Motion Prediction Using LSTM Network

Walter A. Freire
andrei.freire@ryerson.ca
Ryerson University, Toronto, Canada.

***Abstract*—Autonomous driving has been rapidly increasing demand and interest over the past several years and one of the most important obstacles an intelligent driver needs, is to understand the behaviour of other road users that share the road. It is critical to understand if a pedestrian/cyclist are 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 paper explores the Waymo motion prediction challenge and attempts to solve the problem using the Waymo motion dataset. The challenge is to predict 8 seconds into the future given 1 seconds of history. This paper will look into Long Short Term Memory networks since it behaves like a special kind of RNN, which have been shown to be strong models for forecast prediction that are already being used such as stock price prediction and more forecasting models. The proposed method is implemented in python using the Tensorflow/Keras framework and is available in Github *https://github.com/wfrei020/motion_prediction_v1*.**

I. Introduction

The rise of Autonomous Vehicles (AV), has been exponentially increasing, with current market estimated at $22.22 billion and estimated to rise to $75.95 billion by 2027. Autonomous vehicles use what is known as embodied artificial intelligence that poses many benefits and risks. These benefits involve reducing human errors and avoid crashes caused by humans. In order to achieve level 5 vehicle autonomy, which involves no human interaction and all driving is automated, the embodied AI will introduce machine error which must be mitigated. An important component of driving is the ability to quickly predict another agents (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 a 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 or trying to pass another vehicle or just making a right turn by looking at road graph lines. This capacity to predict motion of other drivers is what allows us to avoid collisions and make safe driving possible. Similarly, autonomous vehicles require the capabilities of predicting another agents motion to avoid collisions and create safer autonomous vehicles. Motion prediction is the task of predicting the future trajectory of an agent wether 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 aspect of machine learning to develop models that are able to predict human motion for 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 goto method for many time sequence tasks as it has the capabilities to learn data that is distributed over short periods of time.

For the MEng project in electrical and computer engineering, this paper explores possible solution to this problem. This paper discusses the background in motion prediction, the dataset that was used in the project, the proposed solution to attempt to solve this problem and the results obtained from the experiments as well as future development.

II. Background

Trajectory Prediction is a highly researched topic and is considered a time series problem. Time series analysis are of two types, univariate analysis in which the inputs to the time series network is of a single input or 1 feature and a multivariate analysis which consist of multiple inputs which vary over time. [2]. Trajectory prediction has had on going research but recently Waymo introduced their motion prediction challenge in which many researchers attempted to solve the trajectory prediction problem. Waymo conducted a baseline [3] using the Long Short Term Memory approach LSTM. Konev *et al.* [4] had different approaches such as using methods of converting the raw dataset into raster graphics and used the generated images to train a model using a CNN architecture that was pretrained using Image-net and used the negative log likelihood for the loss function. Other methods involved using target driven approaches that involved extracting 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 [5]. These implementation also rely on using RNN based networks with convolutional architecture to extract map and agent features.

More recent there have been goal-based methods such as [6] that achieves really high performance using goal based trajectory prediction. Their method relies on map and agent dense encoding to make their predictions. This approach ended up being one of the best performance for the Waymo motion prediction challenge.

Since motion prediction is based on a basis of temporal prediction, a straight forward approach to solve a problem

like this is using a recurrent neural network [3] [7] [1] [8]. 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 in order to generate potential outputs of the future trajectories. Any one of these methods have potential possibilities to predict trajectories to some extent but each have their problems such as high computing costs, or weaker performance.

Trajectory predictions has also been studied in defence strategies of unmanned vehicles as it is useful to predict an attackers trajectory to obtain enemies position, objectives and intents. [9] [10] LSTM networks have been used in a variety of application one of these applications to predict trajectories of hypersonic flight vehicles to improve defence of HFV from another attacking HFV . Here they 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 leverage LSTM networks for their solution. [9]

Trajectory prediction is also being used in ship trajectories using the speed, course, longitude and latitude difference. In [11] they leveraged the MSE loss to achieve their results.

The authors in [12] built a C-LSTM network (Convolutional LSTM network) which consisted of an input layer, convolutional layer and LSTM multivariate trajectory layer. This is a very common use of CNN-LSTM network. They would first extract feature 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 scenario of images in order to get the most features out of the inputs. This clustering step allowed the model to find hidden features which improves convergence and accuracy of the model. However It only focuses on one highway, and also shows no comparison to other models. Similarly [13] only predict trajectories of taxis in urban areas. There are other methods that use different techniques to solve different types of trajectory predictions problems such as using 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. Finally for real time trajectory prediction the authors of [15] relied on GRU to do fast computational prediction to meet real time requirements.

LSTM has seen great success in stock prediction which is another time series task and many have been studied and implemented using LSTM [16] [2] [17] . However LSTM networks have also been used for other task such as creating an action prediction task for general purpose robots [18]. 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 is monitored [19]. Finally activity prediction in smart homes environment to detect future events [20]. LSTM networks have a wide variety of applications in time series analysis and new models and use cases are continuously emerging.

III. Waymo Challenge

A. 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 ID, labels for 3 object classes (pedestrians, vehicles and Cyclist). 3D bounding box for each object 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% percent validation data. As mentioned, the data set 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 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.

B. Dataset Features

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

1) Scenario
   a) id: a unique String ID for the scenario represented by this 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 points
   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
1	LaneCenter-Freeway
2	LaneCenter-SurfaceStreet
3	BikeLane
6	RoadLine-BrokenSingleWhite
7	RoadLine-SolidSingleWhite
8	RoadLine-SolidDoubleWhite
9	RoadLine-BrokenSingleYellow
10	RoadLine-BrokenDoubleYellow
11	Roadline-SolidSingleYellow
12	Roadline-SolidDoubleYellow
13	RoadLine-PassingDoubleYellow
15	RoadEdgeBoundary
16	RoadEdgeMedian
17	StopSign
18	Crosswalk
19	SpeedBump
0, 4, 5, 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 poly lines 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 sample 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 feature 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 object set to 1 if 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 objects velocity at each time step in m/s
speed	the speed of the objects velocity 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
0	unknown
1	arrow stop
2	arrow caution
3	arrow go
4	stop
5	caution
6	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.

IV. Project Background

To solve this problem it is necessary to split up the tasks in order to fully understand what approach can be taken. In this section we will discuss feed forward networks, RNN and LSTM networks [21] [22].

A. Feed Forward Neural Network

In FFNN, 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 output present signals to the environment via hidden neurons that processes the input and output signals. Feed forward networks are loop free as seen in Figure 1 and fully connected which means that each neuron provides an input to each neuron in the following layer and none of the weights effect a neuron in the previous layers.

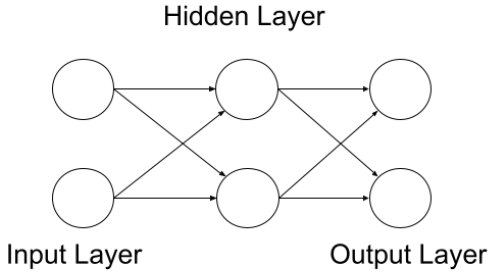


Fig. 1: Feed Forward Neural Network

B. RNN

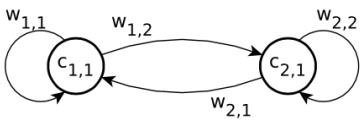
The recurrent neural networks are dynamic systems, they have the internal state at each time step of the classification. This is due to circular connections between higher and lower layer neurons as seen in figure 2a. This feedback connection enable RNNs to propagate data from earlier events to current processing steps which create this memory of time series events.

C. Vanishing Error in RNN

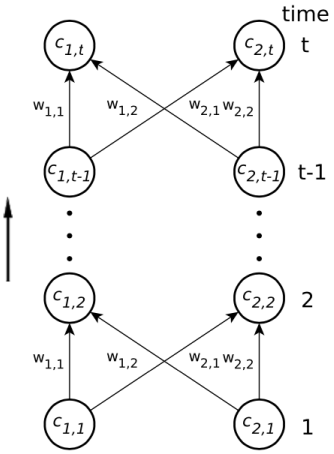
Standard RNN can not bridge more than 5 - 10 time steps due to the back propagation error signals. The signals tend to either grow or shrink with every time step. over many time steps the error typically blows up or vanishes. When the error blows up this leads to oscillating weight and when the error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

D. LSTM Networks

LSTM can learn how to bridge minimal time lags of more than 1000 discrete time steps. This is done using Constant Error Carousels CEC, which enforce a constant error flow within special cells. Access to these cell are handled by multiplicative gate units which learn when to grant access. CECs is the preservation of error where short term memory storage is achieved for extended periods of time. 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 are needed because of the continuous input being insert into the RNN and can have conflicting weight update signals. The memory blocks are used to compute the weight updates that is contributed by the previous input and other neurons. In order to avoid this problem of conflicting



(a)



(b)

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

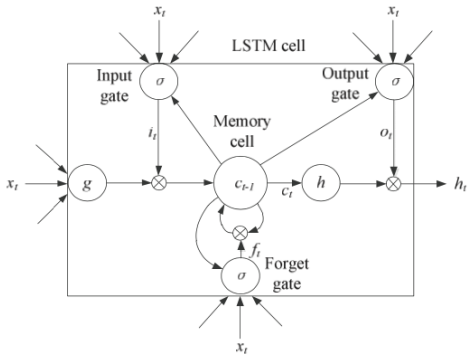


Fig. 3: Structure of LSTMCell [23]

E. Strength and limitation 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 forget gates which allows for continuous prediction because they make cells completely forget their previous state, preventing unbiased prediction. One of the main problem with LSTM is that the number of memory blocks in networks does not change dynamically so the memory of the network is ultimately limited. This mean 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.

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 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].

V. Artificial Dataset

To first test the effectiveness of LSTM, I create an artificial dataset that has the trajectory x^2. To generate this data such that it represents closely to the motion prediction task, we set the limits of $0 < x <= 2$ where x is in km. In this dataset I 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. In order to do this we had to create RNN training set using window and strides.

	set	input step indexes	output step indexes
Stride 1	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

Capturing input sequences using table **??**, we have around 1800 set in the artificial dataset and we further split it up into 56% training and 44% testing. For the training we set learning rate to be 0.0001 and used the mean squared error loss, trained it for 600 epochs and batch size of 1. Figure 4 are the results of the predicted trajectory only. As you can see in figure 4, the curve represent the x^2 curve as expected.

In figure 5 and 6 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. Even still it was capable of generating

Fig. 4: X^2 dataset Results using LSTM Network

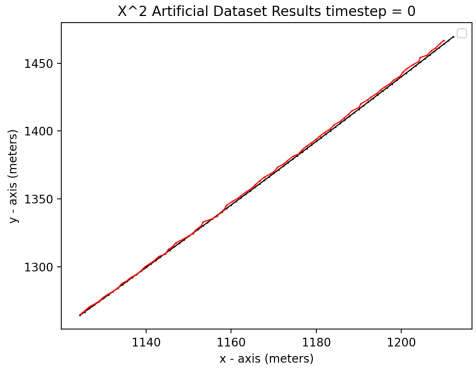


Fig. 5: X^2 dataset Result at $timestep = 0$ using LSTM Network. (red: predicted, black: target)

decent predictions. In order 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 7

VI. Preliminary work on Waymo Dataset

The Waymo dataset is a very large dataset and in order to test any models it would be unrealistic to try different models to see. So in order to test the LSTM network, I modified the dataset to filter out 1 vehicle. This was done selecting

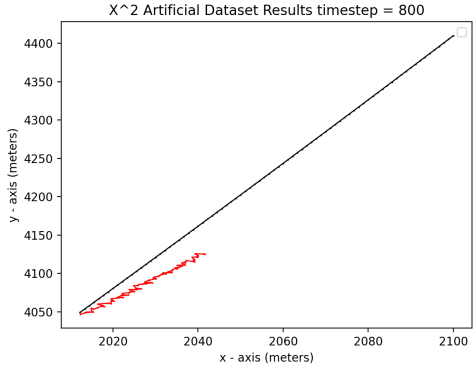


Fig. 6: X^2 dataset Result at $timestep = 800$ using LSTM Network. (red: predicted, black: target)

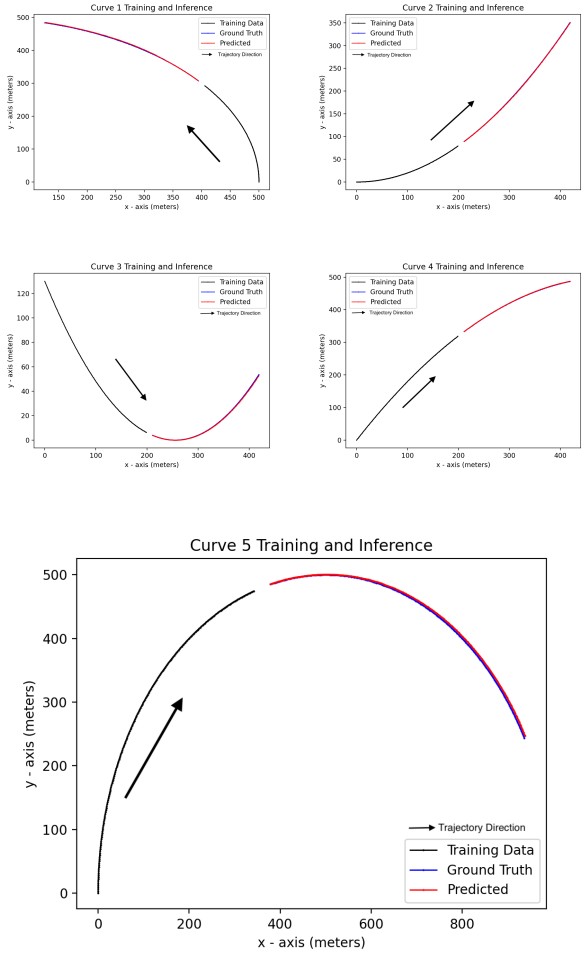


Fig. 7: Vehicle (ID: 0) results

$object_{type} = Vehicle$, 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 I isolated the vehicle dataset and started to train and test using x, y, and velocities. We stopped the training after 250 epoch and saw that the LSTM model that was used was reaching a minimum as you can see in figure 8

Some examples of the results are seen in figure 9. You can clearly see that the model can predict certain movements easily such as small curves and straight lines. However, predicting turns are very complicated without using any map features.

VII. Proposed Solution

The proposed solution solution only takes in the agent/object state features and the traffic light features that are near the objects. 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 LSTM network on time series data for position. We will discuss later but some problem this raises are that we 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

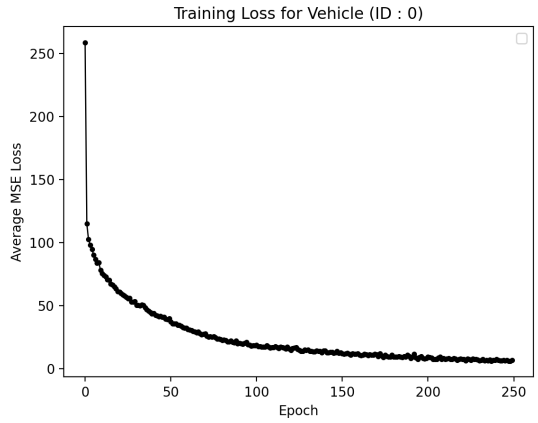
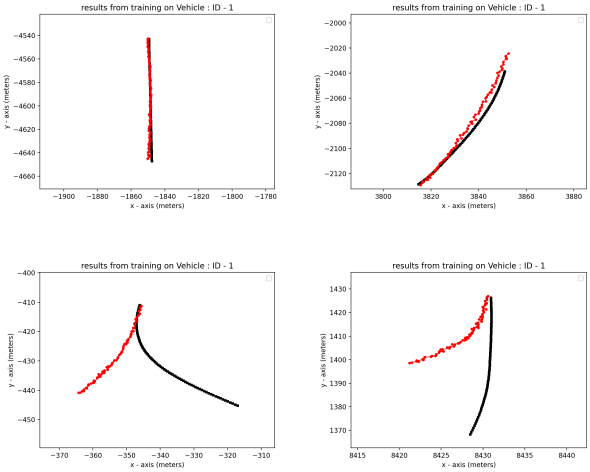


Fig. 8: Loss during training



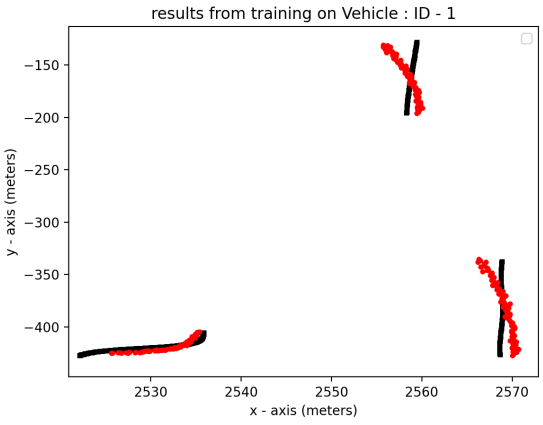


Fig. 9: Vehicle (ID: 0) results

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.

A. Dataset Prep

For the proposed solution the input features that will be used as input into our 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 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 our inputs with 11 time steps for each.

B. 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 10.

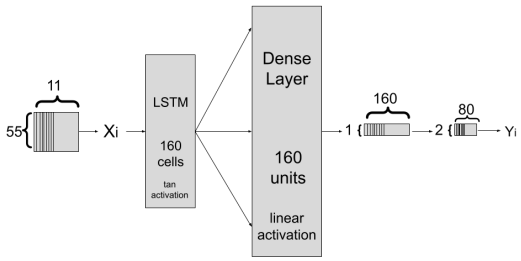


Fig. 10: Motion Model

C. Loss

For simplicity i used a mean squared error loss to find the error between the predicted sample and ground truth target. The error was calculated as followed

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

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

D. Algorithm

In order to overcome the varying position for multiple agents we found out that each agents displacement was between 0 - 300 meters. So in order to greatly improve our results and model training we shifted all of the starting positions of each agent to the origin (0, 0) and then used the new x, y coordinates as our inputs to the models. In figure 11 you can see an example of the training data shift.

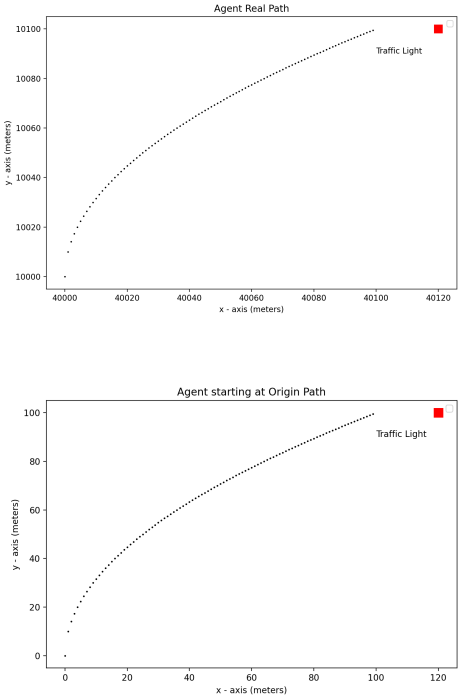


Fig. 11: Agent shifted to origin for training

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

1) Training
   a) store the Agents starting position $P_o = (X_x^0, X_y^0)$
   b) subtract the agents starting position to the agent time steps and shift the traffic lights with reference to agent initial position such that $\bar{X} = X_{x,y}^i - P_0 \ \forall \ i$ where $0 \leq i < 10$ and $\bar{Y} = Y_{x,y}^i - P_0 \ \forall \ i$ where $10 \leq i < 91$. X is the input features and Y are the targets.
   c) Use our new Inputs and target to feed into our 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 next agent
2) Testing
   a) store the Agents starting position $P_o = (X_x^0, X_y^0)$
   b) subtract the agents starting position to the agent time steps and shift the traffic lights with reference to agent initial position such that $\bar{X} = X_{x,y}^i - P_0 \ \forall \ i$ where $0 \leq i < 10$
   c) Use our new Inputs to feed into our 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

E. Metrics

The most important metric used in the Waymo prediction is the mean Average Precision metric. Waymo [24] developed an algorithm in order to calculate the mAP which was their first criteria of comparing different models for their competition score. Their second criteria was to compare the miss rate of the trajectory predicted with respect to the ground truth. Waymo released a library to calculate the mAP and miss rate metrics.

VIII. 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 P2 instance with a Nvidia K80 GPU, 4 vCPU, 61 GB RAM and 12 GB GPU Memory for 36 hours of training. The dataset was obtained from Waymo [1] server. We implemented our solution using the python programming language and used the Tensorflow/Keras library for our machine learning framework. The code is publicly available in our Github [2].

IX. Result Discussion

For the results we tested on Waymo validation dataset. Figure 12 - 16 are some of the results from the experiment.

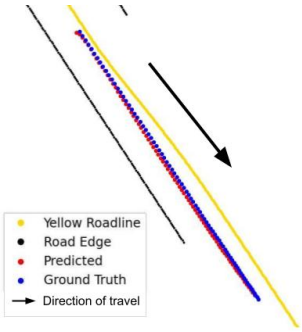


Fig. 12: Straight trajectory

As mentioned, since we did not included any map features we had predictions that can be improved using CNN modelling. You can see in Figure 15, on turns the agent does not have any referenced to follow such as a side walk or road line for turning or turning into. Another example is in Figure 16, when a vehicle is coming towards an intersection, it does not know it needs to slow down or where to stop.

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

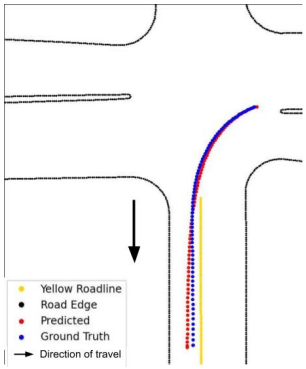


Fig. 13: Curve trajectory

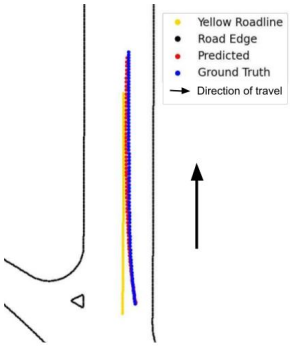


Fig. 14: Lane Switching

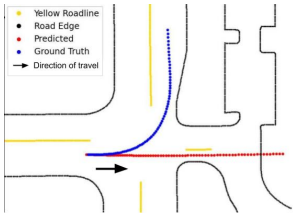


Fig. 15: Right Turn

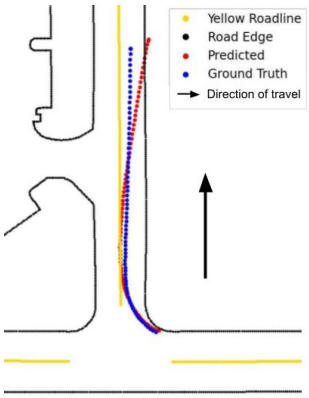


Fig. 16: Lane Switching

We calculated the metrics as seen in table I 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 scored reduced significantly.

seconds	agent	mAP	Miss Rate
3	vehicle	0.224	0.599
	cyclist	0.241	0.547
	pedestrian	0.499	0.271
	average	**0.321**	**0.472**
5	vehicle	0.056	0.886
	cyclist	0.068	0.806
	pedestrian	0.203	0.496
	average	**0.109**	**0.729**
8	vehicle	0.034	0.927
	cyclist	0.026	0.870
	pedestrian	0.144	0.564
	average	**0.068**	**0.787**

TABLE I: Our Model Results.

For the models multipath++, AE LSTM, Basic LSTM and Dense TNT can be found at the Waymo competition website for their results [3]. 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 in order to see the performance of my model, I used validation dataset. I expect the results in the test dataset to be similar. As you can see from table II, my model performed better than the AE LSTM model at the 3 , 5 and 8 second time step. As you can see for the Multipath++ and DenseTNT, which used map and other agent features for their predictions had superior results and are at the top of the charts for their models.

Model	second	mAP	Miss Rate
Basic LSTM	3	0.000	0.002
	5	0.000	0.002
	8	0.000	0.002
AE LSTM	3	0.135	0.587
	5	0.084	0.672
	8	0.047	0.749
Multipath++	3	0.481	0.099
	5	0.417	0.127
	8	0.329	0.176
DenseTNT	3	0.406	0.117
	5	0.320	0.169
	8	0.259	0.186
Ours	3	0.321	0.472
	5	0.109	0.729
	8	0.068	0.787

TABLE II: Comparing Results.

For the current competitors in the Waymo challenge I am in spot 18 out of all the state of the art attempts without using

[3]https://waymo.com/open/challenges/2021/motion-prediction/

map features or other agents. Table III shows the results of Waymo challenge. Considering we did not include our all the available features given to us, it was at par with some other models implemented.

Place	Model	mAP
1^{st}	Multipath++	0.4092
16^{th}	CNN-MultiRegressor	0.1944
17^{th}	xyzResnet18	0.1927
18^{th}	ours	0.1660
19^{th}	Diffusion_based_RNN	0.1207
20^{th}	RasterMP	0.0986

TABLE III: Comparing Average Results to finalist.

X. Further works

One of the main areas to continue working on 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 15, these are targets we can achieve using CNN Modelling with Deep Network using image processing technique or Generative adversarial networks similar to [5] and [6]. An approach I am thinking of taking is using LSTM-CNN Approaches to find hidden features which would serve another set of inputs to the dense linear layer seen in figure 10. 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 I could not include map features in this project due to time and resource constraints. [4] had a good implementation to create raster images to use as inputs to a CNN model but I had other ideas that did not align with their solution. 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, pedestrian and cyclist each, and would fine tune each model according to the agent. This would greatly enhance models for each agent because creating a general models means that you are assuming all agent behaves 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.

XI. Conclusion

In this report, we proposed an LSTM based approach to the motion prediction problem for the Master of Engineering final project. It does not perform better than than some approaches found in Waymo competition entries because we did not use map features which decreased performance. However this project has shown all the areas that comes 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.

References

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

[2] 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)*, pp. 1–5, 2019.

[3] Z. Gu, Z. Li, X. Di, and R. Shi, "An lstm-based autonomous driving model using a waymo open dataset," *Applied Sciences*, vol. 10, p. 2046, Mar 2020.

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

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

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

[7] 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)*, pp. 1186–1194, 2018.

[8] 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.

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

[10] 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)*, pp. 725–730, 2020.

[11] 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)*, pp. 1356–1364, 2020.

[12] 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)*, pp. 1–6, 2021.

[13] 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)*, pp. 1–5, 2021.

[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)*, pp. 1–8, 2019.

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

[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)*, pp. 2787–2790, 2018.

[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)*, pp. 10548–10555, 2020.

[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, pp. 1–4, 2019.

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

[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.

[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)*, pp. 99–102, 2020.

[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.