.517 to 102.242; a small improvement of 3%. See Table 5.2. Both models iterate on the earlier found improvements of the frame history and future parameters.

The small change can be attributed to the fact that MSE (hist-fut-mse) and the negative log-likelihood (hist-fut-nll) can be shown to be equivalent [28] when considering only one trajectory. Further testing is needed to see if the small change is consistent or just a deviation.

| method | score |
|---|---|
| Baseline_100k hist-fut-nnl | 102.242 |
| Baseline_100k hist-fut-mse | 105.517 |
| Baseline_100k-mse | 124.822 |

Table 5.2: Table comparing the improved baseline score with MSE against NNL including earlier results for reference.

Moving to a multi-modal prediction model shows the largest increase with a change from the previous score of 102.242 to 44.749, all on 100k iterations. The introduction of the multi-modal prediction showed a major improvement of 56%. See table 5.3.

[1] https://github.com/lyft/l5kit/blob/master/competition.md

| method | score |
|---|---|
| Multimode_100k hist-fut-nnl | 44.749 |
| Baseline_100k hist-fut-nnl | 102.242 |

Table 5.3: Table comparing the multi-mode prediction model against the previously improved single-mode baseline.

## 5.3 EfficientNet

The winner of the competition submitted both a single model solution and an ensemble solution. The ensemble solution will be further discussed in Future work 6. Their single model solution is similar to the earlier discussed baseline but used EfficientNetB3 instead of ResNet50 as the backbone. With a claimed score of 9.070 which is the second-highest score, only improved by their ensemble model.

In a attempt to replicate their results, we used Figure 5.1 found in their discussion post [21].

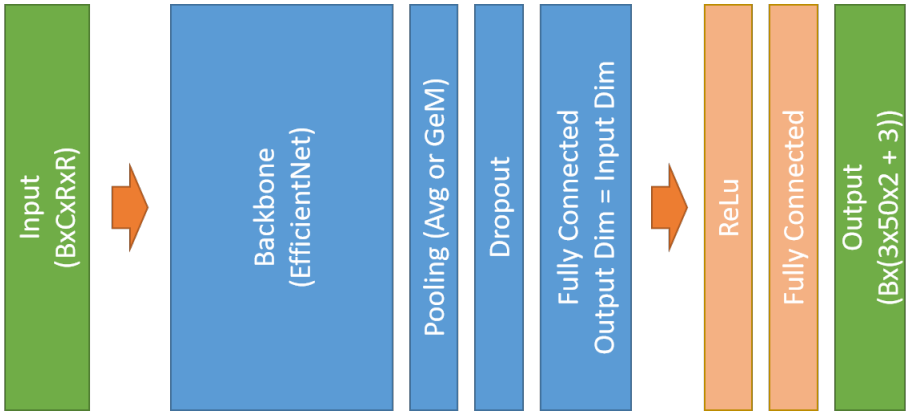



Figure 5.1: Diagram of the best single model with the EfficientNetB3 backbone. *Image from:* https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/201493 [21]

Implementing their EfficientNet model resulted in an improvement over the ResNet based multi-mode model of 13% improving the score from 44.749 to 38.622 at 100k iterations. Seen in Table 5.4.

| method | score |
|---|---|
| Multimode_100k hist-fut-nnl-effb3 | 38.622 |
| Multimode_100k hist-fut-nnl | 44.749 |
| Baseline_100k hist-fut-nnl | 102.242 |

Table 5.4: Table comparing the EfficientNet based prediction model against the previous improvements.

## 5.4 Iterations

The overarching improvement suggested by the top competitors [21–23] is training over the full dataset amounting to 70GB of data. The competition winner achieved this by training for 2 full days using a multi-GPU array consisting of 8 Nvidia Tesla V100's. The third-place winner [22] mentioned a 5-7 day training time using a single RTX2080ti for each of their seven models used their assemble.

In order to confirm that training for longer time does result in an improvement multiple test were done on 200k and 300k iterations. Training over the full dataset was not possible due to time constraints. The best performing multi-mode model at 300k iterations was selected to be further trained until 700k iterations. All these models use the same batch size of 12 just like the baseline.

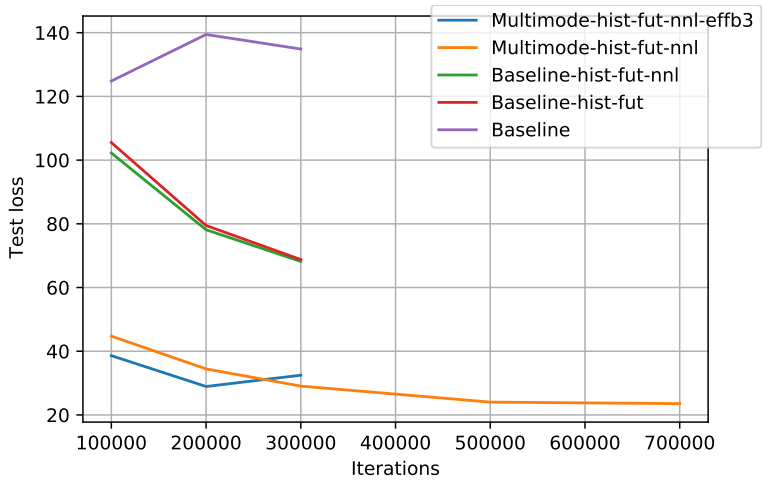



Figure 5.2: Plot showing the test loss for all tested models given the amount of iterations.

When looking at the results in Table 5.5 and Figure 5.2 we see improvements for most of our models with the two exceptions being the baseline and our EfficientNetB3 model.

The baseline was not able to train and increase its score past 100k iterations

| method | score |
|---|---|
| Multimode_700k hist-fut-nnl | 23.151 |
| Multimode_500k hist-fut-nnl | 24.025 |
| Multimode_300k hist-fut-nnl | 29.055 |
| Multimode_200k hist-fut-nnl | 34.454 |
| Multimode_100k hist-fut-nnl | 44.749 |
| Multimode_300k hist-fut-nnl-effb3 | 32.469 |
| Multimode_200k hist-fut-nnl-effb3 | 28.931 |
| Multimode_100k hist-fut-nnl-effb3 | 38.622 |
| Baseline_300k hist-fut-nnl | 68.193 |
| Baseline_200k hist-fut-nnl | 78.137 |
| Baseline_100k hist-fut-nnl | 102.242 |
| Baseline_300k hist-fut-mse | 68.776 |
| Baseline_200k hist-fut-mse | 79.467 |
| Baseline_100k hist-fut-mse | 105.517 |
| Baseline_300k-mse | 134.875 |
| Baseline_200k-mse | 139.436 |
| Baseline_100k-mse | 124.822 |

Table 5.5: Table comparing different amount of iterations for the different all the models mentioned earlier.

without the adjustments to history and future frames present in the training data. It seems that blindly training on the full dataset is not possible, this could imply that the dataset contains quite a lot of noise.

The multi-mode model with the EfficientNet backbone showed promising results until the 200k mark. But went backwards with a small increase in the loss score after that. Whether this is the result of not correctly replicating the model proposed by the competition or some other error is not clear. But we were not able to replicate the performance claimed by the competitor.

The multi-mode model with the ResNet model does show promising improvements until at least 700k iterations. This does show the importance of training on the full dataset with the right dataset parameters being claimed by the top competitors [21–23]. That this model will reach a score close to the winning score does seem unlikely when we look at the flattening curve.

The full dataset contains around 140M samples compared to the 8.4M ($700k \cdot 12$) samples used on our longest test. Which is only 6% of the full dataset. This score of 23.151 manages to take 171st place in the competition.

# Chapter 6

# Future research

Due to limited time, processing power and lack of shared code-solutions there are still some improvements over the baseline that were not tested. This chapter discusses these improvements including the claimed results of the contestants.

## Ensemble learning

The competition winners [21], including others [22, 23], noticed that the models based on pre-trained CNN models were very hard to beat. To improve their score they moved to ensemble learning. Ensemble learning is a machine learning technique that combines several models to produce a result better than any of the models produce on their own [29].

In order to combine the multiple outputs generated by these models a technique called stacked generalisation is used[30]. Here a separate algorithm is trained on the output of the other models to learn how to best combine the input predictions to make a better output prediction. The winning solution trained a simple fully connected layer on a combination of the outputs of four EfficientNet models of differing sizes. See Figure 6.1.

Combining the predictions of these models posed to be challenging. Taking the predicted x and y coordinates of each model as input to the ensemble layer did not result in an improved score. Instead, using the features from the fully connected layer before the coordinate generation did result in an improved score over their best single model. They improved their best single-model score of 9.070 with 5.4% resulting in their winning score of 8.579.

Future research should try and replicate this model to validate these claims.

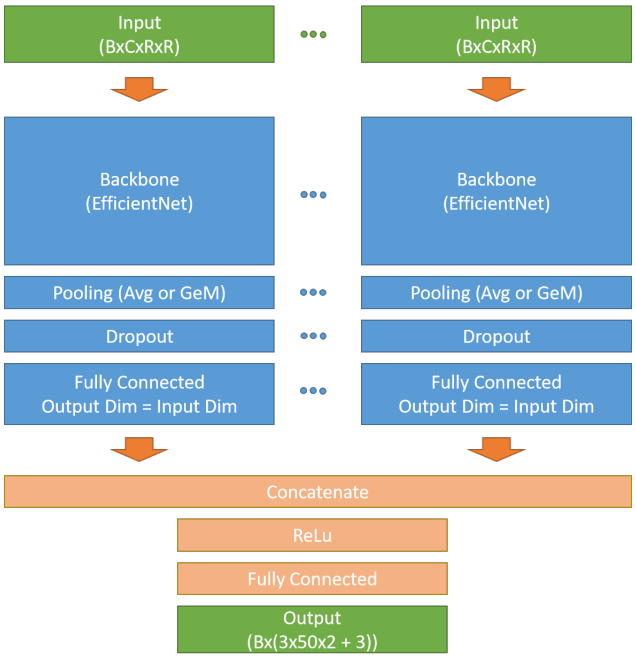



Figure 6.1: Diagram of the winning ensemble model showing the ensemble layer. *Image from:* https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/201493 [21]

## Batch sizes

The use of different batch sizes has also not been considered by this thesis. The participants [21–24] report success with different batch sizes ranging from smaller batch sizes like 12 (used by the baseline) to larger batch sizes of 128. Larger batch sizes generally offer an increase in the computational speed at the cost of being able to generalise less well [31]. Analysing the effect of these different batch sizes on the test loss and train time is also interesting for future research.

## Rasteriser optimisation

The supplied rasteriser has been shown to be the bottleneck while training [21]. This made the training CPU-limited, which means that adding more GPU processing power does not increase performance. Looking at the resources of our computer during training we can confirm that our 6 core CPU was showing usage of 100% while the GPU usage was reporting an average usage of about 45%. The winners managed to speed up the rasteriser by a factor of 4. This enabled them to utilise their multi-GPU array consisting of 8 Nvidia Tesla V100's.

Optimising the rasteriser will allow for faster experimentation and training in future research.

## Iterations

In our research, we were only able to train on a subset of the full dataset. With at most 8.4M samples, we only used about 6% of the full dataset consisting of 140M samples. In order to confirm the claims made that training on the full dataset can result in scores nearing the winning score of 8.579, tests on larger samples sizes need to be performed.

# Chapter 7

# Conclusion

In the introduction we asked the following questions:

RQ1 **What training methods are the most effective in the Lyft motion prediction competition?**

RQ2 **What specific machine learning algorithms are the most effective?**

To answer the first question: We first show the importance of selecting the right parameters when preparing the training data. Choosing the right history and future frames not only showed an increase over the baseline but also seemed to be necessary to consistently train past 100 iterations.

Our largest improvement is found when we change the way we make predictions. Changing to a multi-mode prediction model consistently halves the loss compared to the single-mode baseline. The last step described by most competitors is the importance of training on the full dataset for as many iterations as possible.

As for the second question: The machine learning algorithms used by most of the competitors are some derivative of a Convolutional Neural Network. Our tests with the multi-mode model using the ResNet backbone show promise when trained for 700k iterations placing 171st in the competition while being only trained on 8M samples. Which is only 6% of the full dataset.

Changing to the EfficientNet model, as proposed by the winning competitor, showed promising results when tested with 100k and 200k iterations but showed worse performance at 300k iterations. This conflicts with the results described by the winning competitor and suggest further research.

# Bibliography


[1] W. H. O. R. O. for South-East Asia, "Road safety," 2017.

[2] U. G. Assembly, *Transforming our world : The 2030 agenda for sustainable development*, 2015. [Online]. Available: available%20at:%20https://www.refworld.org/docid/57b6e3e44.html.

[3] T. Allahyari, G. N. Saraji, J. Adi, M. Hosseini, M. Iravani, M. Younesian, and S. J. Kass, "Cognitive failures, driving errors and driving accidents," *International Journal of Occupational Safety and Ergonomics*, vol. 14, no. 2, pp. 149–158, 2008, PMID: 18534151. DOI: 10.1080/10803548.2008.11076759. eprint: https://doi.org/10.1080/10803548.2008.11076759. [Online]. Available: https://doi.org/10.1080/10803548.2008.11076759.

[4] , *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*, Jun. 2018. DOI: https://doi.org/10.4271/J3016_201806. [Online]. Available: https://doi.org/10.4271/J3016_201806.

[5] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies*, vol. 89, Mar. 2018. DOI: 10.1016/j.trc.2018.02.012.

[6] T. Resink, *Vehicle motion prediction for autonomous driving: A deep learning model based on vehicle interaction and road geometry using a semantic map*, 2019. [Online]. Available: available%20at:%20https://repository.tudelft.nl/islandora/object/uuid:bb469461-b879-40fd-abbc-1a6e0233bf1b?collection=education.

[7] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, no. 1, p. 1, Jul. 2014, ISSN: 2197-4225. DOI: 10.1186/s40648-014-0001-z. [Online]. Available: https://doi.org/10.1186/s40648-014-0001-z.

[8] J. Deng, W. Dong, R. Socher, L.-j. Li, K. Li, and L. Fei-fei, "Imagenet: A large-scale hierarchical image database," in *In CVPR*, 2009.

[9] B. J., "A model of inductive bias learning," *Journal of Artificial Intelligence Research*, vol. 12, pp. 149–198, Mar. 2000, ISSN: 1076-9757. DOI: 10.1613/jair.731. [Online]. Available: http://dx.doi.org/10.1613/jair.731.

[10] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191.

[11] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, Jan. 2012. DOI: 10.1145/3065386.

[13] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Sep. 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html.

[14] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, *One thousand and one hours: Self-driving motion prediction dataset*, 2020. arXiv: 2006.14480 [cs.CV].

[15] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, *The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems*, 2018. arXiv: 1810.05642 [cs.CV].

[16] M.-F. Chang, J. Lambert, P. Sangkloy, *et al.*, *Argoverse: 3d tracking and forecasting with rich maps*, 2019. arXiv: 1911.02620 [cs.CV].

[17] Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, 1403–1410 vol.2. DOI: 10.1109/ICCV.2003.1238654.

[18] C. Rupprecht, I. Laina, R. DiPietro, M. Baust, F. Tombari, N. Navab, and G. D. Hager, *Learning in an uncertain world: Representing ambiguity through multiple hypotheses*, 2017. arXiv: 1612.00197 [cs.CV].

[19] R. Ruizendaal. (2017). "How and what do cnns learn?" [Online]. Available: https://towardsdatascience.com/deep-learning-2-f81ebe632d5c (visited on 01/18/2021).

[20] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[21] Ilu. (2020). "1st place solution & l5kit speedup," [Online]. Available: https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/201493 (visited on 01/20/2021).

[22] heartkilla. (2020). "3rd place solution: Baseline + set transformer," [Online]. Available: https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/205376 (visited on 01/20/2021).

[23] corochann. (2020). "4th place solution: Ensemble with gmm," [Online]. Available: https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/199657 (visited on 01/20/2021).

[24] ——, (2020). "6th place: Micro-inputs, lots of data + distance order to ensemble!" [Online]. Available: https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/199588 (visited on 01/20/2021).

[25] B. beluga. (2020). "7th place solution - peter & beluga," [Online]. Available: https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/199649 (visited on 01/20/2021).

[26] nosound. (2020). "9th place solution," [Online]. Available: https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/199636 (visited on 01/20/2021).

[27] F. Pan. (2020). "Large deviation between training loss and evaluation loss," [Online]. Available: https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/185762 (visited on 01/20/2021).

[28] M. Shariatnia. (2020). "Mse is cross entropy at heart: Maximum likelihood estimation explained," [Online]. Available: https://towardsdatascience.com/mse-is-cross-entropy-at-heart-maximum-likelihood-estimation-explained-181a29450a0b (visited on 02/05/2021).

[29] Y. Xu, "Ensemble learning," May 2014.

[30] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992, ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(05)80023-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[31] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, *On large-batch training for deep learning: Generalization gap and sharp minima*, 2017. arXiv: 1609.04836 [cs.LG].