Motion Prediction Using LSTM Network in a vehicular environment

by

Walter Andrei Freire

B.A.Sc Electrical Eng., University of Ottawa, 2016

B.Sc Computing Technology, University of Ottawa, 2016

An MRP

presented to Toronto Metropolitan University

in partial fulfillment of

the requirements for the degree of

Master of Engineering

in the program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2023

Author's Declaration For Electronic Submission Of An MRP

I hereby declare that I am the sole author of this MRP. This is a true copy of the MRP, including any required final revisions. I authorize Toronto Metropolitan University to lend this MRP to other institutions or individuals for the purpose of scholarly research. I further authorize Toronto Metropolitan University to reproduce this MRP by photocopying or by other means, in total or in part, at the request of other institutions or individuals for scholarly research. I understand that my MRP may be made electronically available to the public.

MOTION PREDICTION USING LSTM NETWORK IN A VEHICULAR ENVIRONMENT

Walter Andrei Freire

Master of Engineering

Electrical and Computer Engineering

Toronto Metropolitan University, 2023

Abstract

Autonomous driving has been rapidly increasing in demand and interest over the past several years. It is critical to understand if a pedestrian/cyclist is about to cross the road or if a car is in the middle of parallel parking or making a right turn. There are hundreds of possible actions a road user can take and it is imperative to accurately predict the behaviour of other road users which makes this a challenging problem in autonomous driving. Correctly predicting and evaluating other road users' movements is essential to being able to reduce and avoid crashes thus improving the safety of autonomous vehicles. This MRP solves a part of the Waymo motion prediction challenge using the Waymo motion dataset. The problem is to predict 8 seconds into the future given 1 second of history. The work in this MRP uses (LSTM) Long Short Term Memory networks which is a special type of (RNN) Recurrent Neural Network, which have been widely used for time series prediction such as stock price prediction and electric load forecasting. This problem has been approached by many researchers using different machine learning algorithms such as (CNN) Convolutional Neural Network, LSTM and encoders/decoders. We found that LSTM networks can outperform several state-of-the-art approaches without the use of all available features. Surprisingly, we found that only using the agents' location, speed and angle was sufficient to get state-of-the-art results with a simple LSTM model and data preprocessing. The results obtained from the proposed solution outperformed current LSTM-based methods. The proposed method is implemented in python using the Tensorflow/Keras framework and is available in GitHub *https://github.com/wfrei020/motion_prediction_v1*. The MRP presents results from eight different road scenarios.

Acknowledgments

I am grateful to have the support of Professor Xavier Fernando and Dr. Illanko. Professor Fernando helped me and guided me to complete my master of engineering project, he continuously and quickly provided valuable feedback during the completion of my paper and explained ways to improve my project and final paper to meet thesis expectations. Working with professor Fernando has improved my technical details and writing which I have already begun using in my professional career. I was very lucky and happy to work with professor Fernando, I hope to continue working with him in the future. Dr. Illanko first introduced me to neural networks and gave me the correct guidance to complete my project in LSTM networks. Dr. Illanko would find the time during the pandemic to provide feedback and guidance for my project. I have learned a lot in the past year due to their guidance and experience in the field of machine learning and engineering. I would not have been able to complete my project without their help, I will be forever grateful.

Dedication

This MRP is dedicated to my family, who without their help and belief, I would not have achieved everything I have. Their hard work and the life they provided for me is the reason I have achieved everything I have done.

Table of Contents

List of Tables

List of Figures

Acronyms

(LSTM) Long Short Term Memory

(RNN) Recurrent Neural Network

(CNN) Convolutional Neural Network

(FFNN) Feed Forward Neural Network

(AWS) Amazon Web Services

(GAN) Generative adversarial network

(AV) Autonomous Vehicle

(QOS) Quality of Service

(MEng) Master of Engineering

(HFV) Hypersonic Flight Vehicles

(MSE) Minimum Squared Error

(C-LSTM) Convolutional Long Short Term Memory

(GRU) Gated Recurrent Unit

(LiDAR) Light Detection and Ranging

(2D) 2 Dimensional

(3D) 3 Dimensional

(CEC) Constant Error Carousels

(mAP) mean Average Precision

(RAM) Random Access Memory

(GPU) Graphics Processing Unit

(vCPU) Virtual Central Processing Unit

(GB) Gigabyte

Chapter 1

Introduction

The (AV) Autonomous Vehicle market, has been exponentially increasing, with the current market estimated at $22.22 billion and it is estimated to rise to $75.95 billion by 2027. Autonomous vehicles are embodied with artificial intelligence that poses many benefits and risks. These benefits involve reducing human errors and avoiding crashes caused by humans. To achieve level 5 vehicle autonomy, which involves fully automated driving without human interaction. An important component of driving is the ability to quickly predict another agent's (vehicle/cyclist/person) motion while driving. Humans can accurately predict that when a car approaches a red light it will come to a stop or when an agent is at a red light and green comes on, it is likely that they will start driving or move forward. We can also predict when a driver is about to make a lane switch, trying to pass another vehicle, or just making a right turn by looking at road graph lines. This capacity to predict the motion of other drivers is what allows us to avoid collisions and make safe driving possible. Similarly, when autonomous vehicles go into a complex traffic flow with a mixture of manned and unmanned driving, the uncertainty and randomness of surrounding vehicles driving behaviours bring great challenges to the safety of autonomous vehicles and affect the driving decisions of the automatic vehicles. So autonomous vehicles need to have the ability to predict the movement of surrounding vehicles to avoid collisions and create a safe driving environment.

Designing policies for an autonomous driving system is particularly challenging due to demanding performance requirements in terms of both making safe operational decisions and fast processing them in real-time. Despite recent advancements in technology, such systems are still largely under exploration with many fundamental challenges such as motion prediction still unsolved. Motion prediction is the task of predicting the future trajectory of an agent whether it is a vehicle, cyclist or person.

Using advanced methods and hardware such as Lidar or stereo cameras, it is much easier and quicker to collect data such as bounding boxes, location, speeds, map features, labels and other features. These features can be used to find a solution to the motion prediction problem. Researchers have been using many aspects of machine learning to develop models that can predict different types of tasks such as gesture recognition, speech recognition, stock prediction and human motion prediction [1]. These all involve teaching a model based on time sequence inputs to make predictions. Recurrent Neural Networks RNN, have shown superior results and it is the go-to method for many time sequence tasks as it can learn data that is distributed over short periods.

In this report, the approach that was considered was leveraging (LSTM) Long Short Term Memory, due to its strengths in approximating time series problems. This report will introduce the dataset and the data preprocessing, followed by some background literature on the theory behind the proposed solution. This paper will then explain in detail the solution taken to solve motion prediction to predict an agent's trajectory given 1 second of trajectory data and show the results of the experiments. The focus of this paper is to learn models of agents which can be either a human, a car or a cyclist and predict their motion. More specifically, the use of many features obtained from other perception algorithms to collect features such as speed, location, angles and some traffic light features to predict the agents' motion 8 seconds into the future given 1 second of input data. The architecture implemented in this paper is significantly simpler than other works with some data preprocessing and using some of the map features. With this simple architecture, the training resources that were required were easy to source and training time was much quicker than originally expected for a dataset of over 500 hours of data. If autonomous vehicles can predict the trajectory of surrounding vehicles, the AV can have safer and more accurate control.

Motion prediction task is also a topic of high interest due to the vast domain applications including but not limited to mobility management, data traffic optimization, data placement, to the optimization of the (QOS) Quality of Service of vehicular and wireless networks. For example, a useful application of trajectory prediction is the estimation of the next location where a mobile user will require a network service. Thus, the network operators can reserve resources in advance and react efficiently to the network changes [2]. This report was done as part of the (MEng) Master of Engineering project requirement in electrical and computer engineering.

Chapter 2

Background

Trajectory Prediction is an active research topic and is considered a time series problem. Time series analysis is of two types, univariate analysis in which the inputs to the time series network are of a single input or 1 feature and multivariate analysis which consists of multiple inputs which vary over time [3]. Trajectory prediction has ongoing research but in 2021 Waymo introduced its motion prediction challenge to researchers across the world to solve the trajectory prediction problem. Waymo conducted a baseline [4] using the (LSTM) Long Short Term Memory approach. Alternatively, many modelling techniques can be used to solve trajectory prediction tasks such as using (CNN) Convolutional Neural Network, encode-decoder transformer methods, recurrent neural networks and generative networks. However, there are also classical methods such as using kinematics equations using speed and angle features, which are not very accurate due to the uncertain nature of humans. Other approaches can be used to accurately predict motion displacement without equations such as Riemann sums. Riemann sums have the potential to integrate over velocity over some time to determine the displacement of agents. However, these mathematical models do not take into account external parameters such as human logic or random events that happen during everyday driving that only large amounts of data can determine.

Konev *et al.* [5] had a CNN approach, to convert the raw dataset features into raster graphics and use the generated images to train a model using a (CNN) Convolutional Neural Network architecture that was pretrained using Image-net. For their optimization, they used the negative log-likelihood as their loss function. Other methods involved using target-driven approaches that extract high-level features of the scene using raster images and then predict the trajectory with the highest probability.

Social Generative Networks have also been used to solve trajectory prediction problems but

mainly on pedestrian trajectories such as if two pedestrians want to avoid walking into each other [6]. They propose a novel (GAN) Generative adversarial network based encoder-decoder framework for trajectory prediction capturing the multimodality of the future prediction problem. This implementation also relies on using (RNN) Recurrent Neural Network based networks with convolutional architecture to extract map and agent features.

More recently there have been goal-based methods such as [7] that achieve high performance using goal-based trajectory prediction. Their method relies on map and agent-dense encoding to make their predictions.

Since motion prediction is based on a basis of temporal prediction, a straightforward approach to solving a problem like this is using a recurrent neural network [4] [8] [1] [9]. Since the motion prediction problem is the task of generating the most likely trajectory it can be solved using generative architectures so we may also consider using GAN to generate potential outputs of the future trajectories. Any one of these methods has the potential possibilities to predict trajectories to some extent but each has its problems such as high computing costs, or weaker performance.

Trajectory predictions have also been studied in defence strategies of unmanned vehicles as it is useful to predict an attacker's trajectory to obtain enemies' position, objectives and intents. [10] [11] LSTM networks have been used in a variety of applications one of these applications to predict trajectories of (HFV) Hypersonic Flight Vehicles to improve the defence of HFV from another attacking HFV. [10] showed that using dynamic equations was not feasible due to too many unknown parameters that can influence the equations. It also showed that using regression neural networks was a plausible solution however it required large training samples. The authors propose a trajectory prediction method based on the kinematics equations of hypersonic aircraft, which fully considers the influence and error caused by ignoring the parameters changing with the voyage in the simplification process of the mathematical model. The LSTM network is used to estimate the parameter range of the trajectory motion model to achieve the prediction of trajectory interval. The simulation results show that the LSTM-based trajectory reachable area prediction algorithm requires less prior knowledge, and the error of trajectory prediction is significantly reduced [10].

Trajectory prediction is also being used in ship trajectories using the speed, course, longitude and latitude difference. In [12] they leveraged the (MSE) Minimum Squared Error loss to achieve their results. The authors proposed a ship trajectory prediction method combining Automatic

Identification System information and LSTM recurrent neural network through the study of ship trajectory. The proposed method is used to test the navigation trajectory data of nearly 100 ships in the target water area, and the trained LSTM model is used to predict the follow-up ship trajectory characteristics. By comparing and analyzing the error between the predicted value and the original real navigation data, it is found that the data preprocessing has a greater impact on the prediction result, and the LSTM model can effectively learn and identify the ship's track eigenvalues such as longitude, latitude, sog and cog.

The authors in [13] built a (C-LSTM) Convolutional Long Short Term Memory which consisted of an input layer, convolutional layer and LSTM multivariate trajectory layer. This is a very common use of the CNN-LSTM network. They would first extract features from CNN and use those as inputs to the LSTM network along with other data into the network using a sliding window approach. A very key and useful step in their algorithm is utilizing a density clustering step. They collected similar scenarios of images to get the most features out of the inputs. This clustering step allowed the model to find hidden features which improves the convergence and accuracy of the model. Their experiment results show that the proposed model can provide accurate traffic trajectory results. However It only focuses on one highway, and also shows no comparison to other models. Similarly [2] only predict trajectories of taxis in urban areas. A key step in their method is preprocessing the vehicles' trajectory raw data of the real traces performed by 442 taxis running in an urban area. Then, they trained the LSTM neural network with the dataset of sequences that results from the preprocessing method. Their experimental results demonstrate the effectiveness of the proposed solution, exhibiting a prediction performance higher than 89%, and showing that the prediction process is improved when more observations are considered before performing the prediction.

Other methods use different techniques to solve different types of trajectory prediction problems such as using the encoder-decoder strategy as done by the authors of [14]. They encode the past or input data and the decoder is the future. They utilize two LSTM networks one for the encoder and another for the decoder. This proposed method was called a structural LSTM and was used to learn interaction patterns among surrounding vehicles and thus predict their long-term (5s in the future) trajectories. The proposed network has the following advantages: (1) adaptability to various road geometry due to the special representation of lateral deviation from the center of the target lane;

(2) considering multiple surrounding vehicles as a whole which reflects interaction among these surrounding vehicles; and (3) make use of the interactions not only in understanding historic trajectories (in the encoder) but also in predicting future trajectories (in the decoder). Finally, for real-time trajectory prediction, the authors of [15] relied on (GRU) Gated Recurrent Unit to do fast computational prediction to meet real-time requirements. The authors showed two key points, firstly compared with other training models, the GRU model has a simpler door structure, so the model training is faster and more convenient, but the accuracy is similar to LSTM. Secondly, the loss value of the GRU model at a steady state and the speed of training the model are closely related to the learning rate. If the learning rate is too high or too low, there will be problems of over-fitting or under-fitting, which will affect the performance of the model.

LSTM has seen great success in stock prediction which is another time series task and many have been studied and implemented using LSTM [16] [3] [17]. In [16] the authors proposed a stock trend prediction method based on the attention mechanism LSTM deep learning model using the opening price, highest price, and closing price and the model predicted the next day's closing price. The authors of [3] and [17] proposed a stacked LSTM topology in which the LSTM cells were stacked on top of each other, that is, the output of one LSTM layer would become the inputs into the sequential LSTM layer. This method was able to predict with some accuracy the closing price, however, they mentioned the ambiguities of political and economic effects that the model could not predict. LSTM networks have also been used for other tasks such as creating an action prediction task for general-purpose robots [18]. [18] used a stacked LSTM architecture to learn the spatial and temporal representations from unlabelled training data by generating future frame predictions.

Generative adversarial network training was also used to improve prediction performance. Their experiments show that the proposed architecture achieves state-of-the-art video prediction results while using a compressed, lower-dimensional latent space. Using this latent space enabled real-time inference and control of general-purpose robots. LSTM has been used in smart agriculture solutions in which models are used to achieve precision in agriculture to rapidly increase agricultural production. From the smart agriculture system, data on the environmental conditions of plants are monitored [19]. The authors were also able to show that in time-related tasks LSTM networks would perform better than backpropagation methods. Finally activity prediction in smart

homes environment to detect future events [20]. They have found that LSTM neural networks outperform the other approaches for the prediction of the direct next event, and have also found that applying multi-task learning by jointly predicting the next activity and the timestamp of the next event outperforms separate LSTM models for both tasks separately. Additionally, the multi-task LSTM also outperforms other prediction tasks for the prediction of the timestamp of the next event. LSTM networks have a wide variety of applications in time series analysis and new models and use cases are continuously emerging.

Chapter 3

Proposed Model and Solution

The proposed solution only takes in the agent/object state features and the traffic light features that are near the agents of interest. The map features for this project were not taken into account as that will be left for a future project to research on. This method allows us to simply use the LSTM network on time series data for the position. We will discuss this later but some problems this raises are that the model won't have accurate predictions for turns such as making a right turn as there are no features of the map. Another key feature we are not including is the use of other agents in the area. For example, slowing down because a vehicle is in front of the agent. For now, those are the two important features that are not included because it would be better to use a convolutional network block.

3.1 Project Background

To solve this problem it is necessary to split up the tasks to fully understand what approach can be taken. In this section, we will discuss feed-forward networks, (RNN) Recurrent Neural Network and (LSTM) Long Short Term Memory networks [21] [22].

3.1.1 Feed Forward Neural Network

In (FFNN) Feed Forward Neural Network, sets of neurons are made of layers where each neuron computes a weighted sum of inputs. The input neurons would take signals from the environment and the output presents signals to the environment via hidden neurons that process the input and output signals. Feed-forward networks are loop-free as seen in Figure 3.1 and fully connected which means that each neuron provides an input to each neuron in the following layer and none

of the weights affect a neuron in the previous layers.

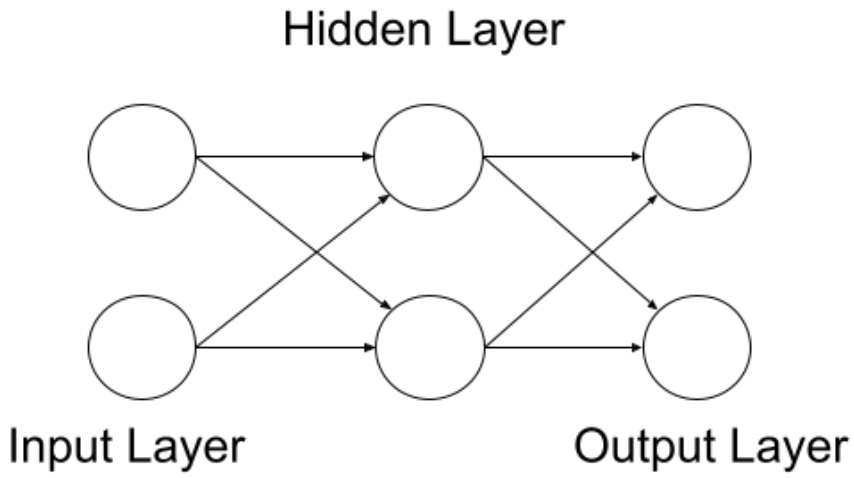


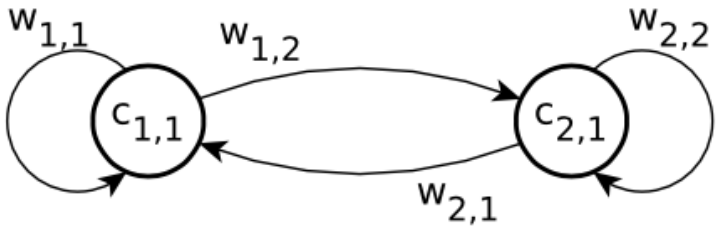
Figure 3.1: Feed Forward Neural Network

3.1.2 Recurrent Neural Network

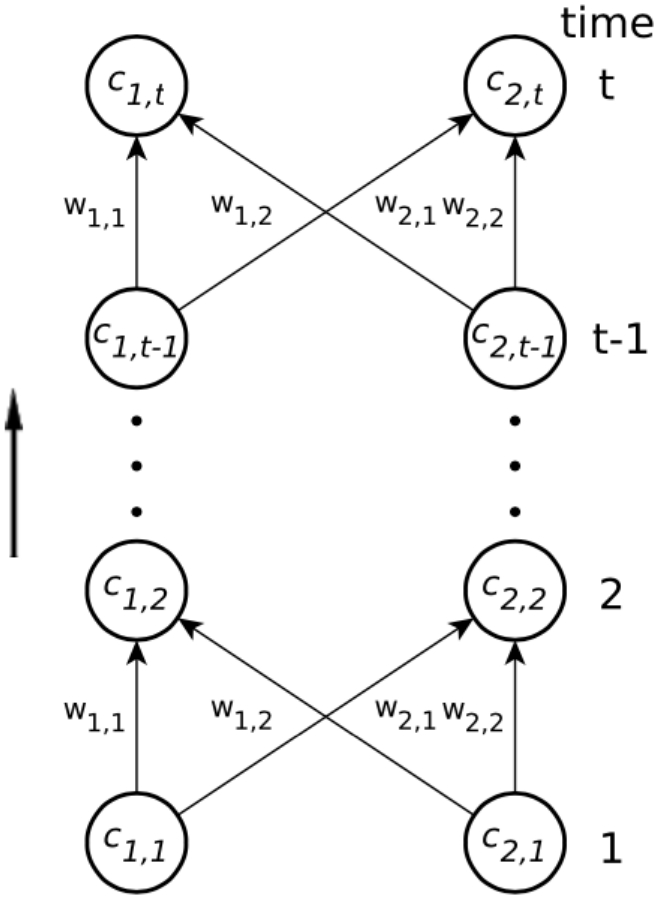
The recurrent neural networks are dynamic systems, they have an internal state at each time step of the classification. This is due to circular connections between higher and lower layer neurons as seen in Figure 3.2a. This feedback connection enables RNNs to propagate data from earlier events to current processing steps which creates this memory of time series events. In Figure 3.2 the w represents the weights applied from the output of one cell to the other and c is the differentiable, non-linear function output by each cell. The network starts at a point in time t' and runs until a final time t.

3.1.3 Vanishing Error in RNN

Standard RNN can not bridge more than 5 - 10 time steps due to the backpropagation error signals. The signals tend to either grow or shrink with every time step. over many time steps, the error typically blows up or vanishes. When the error blows up this leads to oscillating weight and when the error vanishes the learning can take an extremely long time or not converge at all. To fix this problem one solution is proposed (LSTM) Long Short Term Memory.



(a)



(b)

Figure 3.2: 3.2a is a fully connected RNN with 2 neurons. 3.2b is the representation of an RNN during time steps. It behaves similarly to a feed-forward network. [21]

3.1.4 Long Short Term Memory Networks

LSTM can learn how to bridge minimal time lags of more than 1000 discrete time steps. This is done using (CEC) Constant Error Carousels, which enforces a constant error flow within special cells. Access to these cells is handled by multiplicative gate units which learn when to grant access. CECs are the preservation of error where short-term memory storage is achieved for extended periods. Another part of LSTM is the handling between connections from other units which is done using memory blocks. These memory blocks as seen in Figure 3.3 are needed because of the continuous input being inserted into the RNN and can have conflicting weight update signals. The memory blocks are used to compute the weight updates that are contributed by the previous input and other neurons. To avoid this problem of conflicting weight updates, the LSTM extended the CEC with input and output gates connected to the network input layer and other memory cells which result in a more complex LSTM network called memory block.

The essence of LSTM is an improvement based on the RNN structure, which has a more sophisticated information transmission mechanism than the traditional RNN. The LSTM network controls the state of cells through a structure called "gate" and cuts or adds information to it. Specifically, the LSTM contains three "gates"—forget gate, input gate, and output gate. It can use these three "gates" to protect and control the cell's state. Gates can act in the hidden layers of the LSTM model. The input gate takes input from the output of the LSTM neural network cell in the last iteration, which controls the intensity of input into the memory cell. The forget gate determines when to forget the output, thus it can select the optimal time delay for the input sequence. It controls the strength of the memory cell to maintain the value of the previous moment. The output gate receives all the calculated results and generates the output for the LSTM neural network unit, which controls the strength of the output memory cell. Three gates are three gates of information, which control the transmission of neuron information, control how much new information is allocated to the current neuron and how much information from the current neuron to the next neuron [23].

The structure of LSTM cells as seen in Figure 3.3 is defined as follows, x_t is the input value, h_t is the output value, and $\times$ is the multiplication operation. The input gate and output gate need to multiply the input value and output value of the cell, and the forget gate needs to multiply the

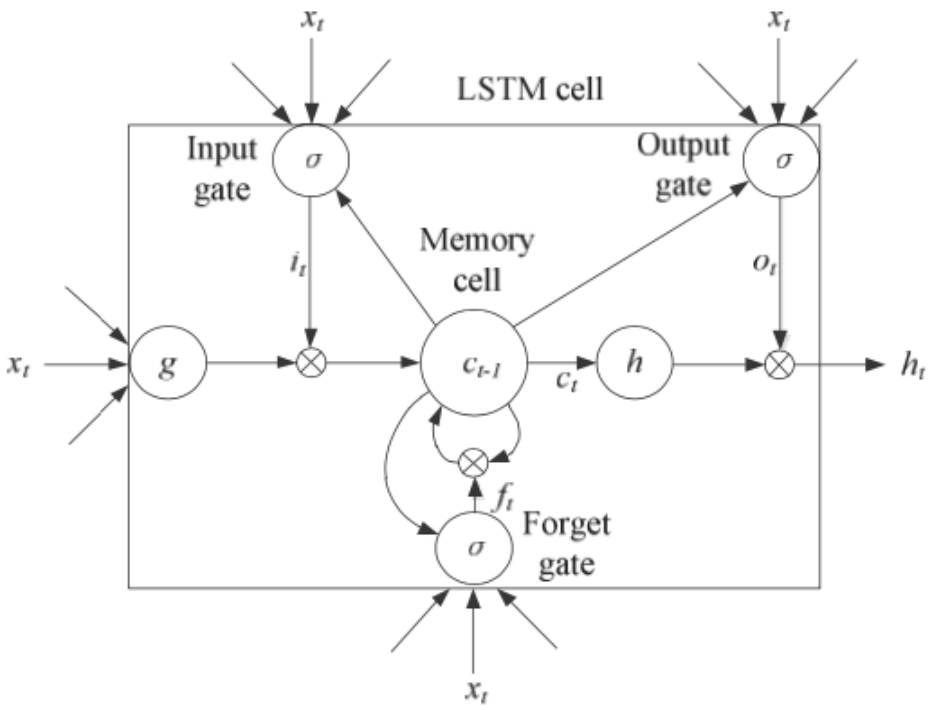


Figure 3.3: Structure of LSTM Cell [23]

previous value of the cell. $\times$ represents the activation function formed by the sigmoid function, which is used to control the weight of the cell flows, and its value range is between 0 and 1. The calculation equation of each node is as follows,

$$h_t = h(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{3.1}$$

$$p_t = W_{hy}yt - 1 + b_y \tag{3.2}$$

W is the weight matrix, the bias term is b, and the calculation equation of the memory cell's hidden state is as follows,

$$i_t = \sigma(W_{ix}x_t + W_{hh}h_{t-1} + W_{ic}c_{t-1} + b_i) \tag{3.3}$$

$$i_t = \sigma(W_{fx}x_t + W_{hh}h_{t-1} + W_{fc}c_{t-1} + b_f) \tag{3.4}$$

$$c_t = f_t * c_{t-1} + i_t * g(W_{cx}x_t + W_{hh}h_{t-1} + W_{cc}c_{t-1} + b_c) \tag{3.5}$$

$$o_t = \sigma(W_{ox}x_t + W_{hh}h_{t-1} + W_{oc}c_{t-1} + b_o) \tag{3.6}$$

$$h_t = o_t * h(c_t) \tag{3.7}$$

In the equation above, $*$ is the scalar product of two vectors or matrices, g and h are the extensions of the standard sigmoid function, which value range is $[-2, 2]$ and $[-1, 1]$ [23].

$$\sigma(x) = \frac{1}{1 + e^x} \tag{3.8}$$

3.1.5 Strengths and limitations of LSTM

LSTM excels on tasks in which a limited amount of data must be remembered for a long time. This property is accomplished by the use of memory blocks. These memory blocks have access control in the form of input and output gates, which prevent irrelevant information from entering or leaving the memory block. Memory blocks also have forgotten gates which allows for continuous prediction because they make cells completely forget their previous state, preventing unbiased prediction. One of the main problems with LSTM is that the number of memory blocks in networks does not change dynamically, so the network's memory is ultimately limited. This means that processing continual input streams without explicitly marked sequence ends. Without resets, the internal state values may grow indefinitely and eventually cause the network to break down. Even though it is limited in memory blocks, it is an advantage over (GRU) Gated Recurrent Unit. GRU is also a variant of RNN which is made up of 2 gates, the forget gate and an update gate. GRU has less complexity than LSTM which is great for some tasks. However, GRU does not have the long-term memory that LSTM offer and so on larger datasets, GRU tend to perform poorly. Since for this experiment we are using one of the largest datasets, GRU networks did not provide adequate results like LSTM did on this dataset.

3.2 Dataset Preprocessing

For the proposed solution the input features that will be used as input into the LSTM network were extracted from the provided features found in Waymo Dataset. These features were past/present state features that include x, y, speed, velocity yaw, velocity x, velocity y and bounding box yaw. To further enhance the model we also added a traffic light feature prepared in the following way. Every set had a max of 16 traffic lights near the objects and the features that were used were the x, y and state of the traffic light. So this results in 55 features for the inputs with 11-time steps for each.

3.3 Network

The LSTM Network works by taking in the feature inputs and going through 160 cell LSTM network followed by a Dense Linear layer as seen in Figure 3.4. The LSTM network used a hyperbolic tangent function (Tanh) activation function and the dense layer used the Rectified Linear Unit (ReLU) activation function. The Tanh was chosen over sigmoid and ReLU because of the symmetry around 0, whereas Sigmoid and ReLU zeros out all values below zero. This means that any derivative below zero will affect hyperbolic functions. This is a problem for deep learning networks because, after some time, these kinds of networks stop learning. The final layers used a ReLU activation function over a sigmoid because we want to map the output of the LSTM layer to that of our required outputs and do not need any classification that benefits from using sigmoid functions.

3.4 Loss

Similar to [12], a mean squared error loss function is used to find the error between the predicted sample and the ground truth target. The error was calculated as followed

$$MSE = \frac{1}{n}\sum_{i-1}^{n}(Y_i - \hat{Y}_i)^2 \tag{3.9}$$

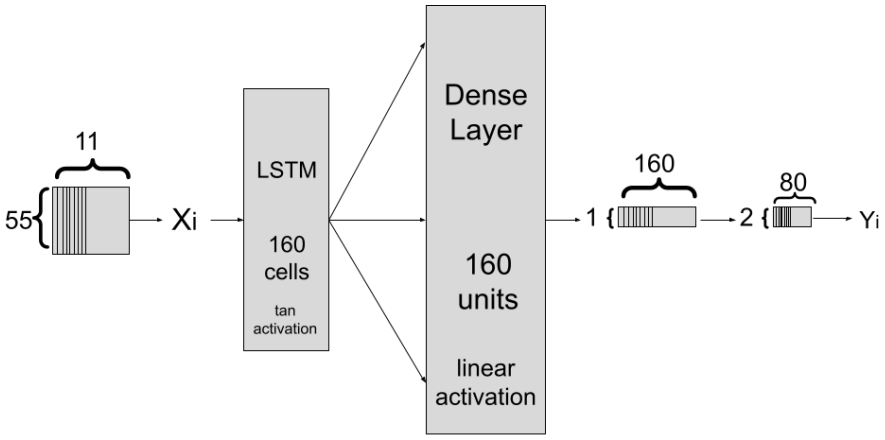


Figure 3.4: Motion Model

where Y is the target trajectory and $\hat{Y}$ is the predicted trajectory. Using the MSE Loss function to optimize the weights for the dense layers and LSTM cells. While experimenting, results showed that the position of the agents had very varying locations. For example, one agent in one scenario had a position at (40000, 10000) while another had a position at (1000, 500). This resulted in the model having really large errors as it was difficult for the model to learn.

3.5 Algorithm

To overcome the varying position for multiple agents, it was found that each agent's displacement was between 0 - 300 meters. So to greatly improve the results and model training, each agent was translated so that they all have a starting position at the origin (0, 0) and then the x, y translated coordinates as the inputs to the models. In Figure 3.5 you can see an example of the training data shift.

The algorithm to train and test the solution is described as followed

1. Training

   (a) Store the Agents starting position $P_o = (X_x^0, X_y^0)$

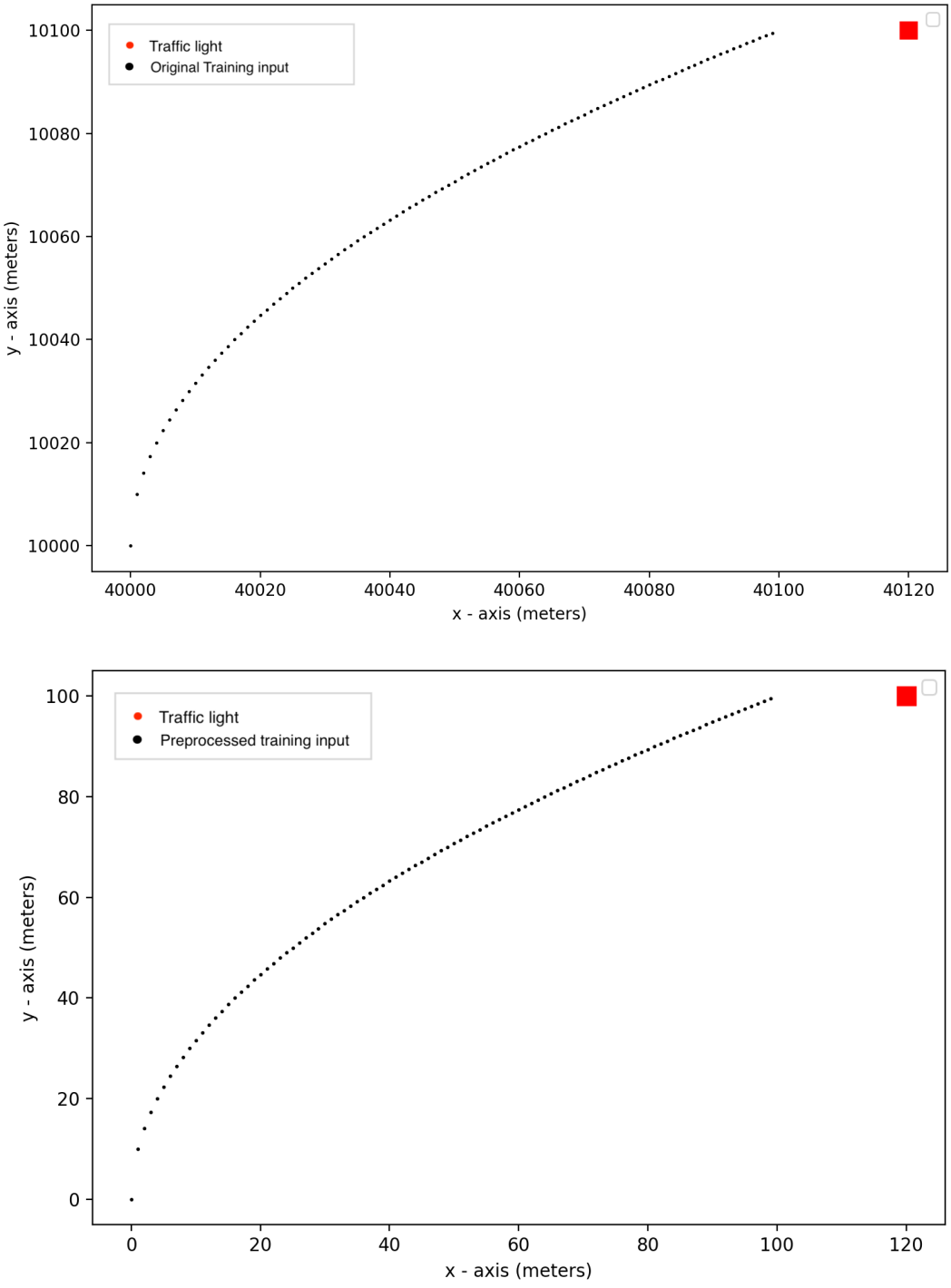


Figure 3.5: Agents' trajectory translated to start at the origin during data preprocessing to be used as inputs to models

(b) Subtracts the agent's starting position from the agent's time steps and shifts the traffic lights concerning the agent's initial position such that $\bar{X} = X^i_{x,y} - P_0 \; \forall \; i$ where $0 \leq i < 10$ and $\bar{Y} = Y^i_{x,y} - P_0 \; \forall \; i$ where $10 \leq i < 91$. X is the input features and Y is the targets.

(c) Use the new Inputs and target to feed into the model, the other features remain the same. $\hat{Y} = Net(\bar{X})$

(d) Find the MSE loss , $loss = MSE(\bar{Y}, \hat{Y})$

(e) Optimize the weights and continue to the next agent

2. Testing

(a) Store the Agents starting position $P_o = (X^0_x, X^0_y)$

(b) Subtracts the agent's starting position from the agent's time steps and shifts the traffic lights concerning the agent's initial position such that $\bar{X} = X^i_{x,y} - P_0 \; \forall \; i$ where $0 \leq i < 10$

(c) Use the new Inputs to feed into the model, the other features remain the same. $\hat{Y} = Net(\bar{X})$

(d) Shift the predicted back to the origin stored such that $Y_{real} = \hat{Y} + P_0$

(e) Calculate Metrics

3.6 Metrics

The most important metric used in Waymo prediction is the mean Average Precision metric. Waymo [24] developed an algorithm to calculate the (mAP) mean Average Precision which was their first criterion for comparing different models for their competition score. Waymo released a python library to calculate the mAP metric.

The mAP computes the area under the precision-recall curve by applying confidence score thresholds c_k across a validation set. Consistent with object detection mAP metrics [25], only one true positive is allowed for each object and is assigned to the highest confidence prediction. In this model, we are only predicting one trajectory. Further as described in [26], they seek an overall metric balanced over semantic buckets, some of which may be much more infrequent such as u-turns, so it reports the mean AP over different driving behaviours. The final mAP metric

averages over eight different ground truths which include straight, straight left, straight right, left, right, left u-turn, right u-turn and stationery. The optimal value for mAP that our model should achieve ideally is 1 for perfect predictions which means that we correctly predicted trajectories with a degree of confidence. This is the metric used to compare other models in our results section of the paper.

The first step to computing the mAP metric is determining a trajectory bucket for the ground truth of the objects to be predicted. The buckets include straight, straight-left, straight-right, left, right, left u-turn, right u-turn, and stationery. For each bucket, the following is computed. Trajectory predictions are sorted by confidence value. Using the definition of a miss is defined as followed, a miss is defined as the state when none of the individual K predictions for an object are within a given lateral and longitudinal threshold of the ground truth trajectory at a given time T.

For prediction i, the displacement vector at time T is rotated into the ground truth agent coordinate frame.

$$D_i^j = (\hat{s}_{G_j}^{iT} - s_{G_j}^{iT})\dot{R}_j^T \tag{3.10}$$

where R_j is a rotation matrix to align a unit x vector with the j^{th} agent's ground truth axis at time T. If for any prediction i, $d_y^i < Threshold_{lat}$ and $d_x^i < Threshold_{lon}$ the prediction is considered a correct prediction rather than a miss, otherwise a single miss is counted for prediction i. The miss rate is calculated as the total number of misses divided by the total number of objects predicted. The thresholds change with both velocity and measurement step T as follows:

	$Threshold_{lat}$	$Threshold_{lon}$
T = 3 seconds	1	2
T = 5 seconds	1.8	3.6
T = 8 seconds	3	6

Table 3.1: Comparing mAP Average Results

The thresholds are also scaled according to the initial speed of the agent. The scaling function

is a piecewise linear function of the initial speed v_i.

$$Scale(V_i) = \begin{cases} 0.5, & \text{if } V_i < 1.4m/s \\ 0.5 + 0.5\alpha, & \text{if } 1.4m/s < V_i < 11m/s \\ 1, & \text{if } V_i > 11m/s \end{cases} \tag{3.11}$$

where $\alpha = (v_i - 1.4)/(11 - 1.4)$ and the thresholds are calculated as followed

$$Threshold_{lat}(v_i, T) = Scale(v_i) * Threshold_{lat}(T) \tag{3.12}$$

$$Threshold_{lon}(v_i, T) = Scale(v_i) * Threshold_{lon}(T) \tag{3.13}$$

Using the miss definition above, any trajectory predictions classified as a miss are assigned a false positive and any that are not considered a miss are assigned a true positive. Consistent with object detection mAP metrics, only 1 true positive is allowed for each prediction - it is assigned to the highest confidence prediction, since we only have one prediction it will either be a true positive or false positive per prediction. True positives and false positives are stored along with their confidence in a list per bucket. To compute the metric, the bucket entries are sorted and a precision-recall curve is computed. The mAP metric is computed using the interpolated precision values as described in [27] using the newer method that includes all samples in the computation consistent with the current PASCAL challenge metrics.

After an mAP metric has been computed for all trajectory shape buckets, an average across all buckets is computed as the overall mAP metric.[1]

3.7 Experiment Setup

The parameters for the experiment are as followed. we used a 0.0001 learning rate, for 200 epochs and a batch size of 32. We used an Adam optimizer with 160 hidden cells for the LSTM network and 160 hidden units for the fully connected dense linear layer. To train the network we used (AWS)

[1]Detailed metrics calculation can be found at https://waymo.com/open/challenges/2021/motion-prediction/

Amazon Web Services P2 instance with an Nvidia K80 (GPU) Graphics Processing Unit, 4 (vCPU) Virtual Central Processing Unit, 61 GB (RAM) Random Access Memory and 12 (GB) Gigabyte GPU Memory for 36 hours of training. The dataset was obtained from Waymo [2] server. We implemented the proposed solution using the python programming language and used the Tensorflow/Keras library for the machine learning framework. The code is publicly available in GitHub [3]. It is worth mentioning that training time for 70% of the dataset took a long time due to all the backpropagation computations, it only took 10 minutes for inference on the validation set which is about 15% of the dataset. The inference only required some data processing and passing through a 2-layer network and the computer used for testing only had around 8GB of RAM.

[2] https://waymo.com/intl/en_us/dataset-motion/
[3] https://github.com/wfrei020/motion_prediction_v1

Chapter 4

Data Preprocessing

4.1 Motion Prediction Dataset

Autonomous driving technology is expected to enable a wide range of applications that have the potential to save human lives, ranging from robot-axis to self-driving trucks. The availability of publicly large-scale datasets and benchmarks has greatly accelerated progress in machine perception tasks, including image classification, object detecting, object tracking, semantic segmentation as well as instance segmentation. Waymo created the largest and most diverse multimodal autonomous driving dataset to date, comprising images recorded by multiple high-resolution cameras and sensor readings from multiple high-quality (LiDAR) Light Detection and Ranging scanners mounted on a fleet of self-driving vehicles. The geographical area captured is much larger than the area covered by any other comparable dataset. Waymos' dataset contains a large number of high-quality, manually annotated (3D) 3 Dimensional ground truth bounding boxes for the LiDAR data and (2D) 2 Dimensional tightly fitting bounding boxes for the camera images [28]. They also provide a high level of features for several agents and traffic/road signs. This is one of the largest datasets available with a wide range of features and a variety of locations to use and it was publicly available. There were other datasets available but they had much fewer features and were difficult to obtain.

4.2 Waymo Dataset

Waymo has released a dataset with extracted features from raw data using state-of-the-art methodologies. These features were collected with over 574 hours of data extracted into 103354 segments

of 20 seconds each at 10 Hz sampling. There are 10.8 million objects with tracking IDs, and labels for 3 object classes (pedestrians, vehicles and cyclists). The 3D bounding box for each object is mined for interesting behaviours and scenarios for prediction research such as unprotected turns, merges, lane changes and intersections. Along with object data, there is also map data for each segment which includes locations in San Francisco, Phoenix, Mountain View, Los Angeles, Detroit and Seattle. This dataset was provided as TFRecords and it is split 70% training, 15% Testing and 15% validation data. As mentioned, the dataset is set up in segments, each segment split up into broken 9-second windows (1 second of history and 8 seconds of future data) with a 5-second overlap. sampled at 10 Hz each second has 10 samples of data and since there is 1 second of past data, 8 seconds of future data and 1 sample data point for the current time, that results in a total of 91 samples in a 9-second time sequence.

All coordinates in the dataset are in a global frame with X as East, Y as North and Z as up. The origin of the coordinate system changes in each scene. The origin is an arbitrary point and may be far from the objects in the scene. All units are in meters.

4.3 Dataset Features

Each segment has the following set of components: $roadgraphfeatures$ which has map features of 20000 samples, $statefeature$ which has the feature of the scenario, $past/current/futurestate$ feature indicating the features for 128 objects and finally $past/current/future$ state features for the traffic lights.

1. Scenario
    (a) id: a unique string ID for the scenario represented by the example.
2. Road graph feature: The road graph component has the following set of features.
    (a) dir: a unit vector for each map feature sample point.
    (b) id: a unique ID for the vector map feature each sample is from.
    (c) $type$: A unique integer for each combination of map feature type and properties.

ID	Feature	ID	Feature
1	LaneCenter-Freeway	11	Roadline-SolidSingleYellow
2	LaneCenter-SurfaceStreet	12	Roadline-SolidDoubleYellow
3	BikeLane	13	RoadLine-PassingDoubleYellow
6	RoadLine-BrokenSingleWhite	15	RoadEdgeBoundary
7	RoadLine-SolidSingleWhite	16	RoadEdgeMedian
8	RoadLine-SolidDoubleWhite	17	StopSign
9	RoadLine-BrokenSingleYellow	18	Crosswalk
10	RoadLine-BrokenDoubleYellow	19	SpeedBump
0	unknown types	4	unknown types
5	unknown types	14	unknown types

(d) $valid$: A valid flag for each map sample point.

(e) xyz: The global coordinate positions of the sampled map data points. Map polylines and polygons are sampled at 0.5 meters. The dir feature gives the direction vector from each sample to the next sample.

3. State Features: Each segment has 128 objects and each has 91 samples for the past, current and future states. Each object has several features that were extracted.

(a) $object\_of\_interest$: the object that the rest of the features are based on and it is also used for interaction prediction (another problem).

(b) $difficulty\_level$: the level of difficulty {0-2}.

(c) id: integer id for each object.

(d) $type$:

0	Unset
1	Vehicle
2	Pedestrian
3	Cyclist
4	other

(e) is_sdc: a mask to indicate if the object is the autonomous vehicle.

(f) $tracks\_to\_predict$: a vector of flags to predict up to 8 for submission purposes.

(g) Past/Current/Future States are the agents' features and it has the following:

Feature	Description
Bounding box yaw	the heading angle of each bounding box at each time step.
Valid	a valid flag 0,1 for all objects set to 1 if the element is populated.
Velocity yaw	the yaw angle for each vector velocity vector at each time step in m/s.
Velocity x, y	the x, y component of the velocity of the object at each time step in m/s.
speed	the speed of the velocity of the object at each time step in m/s.
x, y, z	global coordinate for the centre of the object bounding box, the x, y, z coordinate at each time step in meters.

4. Traffic Light Features Past / Current / Future traffic light features with the following:

(a) id: the lane ID controlled by each traffic light.

(b) $state$: the state of each traffic light at each time step.

ID	Feature	ID	Feature
0	unknown	4	stop
1	arrow stop	5	caution
2	arrow caution	6	Go
3	arrow go	7	flashing stop
8	flashing caution		

(c) $valid$: a valid flag for all elements of feature traffic light state 0,1.

(d) x, y, z: global coordinate of the stop light position.

4.4 Preliminary work on Waymo Dataset

The Waymo dataset is very large and to test any models it would be unrealistic to try different models to see. So to test the LSTM network, the dataset was modified to filter out 1 vehicle. This was done by selecting $object_{type} = Vehicle$, testing and training samples were valid and finally the $ID = 0$ as it was discovered that this vehicle with this unique ID had many valid samples. Once the samples were isolated for the sampled vehicle in the dataset, the model design phase started followed by several training iterations trying out different configs and testing using only the x, y coordinated, and velocities. Using these 3 input parameters and 160 cells for the LSTM network we trained this model for 250 epochs using a 10^{-4} learning rate with 32 batch size using (MSE) Minimum Squared Error loss function and saw that the LSTM model that was used was reaching a minimum as you can see in Figure 4.1. These parameters went through a few iterations of trial and error and observed that increasing the number of epochs past 250 did not affect the results.

Similarly, increasing the number of LSTM cells had negative side effects as the training would take much longer and no significant improvements were obtained. Finally, the learning rate was challenging to determine as selecting a larger value would in return prevent reaching a minimum error. Selecting a smaller learning rate would cause the model to reach a minimum at a much slower rate. After many iterations, it was observed that using these parameters was optimal for this subset of the dataset.

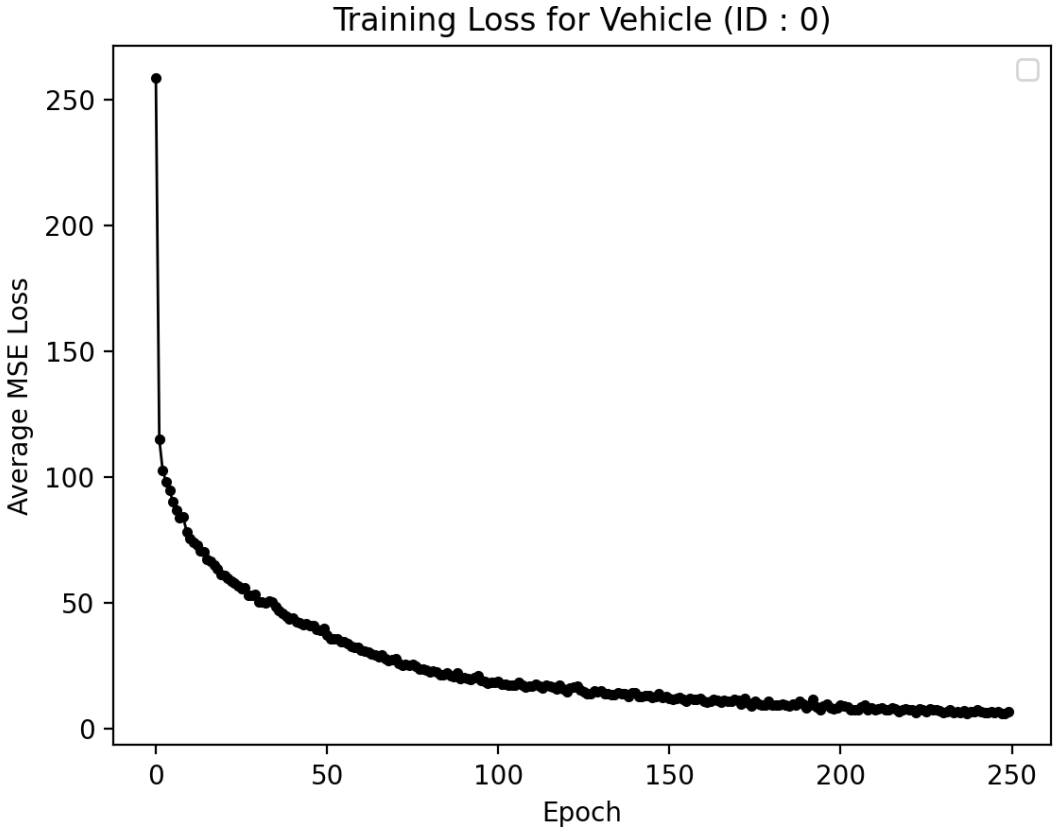


Figure 4.1: Loss during training

As you can see from Figures 4.2 - 4.4 the preliminary training process using a simple LSTM model with very little data it was able to predict some of the motion. This was an extremely important step to study and observe the effect of LSTM networks. With these results obtained from one vehicle, it was clear that LSTM had the potential to solve this problem if more data was given during training and proper data preprocessing. It is also important to note the significant deviations observed in Figure 4.3, this was caused by the vehicle making a sharp turn. These specific trajectories are challenging to predict with just the x, and y coordinates and velocity. To improve these results the use of more data and more input variables such as the yaw of the agents and traffic lights features would improve turn predictions. Another important result was seen in Figure 4.1 as it was confirmation that the minimum squared error loss was a reliable function to use as our main optimization function.

From the results obtained from this preliminary work, we observed that using a subset of the entire Waymo dataset which consisted of 1 vehicle and a data split of 80% for training and 20% for testing, LSTM networks were a feasible solution to motion prediction in a vehicular environment. Using the results from this workflow we were able to identify key steps for our solution such as using the MSE loss, hyperparameters and data translation to start at the origin. With these findings, the next steps to complete this solution were to increase the amount of data to use the entire Waymo dataset and use as many features given by the dataset. This work was a key step in achieving our solution as it helped us separate the problem into smaller tasks and using only a subset of the dataset meant that testing different parameters and building code were easier to prototype and test.

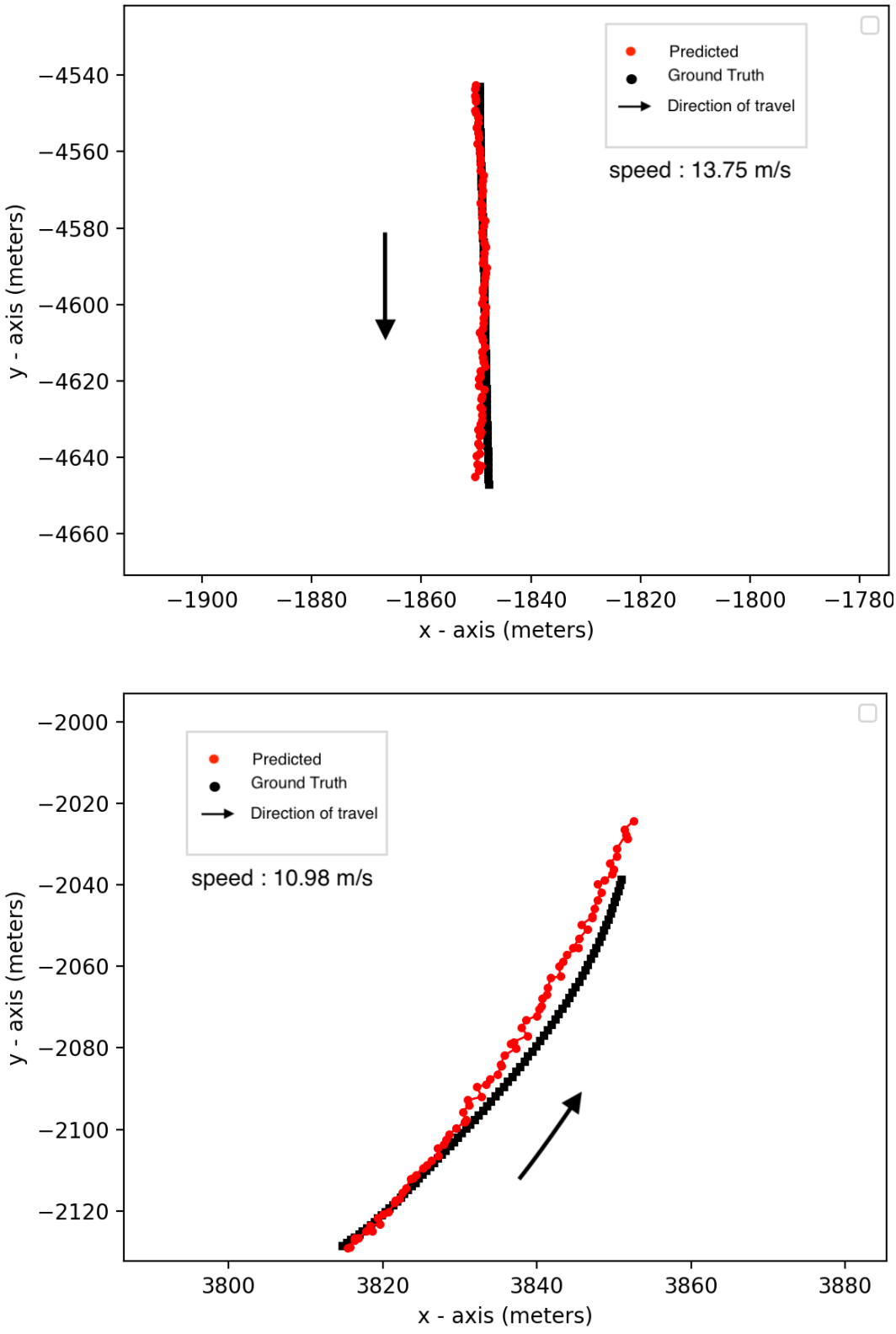


Figure 4.2: Vehicle (ID: 0) results, straight and slight curve at 13.75 m/s and 10.98 m/s

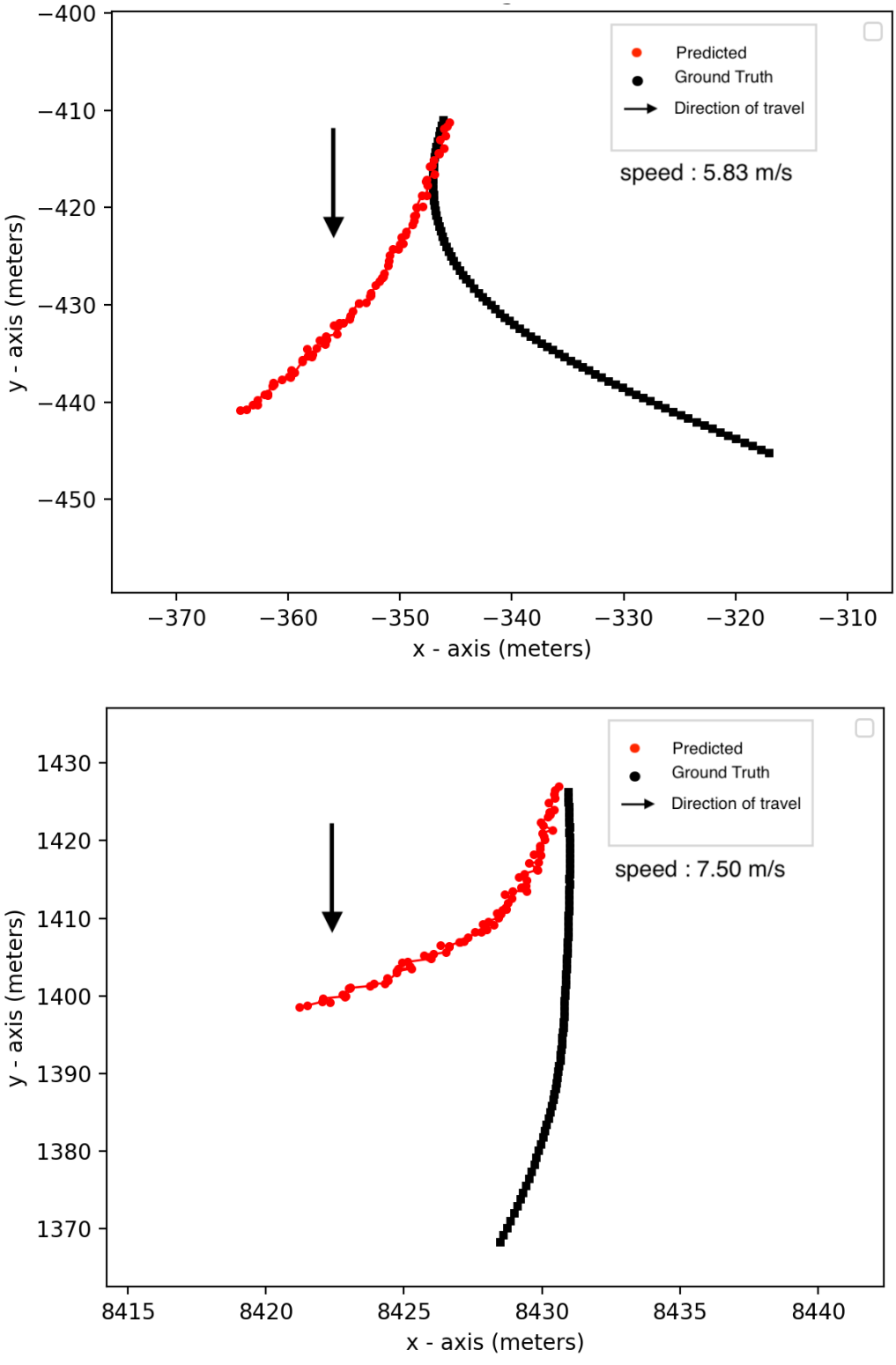


Figure 4.3: Vehicle (ID: 0) results, turns at 5.83 m/s and 7.50 m/s

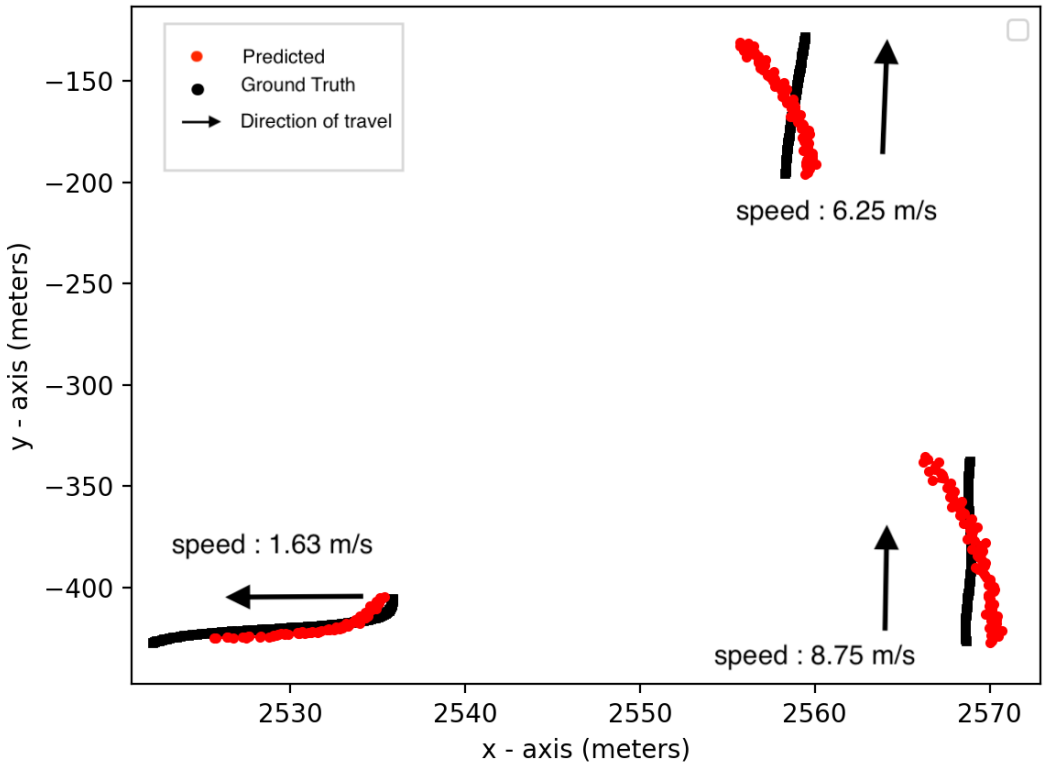


Figure 4.4: Vehicle (ID: 0) results, more results for different trajectories between 1 m/s to 8 m/s

Chapter 5

Accurate predictions under multiple scenarios

5.1 Synthetic Dataset

To first test the effectiveness of LSTM, a synthetic dataset was created that has the given trajectory x^2. To generate this data such that it represents close to the motion prediction task, we set the limits of $0 < x <= 2$ where x is in km. In this dataset, we set the velocity in the x direction to be 40 km/h and took samples at 1Hz. This results in around 1800 samples. The objective is to predict 8 seconds (80 samples) of motion given 1 second of history or 10 samples and 1 current sample. To do this we had to create an RNN training set using windows and strides.

using stride 1

set	input step indexes	output step indexes
1	0 1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 ..90
2	1 2 3 4 5 6 7 8 9 10 11	12 13 14 15 . . . 91
1800	1799 1800 ... 1809	1810 . . . 1889

Table 5.1: Synthetic dataset sequence.

Capturing input sequences using Table 5.1, we have around 1800 sets in the synthetic dataset and we further split it up into 56% training and 44% testing. For the training, we set the learning rate to be 0.0001, used the mean squared error loss, and trained it for 600 epochs and a batch size of 1. Figure 5.1 are the results of the predicted trajectory only. As you can see in Figure 5.1, the curve represents the x^2 curve as expected.

In Figure 5.2 and 5.3, we were able to see that LSTM can effectively predict samples that follow the end of the trained time steps and it becomes much weaker as the prediction increase in time

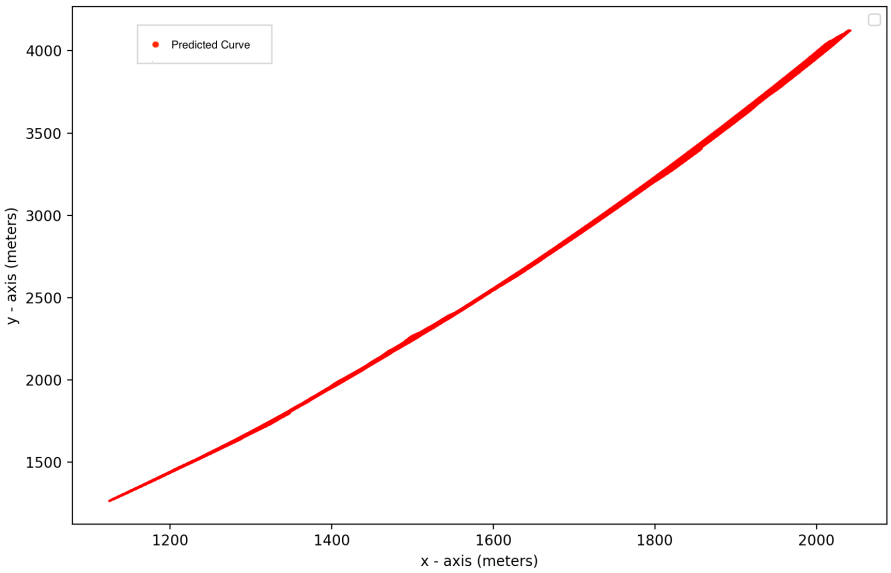


Figure 5.1: X^2 dataset Results using LSTM Network at 11.11 m/s

as seen in Figure 5.3. Even still it was capable of generating decent predictions. This is a clear example of the strength and weaknesses of the LSTM network. However, this does not affect our task as we are predicting a fixed time sequence of 80 samples only.

To further support our analysis we generate several more curves and trained different models to predict each trajectory and the correct trajectories were predicted as seen in Figure 5.4 - 5.5. These results obtained from training a variety of trajectories were used to further support the proposed solution as it can predict many trajectories with different functions.

5.2 Proposed Model Results

5.2.1 Successful Results

For the results, we tested on the Waymo validation dataset. Figure 5.6 - 5.14 are some of the results from the experiment. We will see different scenarios such as straight trajectories, long curve trajectories, lane switching and some turns. Areas, where the model did not do too well, are also discussed and results can be seen in Figures 5.12 - 5.14.

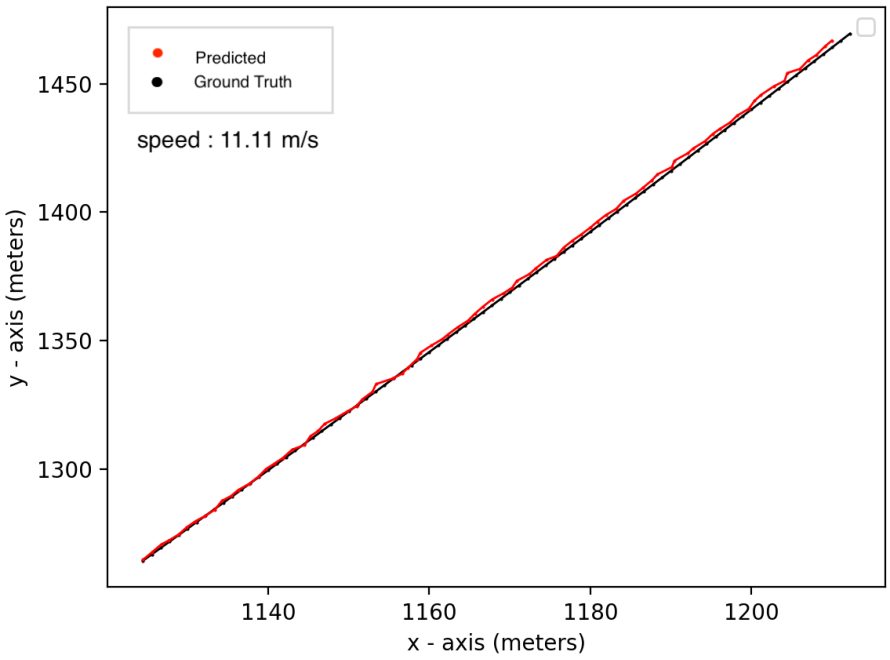


Figure 5.2: X^2 dataset Result at $timestep = 0$ using LSTM Network at 11.11 m/s.

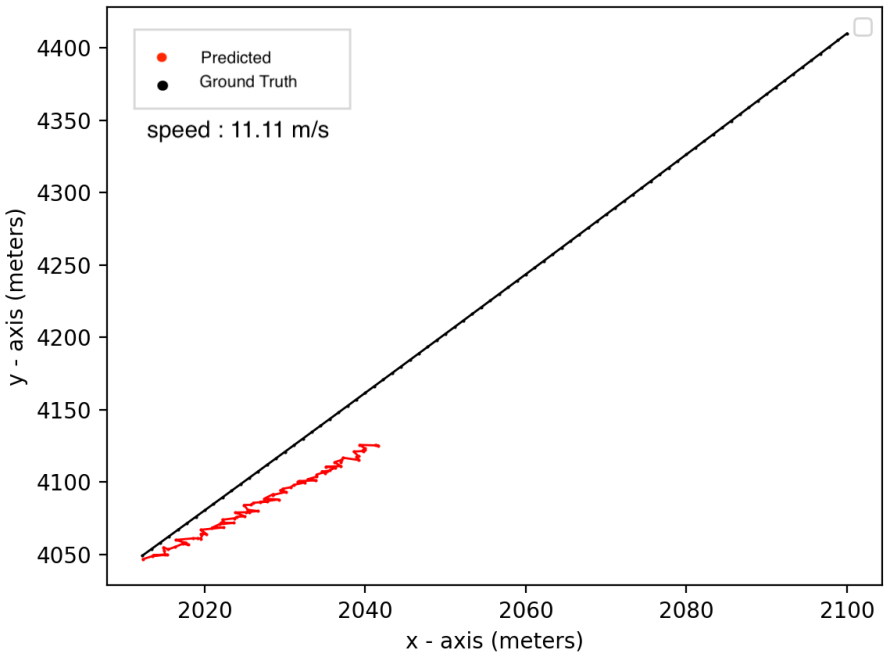
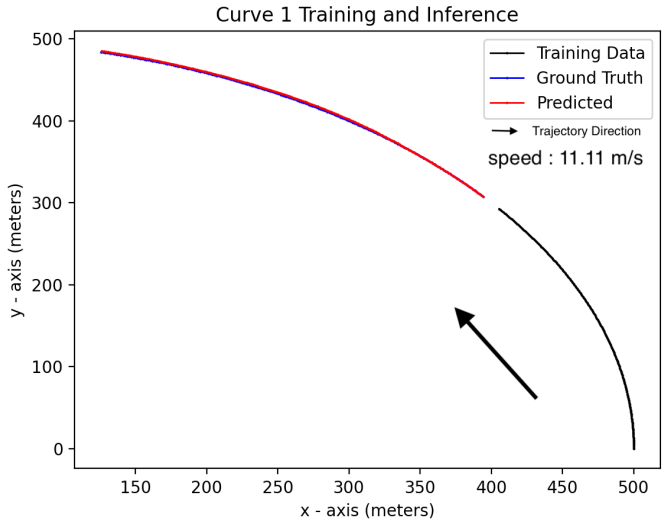
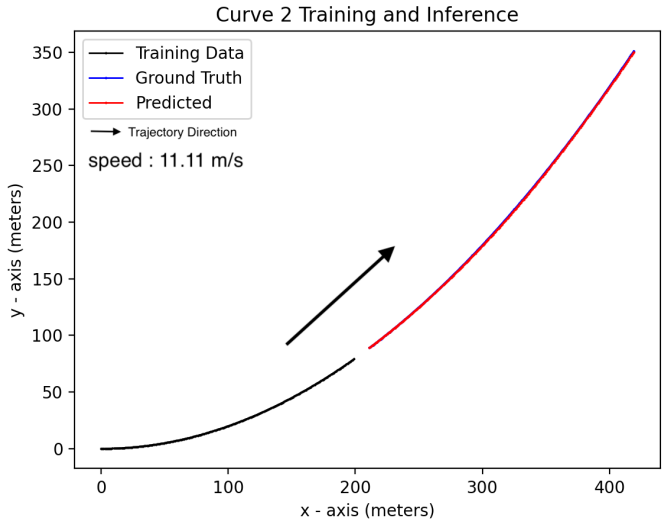


Figure 5.3: X^2 dataset Result at $timestep = 800$ using LSTM Network at 11.11 m/s.





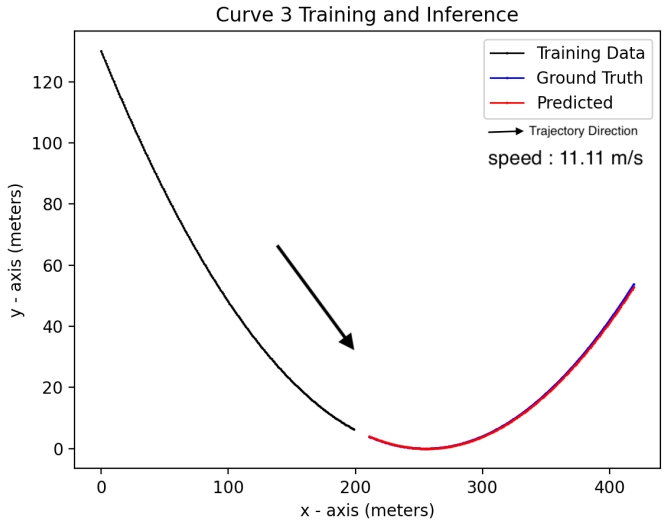


Figure 5.4: Trajectory predictions of 3 distinct functions at 11.11 m/s

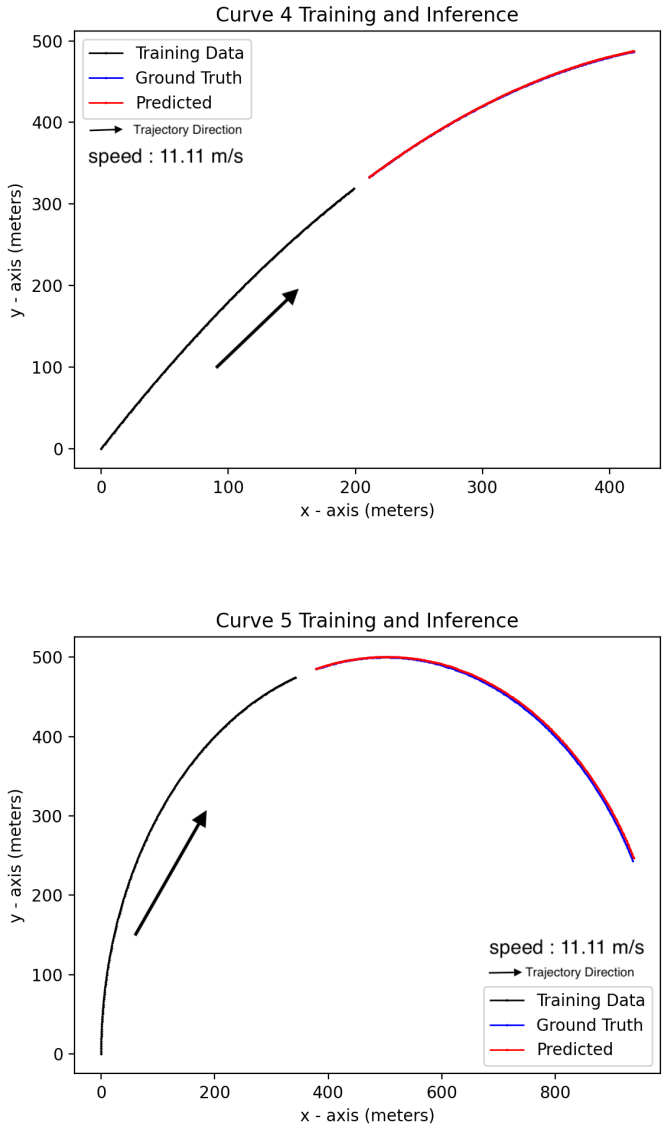


Figure 5.5: Trajectory predictions of 2 distinct functions at 11.11 m/s

5.2.1.1 Straight Trajectory

For Figure 5.6 - 5.7 it is seen that for any straight trajectory the proposed model handles these predictions very well with high accuracy. Although the model inputs are agent features such as speed, coordinates and yaws and traffic light features, the predictions predict very well how far the agent advances in a time of 8 seconds. It can depend on the speed as well as traffic light conditions. This is the easiest and simplest prediction for this kind of task.

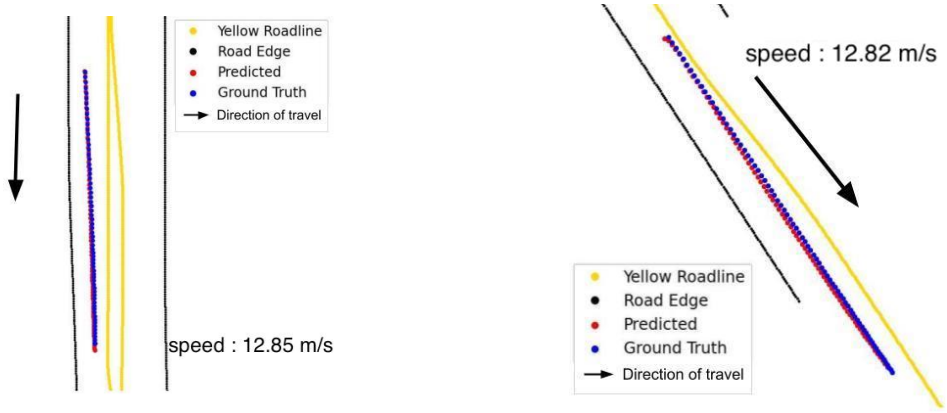


Figure 5.6: Straight Trajectory Predictions of 2 agents at 12.85 m/s and 12.82 m/s

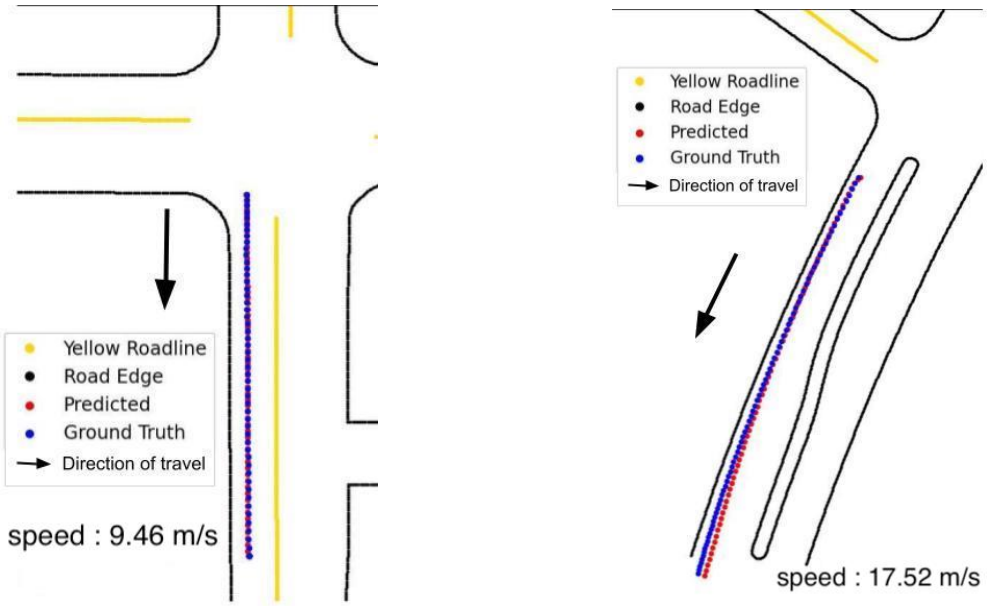


Figure 5.7: Straight Trajectory Predictions of 2 more agents at 9.46 m/s and 17.52 m/s

5.2.1.2 Long Curve Trajectory

In this section, you can see long curves being predicted using the model in Figure 5.8. Even though it does not have any map features it can predict very well the trajectory curves, especially in vehicles. These kinds of problems are more complex for physics equations with too many unknowns. In this case, the proposed model is capable of predicting any curve without the need for too much data and just 1 second of agent history data.

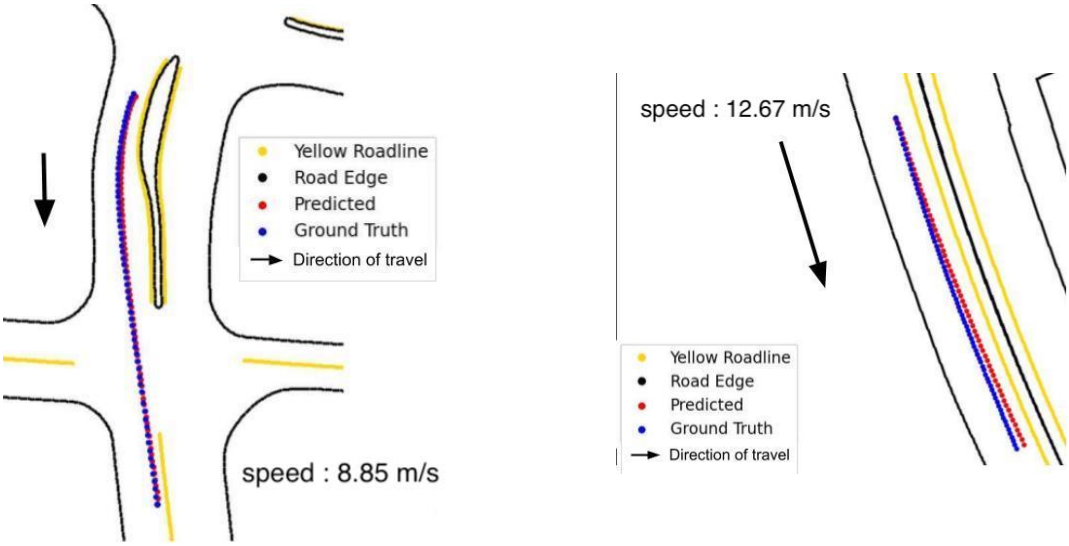


Figure 5.8: Long Curve trajectory Predictions of 2 agents at 8.85 m/s and 12.67 m/s

5.2.1.3 Lane Switching

Next, we can see in Figure 5.9, the model can successfully predict lane switching. This is a much harder task for the model. These types of prediction are greatly affected by other agents (if trying to pass). Since we do not have left turn signal lights as features, predicting lane switches can be tough even for an experienced driver. As drivers, we mainly rely on turn signals to know if a user is going to switch lanes. Our dataset does not have such a feature and hence it finds driver patterns in switching lanes. These kinds of patterns are exactly what motion prediction modelling should be able to identify.

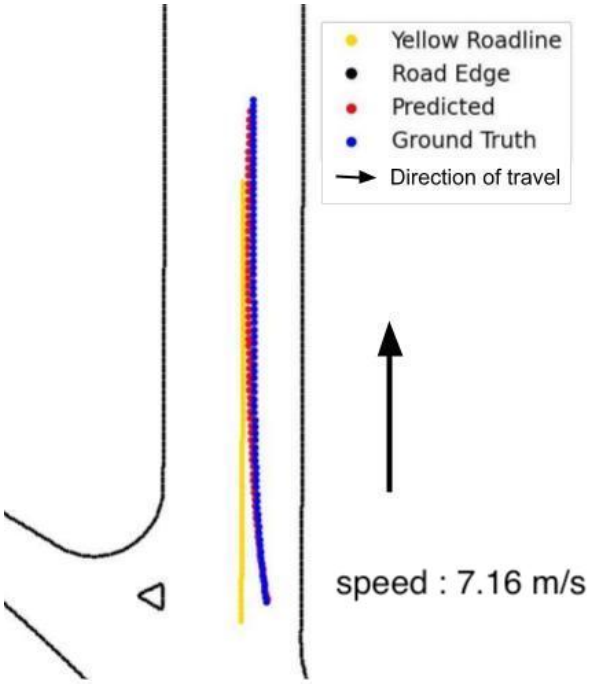


Figure 5.9: Lane Switching Predictions at 7.16 m/s

5.2.1.4 Turn Predictions

Finally, for the last set of predictions, we will discuss turning. As you can see in Figure 5.10, it can predict turns just as long as the input data is of the agent already turning. As you will see in later results, this model has poor results on turns at intersections due to the lack of road graph features. However, we can see that sharp turns can be predicted using the proposed LSTM network.

5.2.2 Areas to Improve

As mentioned, since we did not include any map features we had predictions that could be improved using CNN modelling. You can see in Figure 5.12-5.13, on turns the agent does not have any reference to follow such as a sidewalk or road line for turning or turning into. Another example is in Figure 5.14, when a vehicle is coming towards an intersection, it does not know it needs to slow down or where to stop. As you can see in Figure 5.12-5.13, the model can predict the initial turn but then it is unable to precisely determine the end of the predictions. This can be seen that road features would have helped the model as it is clear the end trajectory is within the lane. However,

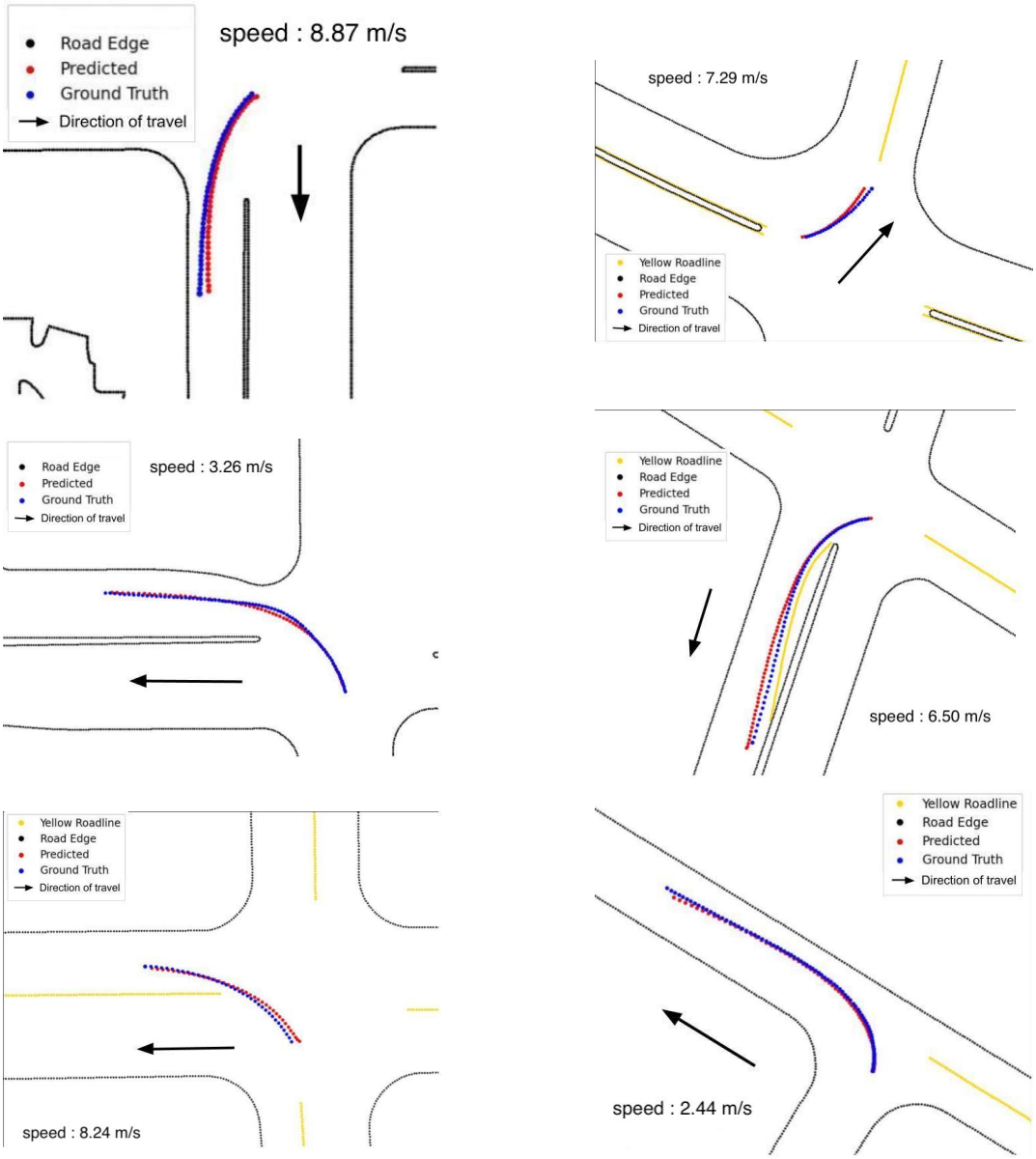


Figure 5.10: Turning Predictions of 6 agents between 2 m/s to 8 m/s

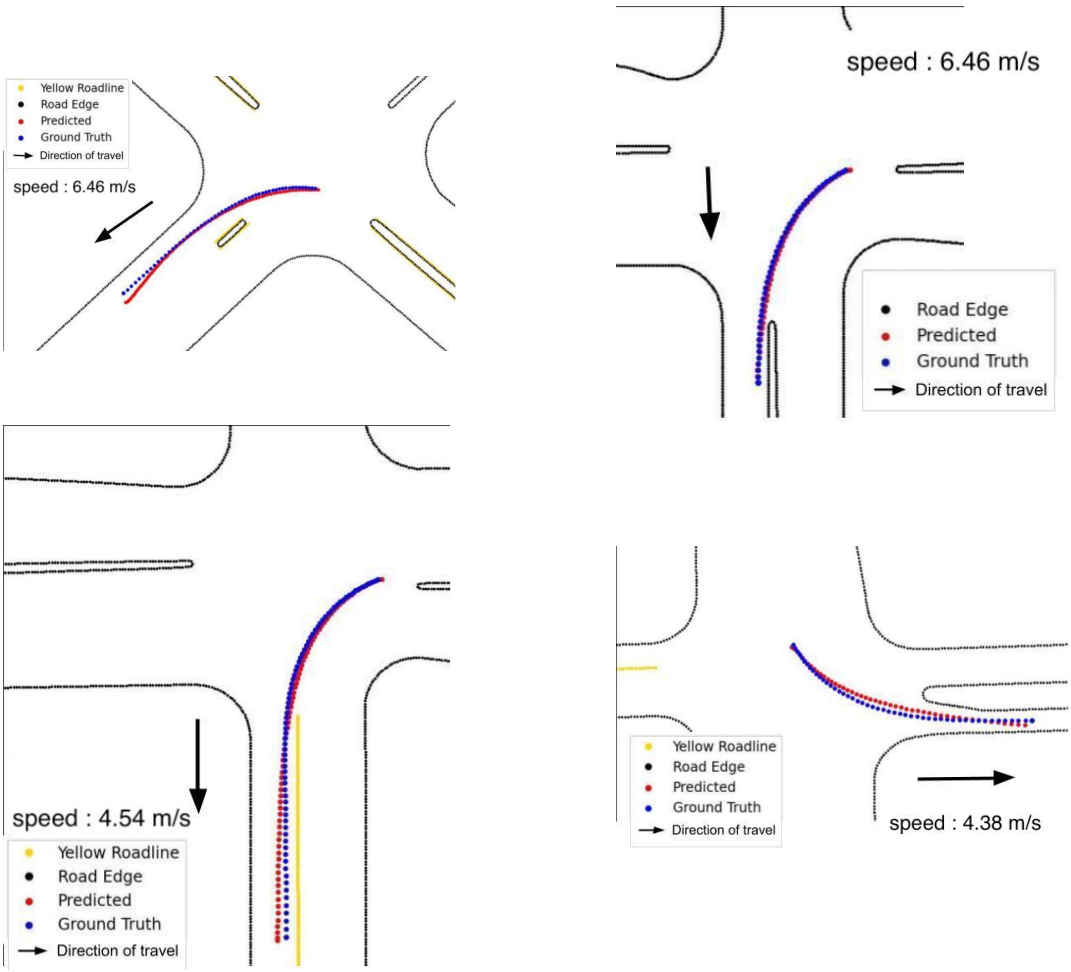


Figure 5.11: Turning Predictions of 4 more agents between 4 m/s to 6 m/s

another possible solution that can be done is reducing the prediction time sequence and instead of predicting for 8 seconds into the future, we can predict 3 seconds and continuously make 3-second interval predictions. The same concept can be said about turning prediction at intersections, Figure 5.14, with more map data and shorter inference intervals, this model would be enough to solve the motion prediction task.

5.3 Model Comparison

In this section, we compare the metrics of the current state-of-the-art models that were sent to Waymo for competition. We calculated the metrics as seen in Table 5.2 and saw that at 3 seconds the average mAP score was much better and degraded a lot as time increased. At 8 seconds the mAP score reduced significantly.

seconds	agent	mAP
3	vehicle	0.224
	cyclist	0.241
	pedestrian	0.499
	average	**0.321**
5	vehicle	0.076
	cyclist	0.088
	pedestrian	0.213
	average	**0.126**
8	vehicle	0.046
	cyclist	0.058
	pedestrian	0.144
	average	**0.082**

Table 5.2: Model Results.

The models LSTM-CV, AE LSTM and diffusion-based RNN can be found at the Waymo competition website for their results [1]. The results that they obtained were taken from the test dataset and mine from the validation dataset. This is done because Waymo did not release the labels for the test dataset for competition reasons, and to see the performance of the proposed model, the

[1] A list of results can be found at the following website https://waymo.com/open/challenges/2021/motion-prediction/

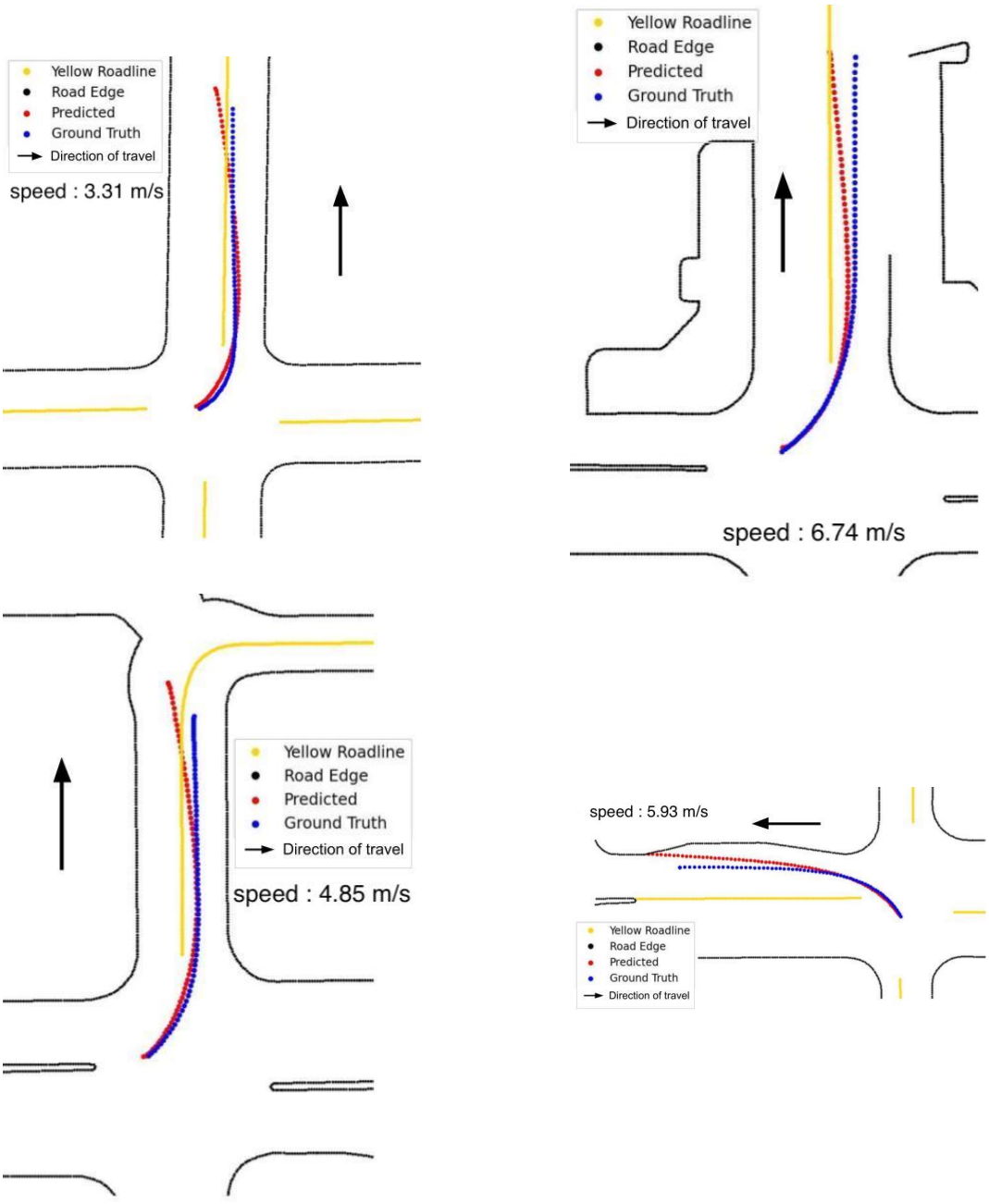


Figure 5.12: Weak Turn Predictions of 4 agents between 3 m/s to 6 m/s

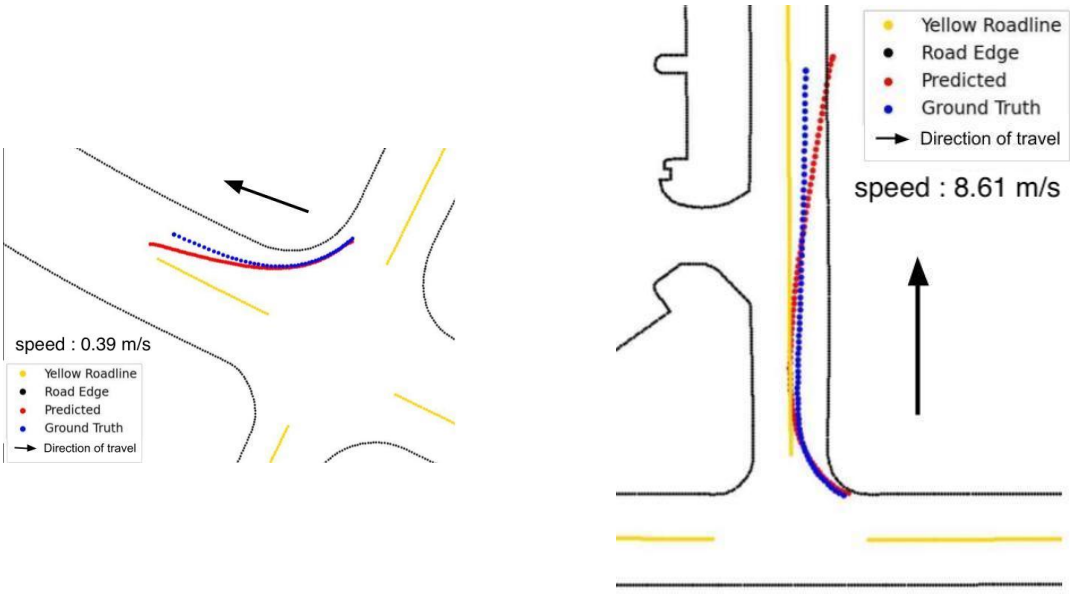


Figure 5.13: Weak Turn Predictions of 2 more agents at 0.39 m/s and 8.61 m/s

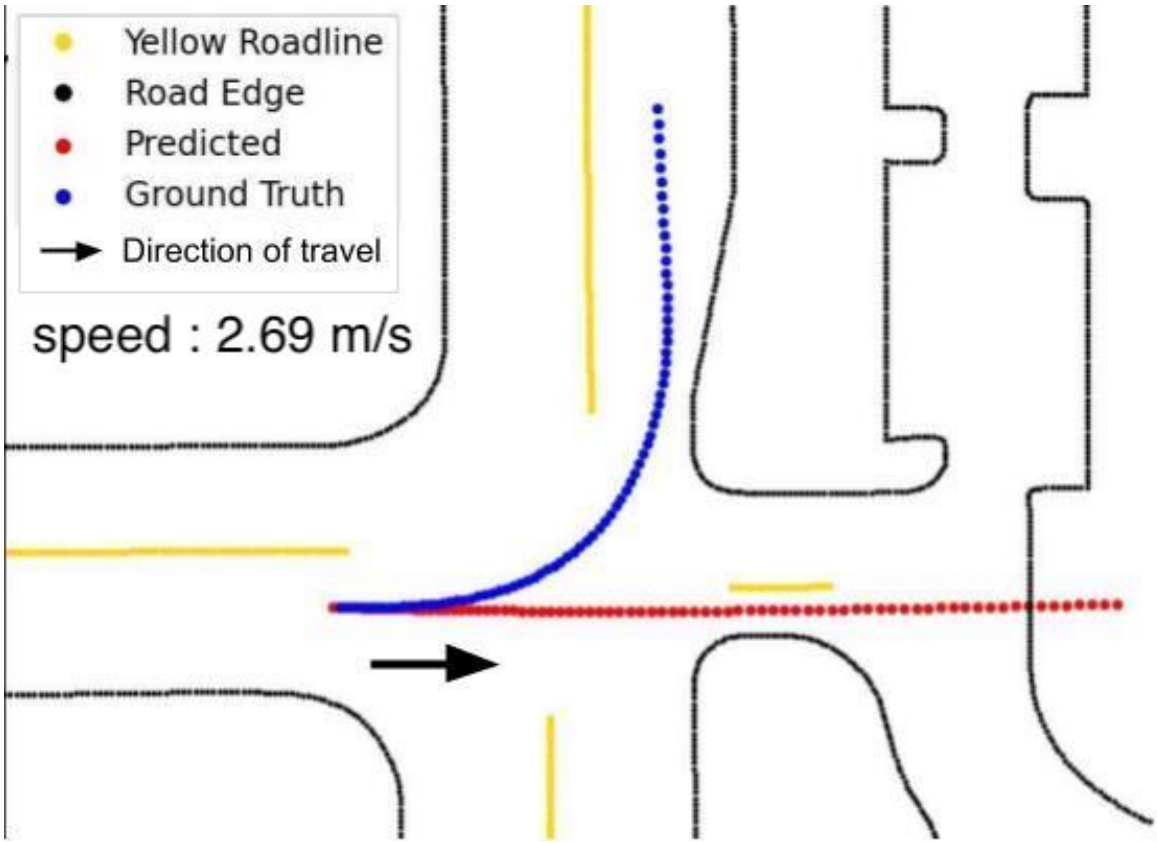


Figure 5.14: Left Turn Predictions at 2.69 m/s

validation dataset was used. The expected results in the test dataset should be similar. As you can see from Table 5.3, the proposed model performed better than the LSTM and RNN-based models at the 3, 5 and 8-second time step. As with most models, it was observed that as time increases there is a degradation in mean average precision. Our model performs with higher precision at 3 seconds and it significantly reduces performance at 5 seconds and even more at 8 seconds. Lastly, since we used the speeds of the agents during our training the results of the speed of the vehicle did not affect the outcome of the predictions.

Model	second	mAP
AE LSTM	3	0.135
	5	0.084
	8	0.047
Diffusion based RNN	3	0.1585
	5	0.1226
	8	0.0809
LSTM-CV	3	0.0643
	5	0.0554
	8	0.0317
Proposed Model	3	**0.321**
	5	**0.126**
	8	**0.082**

Table 5.3: Comparing Results.

Model	Average mAP
Proposed Model	**0.1760**
Diffusion based RNN	0.1207
AE-LSTM	0.0885
AS-LSTM	0.0666
LSTM-CV	0.0505
Basic LSTM	0.0000

Table 5.4: Comparing mAP Average Results

Considering we did not include all the available features given to us, it was at par with some

other models implemented. This was also compared to the 2021 Waymo Motion Challenge. The proposed method was able to outperform other LSTM and RNN-based networks using a smaller LSTM network with only 160 cells. This helped reduce the complexity of LSTM modelling while improving performance.

Chapter 6

Conclusion

In this report, an LSTM-based approach was proposed to the motion prediction problem for the Master of Engineering final project. Using a simple LSTM network and our proposed data preprocessing strategy we were able to obtain a 0.1760 (mAP) mean Average Precision score which was one of the higher accuracies for LSTM model architectures in the Waymo challenge. It does not perform better than some approaches found in Waymo competition entries because the model does not use map features which decreased performance, especially on turning trajectories. However, this project has shown all the areas that come from solving problems using machine learning. These areas include acquiring the dataset and preparing the dataset for designing, training, testing and fine-tuning a model. In practice, creating a model is an important part of machine learning amongst many more areas of research and development that were not discussed in this project but were very important such as computing and efficiency, loss functions, data augmentation and data preparation and finally metric calculations.

6.1 Future Works

One of the main areas to continue working on in this task is to include other agents and map features to minimize the error and find a model that takes into account a lot of important features that this approach does not. This will solve many problems specifically with turns and sudden stop-and-go scenarios. As shown in Figure 5.14, these are targets we can achieve using CNN Modelling with Deep Networks using image processing techniques or Generative adversarial networks similar to [6] and [7]. An approach to be considered is using LSTM-CNN approaches to find hidden features which would serve another set of inputs to the dense linear layer seen in Figure 3.4. This

is a challenging task due to the way the data is provided. This requires a method of generating images from the features given to create the training dataset. This was one of the main reasons the map features were not included in this project due to time and resource constraints. [5] had a good implementation to create raster images to use as inputs to a CNN model but alternative approaches are possible. Another area to work on would be to create three different models to use for the different types of agents. This would include one model for vehicles, pedestrians and cyclists each, and would fine-tune each model according to the agent. This would greatly enhance models for each agent because creating general models means that you are assuming all agents behave the same whether you are driving, cycling or walking. This is not the case because while walking or cycling, one tends to be slower and not travel as fast or as far. Trying to create one general model for three different actions (e.g. driving, cycling, walking) can lead to too much generalization and reduce accuracy. For this task, accuracy is extremely important as it can result in life-changing outcomes. Since this training only used the Waymo dataset, it would be of great value to try this approach on many other datasets to accurately compare the approach with alternative data and see the performance. Since we are working with different types of agents it would help improve results if we created a model for each type of agent that is three different models for vehicles, cyclists and pedestrians. Finally, to compare other models it would be interesting to see how the preprocessing steps would affect other LSTM approaches and implementations and see the performance of other models.

Bibliography

[1] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," 2017.

[2] A. Ip, L. Irio, and R. Oliveira, "Vehicle trajectory prediction based on lstm recurrent neural networks," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–5.

[3] S. O. Ojo, P. A. Owolawi, M. Mphahlele, and J. A. Adisa, "Stock market behaviour prediction using stacked lstm networks," in *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 2019, pp. 1–5.

[4] Z. Gu, Z. Li, X. Di, and R. Shi, "An lstm-based autonomous driving model using a waymo open dataset," *Applied Sciences*, vol. 10, no. 6, p. 2046, Mar 2020. [Online]. Available: http://dx.doi.org/10.3390/app10062046

[5] S. Konev, K. Brodt, and A. Sanakoyeu, "Motioncnn: A strong baseline for motion prediction in autonomous driving," 2021.

[6] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," 2018.

[7] J. Gu, Q. Sun, and H. Zhao, "Densetnt: Waymo open dataset motion prediction challenge 1st place solution," 2021.

[8] H. Xue, D. Q. Huynh, and M. Reynolds, "Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1186–1194.

[9] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[10] S. Lihan, Y. Baoqing, and M. Jie, "A trajectory prediction algorithm for hfvs based on lstm," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 7927–7931.

[11] Y. Zhu, J. Liu, C. Guo, P. Song, J. Zhang, and J. Zhu, "Prediction of battlefield target trajectory based on lstm," in *2020 IEEE 16th International Conference on Control Automation (ICCA)*, 2020, pp. 725–730.

[12] Z. Zhang, G. Ni, and Y. Xu, "Ship trajectory prediction based on lstm neural network," in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2020, pp. 1356–1364.

[13] Z. Zhong, R. Li, J. Chai, and J. Wang, "Autonomous vehicle trajectory combined prediction model based on c-lstm," in *2021 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, 2021, pp. 1–6.

[14] L. Hou, L. Xin, S. E. Li, B. Cheng, and W. Wang, "Interactive trajectory prediction of surrounding road users for autonomous driving using structural-lstm network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4615–4625, 2020.

[15] P. Han, W. Wang, Q. Shi, and J. Yang, "Real-time short- term trajectory prediction based on gru neural network," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–8.

[16] D. Wei, "Prediction of stock price based on lstm neural network," in *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, 2019, pp. 544–547.

[17] S. Liu, G. Liao, and Y. Ding, "Stock transaction prediction modeling and analysis based on lstm," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2018, pp. 2787–2790.

[18] Y. Fu, S. Sen, J. Reimann, and C. Theurer, "Spatiotemporal representation learning with gan trained lstm-lstm networks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 548–10 555.

[19] P. S. Budi Cahyo Suryo, I. Wayan Mustika, O. Wahyunggoro, and H. S. Wasisto, "Improved time series prediction using lstm neural network for smart agriculture application," in *2019 5th International Conference on Science and Technology (ICST)*, vol. 1, 2019, pp. 1–4.

[20] N. Tax, "Human activity prediction in smart home environments with lstm neural networks," in *2018 14th International Conference on Intelligent Environments (IE)*, 2018, pp. 40–47.

[21] R. C. Staudemeyer and E. R. Morris, "Understanding lstm – a tutorial into long short-term memory recurrent neural networks," 2019.

[22] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar 2020. [Online]. Available: http://dx.doi.org/10.1016/j.physd.2019.132306

[23] Y. Du, N. Cui, H. Li, H. Nie, Y. Shi, M. Wang, and T. Li, "The vehicle's velocity prediction methods based on rnn and lstm neural network," in *2020 Chinese Control And Decision Conference (CCDC)*, 2020, pp. 99–102.

[24] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov, "Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset," 2021.

[25] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[26] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[27] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, 06 2010.

[28] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, S. Zhao, S. Cheng, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," 2020.