Trajectory Prediction using nuScenes Dataset

Pheh Jing Jie[1], Kanashima Hatsumi[1], and Tan Ze Xin Dylan[1]

[1]Singapore University of Technology and Design, Singapore

https://github.com/jjbecomespheh/Trajectory_Prediction.git

Abstract. Having to predict accurate future trajectories for autonomous vehicles is an essential yet complex task. While there exists many approaches such as methods for multiple agent trajectory prediction, such methods require the modeling of both the a) **sequential aspect**, on how an agent's past state would influence its future states, and b) **social aspect**, on how agents in the environment would affect each other, introducing a high level of difficulty for students to understand the core concepts and establish an approach of their own. Since a vehicle's behavior on conventional roads is mostly linear, predictable and non-erratic, we use a foolproof and straightforward approach by ignoring the social aspect and simply modelling the sequential aspect. We demonstrate our approach on challenging, real-world trajectory datasets and show that our model is still able to provide a reasonable level of accuracy.

Keywords: Long Short-Term Memory · Cost Model · Selectivity Estimation · Bayesian networks.

1 Introduction

Object trajectory prediction is a tricky task that has gained traction in recent years because many modern applications are becoming increasingly relevant. These applications include but are not limited to autonomous robot navigation and self-driving vehicles.[4]

In this project, we are motivated by self-driving vehicles operating in a dynamic and uncertain environment. To be more specific, we focused on the urban traffic scene where the road is shared by a set of distinct agents like cars, pedestrians and cyclists. Self-driving vehicles require accurate and precise future trajectories prediction of the surrounding agents, where a trajectory is referred to as a sequence of x-y points.[2] However, due to the stochastic nature of each agent's future behaviour, predicting future trajectories becomes a relatively complicated process.

To enable us to model a more realistic trajectory prediction, we trained and evaluated our approach using a well-established public autonomous driving dataset, the nuScenes dataset. The team has attempted various ways to improve the evaluation results of the model, such as experimenting on different model designs, trying different normalization techniques and increasing the modality

of the model by incorporating more input features. The evaluation results were then compared with those from the state-of-the-art prediction models, and have shown a reasonable evaluation score.

In this report, we present a framework for object trajectory prediction for self-driving vehicles dependent on deep learning. We first introduce the related work. We then introduce the problem and explain the main techniques used for data pre-processing. Next, we present our problem statement and the implementation details which consist of our model, training and evaluation methods. Finally, we reported our qualitative and quantitative analysis of the findings, the conclusion and future works.

2 Related work

2.1 CoverNet

One of the underlying approaches that we've referred to is the CoverNet model, which was included in the nuScenes Prediction Tutorial. This approach proposes a new method for multimodal, probabilistic trajectory prediction for urban driving by framing the trajectory prediction problem as a classification problem [3].

CoverNet's approach was on the basis that multimodal regression models may degenerate during training into a single model, known as "mode collapse", and may also result in trajectory predictions that are not physically possible to be executed. It reasons that the dynamic constraints would considerably limit the trajectories into only a few distinct and plausible actions over a sensible period of time, and therefore, using classification would avoid the issue of mode collapse while satisfying the problem's requirements.

It uses a deep learning approach by including Convolutional Neural Networks (CNN), providing scene context via a high-definition map. It then concatenates the image features obtained from the CNN with state inputs such as the speed, acceleration and yaw rate of agents, and parses it through a fully connected layer to obtain the mode probabilities. Together with the trajectory sets generated from a trajectory generator that uses the state inputs, CoverNet then classifies over the set of trajectories.

CoverNet uses ResNet-50 as the backbone, pretrained on ImageNet weights, and applies a global pooling layer on the conv5 feature map. It also uses cross-entropy as the loss function with positive samples determined by the element in the trajectory set closest to the actual ground truth in minimum average of pointwise Euclidean distances.

Despite having the CoverNet model provided, no training code was given and with such a complex approach, the team was initially unsure of how we could train and modify the model. The team also initially took inspiration from the

classification approach and wanted to re-scope the problem into a classification task by classifying the drivers' intentions such as changing lanes, making a left turn etc. However, upon further analysis of the nuScenes Dataset, the scenes were found to be unsuitable for this approach and therefore the team sought to explore other methodologies to approach the problem.

2.2 AgentFormer

One of the approaches that the team sought inspiration and took reference from, was AgentFormer [4]. In the AgentFormer approach, the researchers sought to predict the future trajectories of multiple agents and modelled both the time and social dimensions simultaneously.

Their motivation was that in conventional methods, the time and social dimensions were usually modelled separately, where a temporal model is first used to summarize the features over time for each agent independently, then the modelling of social interaction is done based on the summarized features. This resulted in an issue where the independent feature encoding over either the time or social dimensions would result in the loss of information and thus, a decrease in the accuracy of the predictions.

The researchers innovated a new transformer model, called AgentFormer, which consisted of encoders and decoders, together with their own attention mechanism. They replaced the conventional positional encoding in original transformers with a time encoder, and created a novel agent-aware attention mechanism, different from the usual scaled dot-product attention. The time encoder used informs AgentFormer about the timestep that is associated with each element in the trajectory sequence while the Agent-aware attention preserves the agent information in the trajectory sequence. This therefore resolves the issues regarding the loss of time information as well as loss of agent information.

The research paper of AgentFormer has also concisely stated the inputs and outputs of their model in mathematical notation, which provided the team insights on what the inputs and outputs of our own model could be. Their model's input was the joint state of all agents' history, represented by $\mathbf{X}$, which includes the position, velocity and (optional) heading angle of the agent. On the other hand, their model's output was the joint state of all agents at a future time, denoted by $\mathbf{Y}$. Given past trajectories $\mathbf{X}$ and contextual information $\mathbf{I}$, their objective was to learn a generative model $P_\theta(Y \mid X, I)$, producing probabilities of future trajectories $\mathbf{Y}$, over a set of model parameters, θ.

Through this paper on AgentFormer, the team has learnt more about how to approach multi agent trajectory prediction, where there is a need to model both time and social dimensions. The team also had a clearer understanding about the possible inputs and outputs of a model relating to trajectory prediction, and learnt about the complexity of trajectory prediction. This led us to scope our

project with a more achievable goal, by only focusing on the modelling of the sequential aspect of agents.

3 Dataset and Collection

3.1 Dataset Description

The nuScenes dataset used in this project is a public large-scale dataset for autonomous driving created by Motional (previously known as nuTonomy), which is an Autonomous Vehicle Joint Venture between Aptiv and Hyundai Motor Group. In the dataset, there are 1000 driving scenes collected from locations including Boston Seaport and Singapore's One North, Queenstown and Holland Village. The dataset includes a full sensor suite of data on the Autonomous Vehicle, with scenes that are of 20 seconds in length, manually selected to display a diverse and exciting set of driving maneuvers, traffic situations and unexpected behaviors.

Some of the features of the NuScenes dataset are[2]:

- Full sensor suite (1x LIDAR, 5x RADAR, 6x camera, IMU, GPS)
- 1000 scenes of 20s each
- 1,400,000 camera images
- 390,000 lidar sweeps
- Two diverse cities: Boston and Singapore
- Left versus right hand traffic
- Detailed map information
- 1.4M 3D bounding boxes manually annotated for 23 object classes
- Attributes such as visibility, activity and pose

3.2 Dataset Utilization

The NuScenes dataset follows a database format where all the data are stored individually in JSON files, each with token numbers tagged to them. This is represented clearly through the nuScenes schema, which shows all the available JSON files and the data it contains, as well as their relation to other JSON files, which will be important to determine which nuScenes API to use.

3.3 NuScenes Dataset Extraction

To be able to extract data such as the vehicle translation and rotation, we would have to use the nuScenes API to access the data through its token number. However to do so, we have to first install the DevKit and set it up with the dataset. Following this, we are able to do things like:

1. Retrieving the list of agents shown in figure 1

```
['bc38961ca0ac4b14ab90e547ba79fbb6_39586f9d59004284a7114a68825e8eec',
 'bc38961ca0ac4b14ab90e547ba79fbb6_356d81f38dd9473ba590f39e266f54e5',
 'bc38961ca0ac4b14ab90e547ba79fbb6_e0845f5322254dafadbbed75aaa07969',
 'bc38961ca0ac4b14ab90e547ba79fbb6_c923fe08b2ff4e27975d2bf30934383b',
 'bc38961ca0ac4b14ab90e547ba79fbb6_f1e3d9d08f044c439ce86a2d6fcca57b',
 'bc38961ca0ac4b14ab90e547ba79fbb6_4f545737bf3347fbbc9af60b0be9a963',
 'bc38961ca0ac4b14ab90e547ba79fbb6_7626dde27d604ac28a0240bdd54eba7a',
 'bc38961ca0ac4b14ab90e547ba79fbb6_be99ffc878b24aca8956bbb4e0f97d0c',
 'bc38961ca0ac4b14ab90e547ba79fbb6_9813c23a5f1448b09bb7910fea9baf20',
 'bc38961ca0ac4b14ab90e547ba79fbb6_023c4df2d451409881d8e6ea82f14704',
 'a60047adc78a4b6895702e86b6d2fe88_39586f9d59004284a7114a68825e8eec',
 'a60047adc78a4b6895702e86b6d2fe88_356d81f38dd9473ba590f39e266f54e5']
```

Fig. 1: List of Agents

2. Retrieving sample annotation shown in figure 2

```
{'attribute_tokens': ['4d8821270b4a47e3a8a300cbec48188e'],
 'category_name': 'human.pedestrian.adult',
 'instance_token': '6dd2cbf4c24b4caeb625035869bca7b5',
 'next': '7987617983634b119e383d8a29607fd7',
 'num_lidar_pts': 1,
 'num_radar_pts': 0,
 'prev': '',
 'rotation': [0.9831098797903927, 0.0, 0.0, -0.18301629506281616],
 'sample_token': 'ca9a282c9e77460f8360f564131a8af5',
 'size': [0.621, 0.669, 1.642],
 'token': 'ef63a697930c4b20a6b9791f423351da',
 'translation': [373.256, 1130.419, 0.8],
 'visibility_token': '1'}
```

Fig. 2: Sample Annotation

3. Retrieving future coordinates of an agent shown in figure 3

```
array([[ 392.836, 1148.208],
       [ 392.641, 1147.242],
       [ 392.081, 1145.988],
       [ 391.347, 1144.208],
       [ 390.512, 1142.201],
       [ 389.29 , 1139.46 ]])
```

Fig. 3: Sample Future Coordinates

4. Retrieving past coordinates of an agent shown in figure 4
5. Velocity, acceleration and heading change rate shown in figure 5

```
array([[ 393.357, 1149.173]])
```

Fig. 4: Sample Past Coordinates

```
Velocity: 4.385040264738063

Acceleration: 0.30576530453207523

Heading Change Rate: 0.0
```

Fig. 5: Sample Velocity, Acceleration, Heading Change Rate

6. Render Ego Poses on Map shown in figure 6

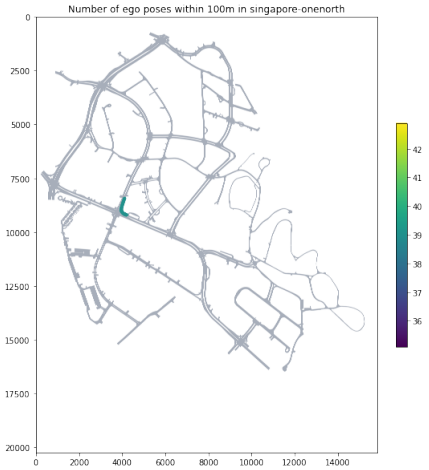

Fig. 6: Ego Poses on Map

7. Lidar View shown in figure 7

4 Data Pre-processing

As the nuScenes dataset is highly complex with large amounts of information provided, there is a necessity to pre-process the data to filter and extract data that would be relevant to our model. The team first adapted AgentFormer's script for the pre-processing of data, as processing the large chunks of data which are organized in a complex manner, would be too time consuming and impede our progress.

To be able to use AgentFormer's processing script. We had to first download the nuScenes dataset, and understand how to structure it. The full original dataset was approximately 300GB large, and as such, due to the lack of storage space, we had to first figure out if it was possible to only download a subset of the full original dataset, such that the AgentFormer's processing script would still work.

Upon further analysis and research on the AgentFormer's processing script, we found that it is indeed possible to just download all the annotations from

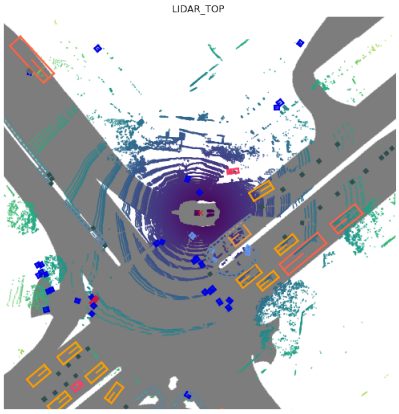

Fig. 7: Lidar View

the full dataset, which was split into 10 parts and about 60GB in total. After downloading the annotations and other required folders such as to metadata and map expansion set, we had to merge them while being mindful not to override each other, since the 10 annotation folders had the same folder names. To do so, we wrote a custom script and the resulting structure was as such:

After structuring the dataset, we were then able to run their process script, giving us a bunch of .txt files of the individual scenes, consisting of useful information such as the frame ids, instance ids and translation of the agents. The .txt files of individual scenes were also already split up into train, val and test folders. Based on these pre-processed data, we then further processed it to extract only the information that is required for our model. To do so, the team created another custom processing script, which allowed us to extract the frame id, instance id, x, y and z translation and heading angle of the agent, and export these data into a .csv format for the training, validation and test sets.

As there is an issue of repeating instance ids, where each individual scene had instance numbering starting from 0, we concatenated the scene number in front of the instance id. This is also beneficial as this straightforward approach enabled us to find out the scene number from the instance id.

Lastly, our custom processing file also allows us to visualize the trajectories in each individual scene, giving us useful insights on the expected trajectories for each agent in a scene.

5 Problem

Predicting accurate future trajectories of multiple agents is essential for autonomous vehicles (AV) so that the AV can anticipate the dynamic agents in the surrounding and prevent collision with them. It is challenging due to the complex interaction between agents and the uncertainty in each agent's future

behavior. In a static environment, the autonomous vehicles can drive along the planned path safely. However, the road traffic in reality is complex and dynamic, especially in urban cities.

In order to navigate safely, human drivers use long term memory and intuition of past driving experiences, real time sensory perceptions of dynamic agents and also short term memory of historical information like position, relative speed of objects around them. These abilities allow the human drivers to efficiently predict the future trajectories of moving agents in the surrounding area in real time. Taking inspiration from the human drivers, the autonomous vehicles can also predict the future trajectories of dynamic agents using machine learning techniques that are analogically similar to the human driver.

This project aims to utilise techniques and knowledge learned in the 50.038 Computational Data Science course to predict future trajectories of agents in the nuScenes dataset.

6 Algorithm

The Long Short Term Memory (LSTM) is normally used to solve many useful tasks like speech recognition, translating languages, stock predictions, synthesize music and control robots. All these tasks have one central similarity, which is the data that is used are in time series. Our project uses a time series dataset which contains information like trajectory coordinates and sensor data like acceleration and heading angles of agents. Hence, LSTM would be a suitable model.

At the initial phase of the project, the team took reference to existing works on trajectory prediction. There are several notable State of the Art (SOTA) solutions like the CoverNet[3], AgentFormer [4] and Trajectron++ [**?**] that the team have researched on. In our attempt to understand the academic papers and the respective source code, the team realised that the complexity of the papers and source code is too advanced for us to manage. As the source code of the existing SOTA solutions are complex with many thousands lines of code, the team was unable to reference or extract useful insights to build our naive model solution for the trajectory prediction problem.

Throughout the project, we made different configurations of models by experimenting with different methods. Also, we used Mean Squared Error Loss as our loss function [1]. MSE loss is calculated as the average of the squared differences between the predicted and true values and is usually adopted in regression problems. Since the problem we are trying to tackle is also a regression problem, MSE would be a good fit. Another reason why we used MSE is because it is sensitive to outliers. Larger mistakes would result in a greater effect as compared to the smaller mistakes, hence the model would be penalized more significantly

when faced with a large mistake.

The Adam Optimizer is used in this project, which is also known as Adaptive Moment Estimation Algorithm. The Adam Optimizer estimates moments and uses them to optimize a function, it is a combination of the gradient descent with momentum algorithm and the Root Mean Square algorithm. The reason that Adam Optimizer was used is because it is easy to implement, computationally efficient and works well with large data sets.

The team also utilised other techniques that were learnt in the course, and one of the techniques is Early Stopping. A range of Patience values were experimented on, where a Patience value of 10 produced an overfit result, a Patience value of 2 produced an under fitted result, and a Patience value of 5 produced the most optimal result.

6.1 Approach A: Multi LSTM Model

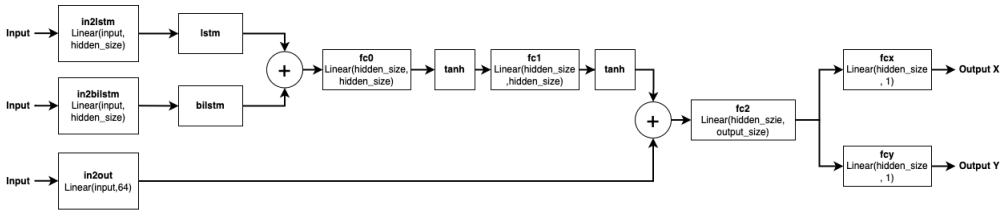

Fig. 8: Multi LSTM Model Architechture

Instead of complex SOTA solutions, the team turned to research on simpler trajectory prediction solutions implemented using LSTM. One solution that the team found has the following model design as shown in figure 8. The model design features one LSTM, one Bi-LSTM and also a Linear Layer taking the input and adding to the eventual output of the LSTMs as residual connection to preserve the input information. We implemented this model and trained it using the nuScenes' trajectory data.

6.2 Approach B: Multi-Task LSTM Model

In the initial model design, the X and Y Coordinates output were passed through a single Fully Connected Layer to produce a 2D Array Output with the X and Y Coordinates. However after consultation with our professor, we were advised to modify our model to use 2 independent fully connected layers to output the corresponding X and Y coordinates prediction instead. Through this method, the new model design incorporates a multi task learning approach of the X and Y Coordinates data at the same time, allowing the model to adjust the hyperparameters relating to X and Y separately.

As the loss was high ranging from over 500,00 to over 2,000,000, the team attempted to tune the hyperparameters via the addition of hidden layers. However, it was shown to have no improvements and instead made the loss, ADE and

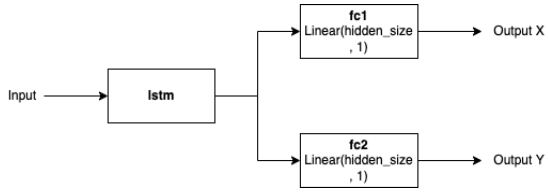

Fig. 9: Multi-Task LSTM Model 1 Architechture

FDE worse. Therefore, reverted to the previous model by removing the series of Hidden Linear Layers.

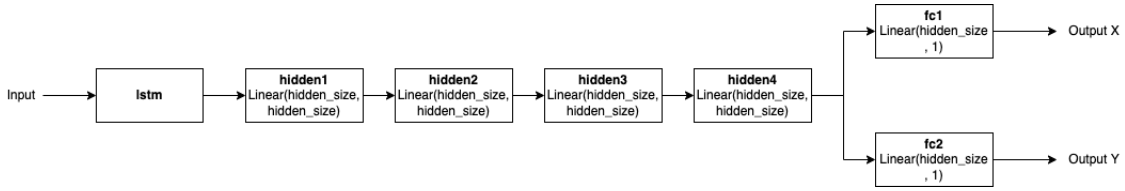

Fig. 10: Multi-Task LSTM Model 2 Architechture

6.3 Approach C: Normalization Experimentation

As the loss was very high, the team implemented Normalization, in order to improve the model's performance. Without normalization, the loss value could range from around 1,000 to 1,000,000, and did not converge even after 30 epochs. This also resulted in poor evaluation results.

The team first implemented a MinMaxScaler which normalised the Trajectory values between 0 to 1 across all the training and validation set values. However, although this resolved the problem of the high loss, which dropped to being lower than a value of 2 consistently, the evaluation results were still not optimal.

Instead of using one MinMaxScaler for the whole dataset, we then tried using a MinMaxScaler for each individual instance by initialising a new MinMaxScaler every time. Surprisingly, this resulted in a significant improvement to the evaluation performance. Lastly, the team experimented with normalising the Trajectory values between -1 to 1, and this method resulted in the best evaluation performance.

6.4 Approach D: Look-Ahead Prediction

The team first implemented the One Step Prediction which uses the past trajectory. For example using a sequence of 4 time steps with X Y Coordinates to predict the next frame of X and Y Coordinate. This implementation produced very impressive results with ADE ¡ 1. However, the team realised that this method is not the application that we require for the project as it only

predicts the next time step of X Y coordinates where it always references to the ground truth as it moves along the input trajectory.

In order to implement a method that is suited for the project's application, which is to predict several time steps ahead of a given trajectory, the team changed to the Look Ahead Prediction instead of the One Step Prediction. The Look Ahead Prediction predicts a longer time frame and produces a result that fulfils the scope of the project. Although the results are not as accurate as that of the One Step Prediction method, the prediction is more useful for the project.

6.5 Approach E: Sequence Length Experimentation

Other changes that were experimented on was the Sequence Length, which is the series of time steps of X Y coordinates used as the inputs for the model. A longer sequence length would mean that more past data has been used to predict the future data, while a shorter sequence length would mean that lesser past data is used to predict the future data. Most of the experiments were experimented with sequence length of 4, and thus the team experimented with sequence lengths of 6 and 8, and observed that the best performance was with a sequence length of 6.

6.6 Approach F: Prediction Length Experimentation

The team also experimented with the prediction length, which is the amount of frames to predict the trajectory. The original Prediction task is to predict a trajectory that is 6 second long into the future, each time step in the nuScenes dataset is 0.5 seconds so 6 seconds equates to 12 frames. For our model, it is too challenging to predict 12 frames into the future as our model is not complex enough, as small errors can propagate into very large errors and the trajectory can be very skewed and does not resemble the actual trajectory. After our experimentation, using a Prediction Length of 6 frames produces reasonable performance.

6.7 Approach G: Increased Modality

To increase the accuracy and robustness of the model, the team added more input features like z translation and heading angle of the agent. Since the preprocessed dataset did not include the velocity and acceleration of the vehicle, the team was unable to include it into the training.

7 Evaluation Methodology

Initially the team uses visual assessment of the ground truth trajectory and the predicted trajectory for one particular scene. This method although is fast and

intuitive to humans, the visual assessment is inaccurate and does not quantitatively show the difference in performance of different models.

Instead of using human visual assessment, the team uses the Standard Metric for Trajectory Prediction in the industry which are the minimum Average Displacement Error (min ADE) and the Average Final Displacement Error (FDE). The Average Displacement Error (ADE) is the average L2 distance between ground truth and our prediction over all predicted time steps. The Final Displacement Error (FDE) is the distance between the predicted final destination and the true final destination at the end of the prediction period Tpred.

$$ADE_k = \frac{1}{T} \min_{k=1}^{K} \sum_{t=1}^{T} \|\hat{y}_n^{t,(k)} - y_n^t\|^2$$
$$FDE_k = \min_{k=1}^{K} \|\hat{y}_n^{T,(k)} - y_n^T\|^2$$

8 Results and Discussion

The best result that our model has achieved is 2.923 for ADE-1, where normalization is used between -1 to 1, and MinMaxScaler is initialized for every single agent. This number is better than the existing works like MTP and CoverNet, but the team do not believe that our current model will perform better than the State of The Art solutions like CoverNet. A possible reason for our results being better, may be due to the evaluation method being different compared to the existing solutions. More work is required to be done to ensure that all the results that we are comparing, are evaluated on the same fair scale. Nonetheless, our results show a satisfactory performance for the scope of our project.

In our initial visualization, we plotted the predictions on a simple line graph with legends to indicate the expected and predicted trajectories, but the plot was not a good visualization as it did not indicate the direction of the moving agent, making our inference vague. In order to mitigate the problem of the unknown direction, we saved a photo of each prediction for a scene and stitched them together to produce a gif. By doing so, we are able to visualize the moving vehicle on the graph, providing us with a clearer understanding of how our model performs.

Based on our testing, we have observed that our model works well for agents that are travelling in a linear or almost linear fashion. A probable cause for this may be that most of the trajectories in the training and validation set are mostly linear, and hence our model has learned to travel well in a linear fashion. We also noticed that our model performs better when the distance of the trajectory points from each other are around the same throughout the whole trajectory. The implication of this is that our model may not perform well for trajectories where the vehicle has acceleration, deceleration or change in velocity, and therefore, including such information as the input features may improve the model.

Approach	One Step	Look Ahead
Multi Task LSTM without Normalization	ADE: 14.233 FDE: 14.233	ADE: 698.29 FDE: 698.29
Normalization across whole training and validation dataset between -1 and 1	ADE: 5.332 FDE: 5.332	ADE: 79.999 FDE: 79.999
Normalization initialized on every instance between 0 and 1	ADE: 0.9839 FDE: 0.9839	ADE: 4.9409 FDE: 4.9409
Sequence length 4 with normalization initialized on every instance between -1 and 1	ADE: 0.7373 FDE: 0.7373	ADE: 3.1019 FDE: 3.1019
Sequence length 6 with normalization initialized on every instance between -1 and 1	**ADE: 0.6629** **FDE: 0.6629**	**ADE: 2.923** **FDE: 2.923**
Sequence length 8 with normalization initialized on every instance between -1 and 1	ADE: 0.829 FDE: 0.829	ADE: 4.176 FDE: 4.176
Increased modality by including the z translation and heading angle	ADE: 1.3113 FDE: 1.3113	ADE: 5.7129 FDE: 5.7129
Existing Solutions	**Results**	
MTP	ADE: 4.42 , FDE: 10.36	
CoverNet	ADE: 3.87 , FDE: 9.26	

Fig. 11

9 Future Work

Based on the evaluation results, we have seen that our model is able to provide a reasonable degree of accuracy for various scenes in the dataset. However, it has been observed that our model works best for trajectories that are linear or similar, and is not yet robust enough to be able to accurately predict the trajectories for scenes that incorporate turns. Therefore, listed below are a few proposed ideas that the team feels would help improve the reliability and robustness of the model:

9.1 VAE-LSTM

LSTM is a type of recurrent neural network that does not take into account the uncertainty of a prediction. This would pose a great limitation for prediction tasks as it could lead to over-confident predictions especially since each traffic scene is quite different from one another. To address this problem of uncertainty, Variational AutoEncoders (VAE) can be implemented together with LSTM. VAE is a type of generative graphical model which consists of an encoder and decoder, and learns the data distribution from the training samples, allowing it to generate new and reasonable predictions from the distribution. Through the addition of VAE, the model would thus be more robust.

9.2 Social-Modelling

Although the team's objective was to provide a simple approach for trajectory prediction, given that we have only modelled the sequential aspect of an agent, the model is not as robust since it does not take into account the influence of the environmental factors. Therefore, by increasing the complexity of the approach through taking the context into consideration, such as the traffic road layout

and how other agents would affect the trajectory of the target agent, it would definitely improve the robustness and accuracy of the model.

9.3 Increasing the Modality

Although the team has already experimented with the increase of the number of input features, such as including the z translation and heading angle of the vehicle, it was observed that it did not improve the results of the model. However, this may be due to the z-translation not affecting the x-y trajectory of the vehicle much and perhaps including more relevant features such as the velocity and acceleration of the vehicle would further improve the model.

9.4 Improved Visualization

The current visualization that we have used does not include the context of the traffic scene, where agents surrounding the target agent as well as the traffic road layout cannot be seen. Therefore, by translating the scene images into a graphical format and plotting the predictions on the graph, this would provide a more intuitive interpretation of the trajectory prediction.

10 Conclusion

The trajectory prediction problem is complex in nature as there are many dynamic agents in the environment and the interactions between agents are dynamic and complex. In order to enable a safe autonomous vehicle, it is essential for the autonomous vehicle to predict the trajectory of the agents in its environment accurately and efficiently.

The team started with learning from the State of The Art existing works on trajectory prediction like the CoverNet, Agentformer etc, and while researching on the existing solutions, the team learnt more about the different approaches to solve the problem and the complexity of the trajectory prediction problem.

The nuScenes dataset was used in this project, which is a public large-scale dataset for autonomous driving research created by Motional. Among the many features in the nuScenes dataset, this project focused on using the trajectory data which contained time series X and Y Coordinates, acceleration and headings of agents from the 1000 scenes of 20 seconds long. Data pre-processing was done to extract relevant trajectory data like the instance, scene identification number, time series X and Y coordinates from the nuScenes dataset.

Many Machine Learning techniques and skills learned from the course of 50.038 Computational Data Science were used in this project, for example LSTM, Early Stopping, Pytorch, Hyperparameter Tuning and useful Visualization Techniques. These techniques have allowed the team to experiment with different

approaches and produce the iterative improvement of the model in this project. The team was able to achieve satisfactory results using the simple model that we implemented from scratch. The 50.038 project has been an enriching and fulfilling learning experience for the team.

References

1. Brownlee, J.: How to choose loss functions when training deep learning neural networks (2019), https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/
2. nuScenes, M.: Nuscenes (2020), https://www.nuscenes.org/
3. Tung Phan-Minh, Elena Corina Grigore, F.A.B.O.B., Eric M. Wolff nuTonomy, a.A.c.: Covernet: Multimodal behavior prediction using trajectory sets (2020)
4. Ye Yuan, Xinshuo Weng, Y.O.K.K.: Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting (2021)