Trajectory Prediction: A Project

Northwestern University, Evanston, IL, USA
Yimin Han ; Partner:Xiaoying Xing

Abstract

Predicting accurate future trajectories of multiple vehicles is essential for autonomous systems but is challenging due to the complex interaction between vehicles and the uncertainty in each vehicle's future behavior. Multiple interacting vehicles, the multi-modal nature of driver behavior, and the inherent uncertainty involved in the task make motion prediction of surrounding vehicles a challenge. This challenge problem can be divided into two aspects: Time Dimension and Social Dimension. (1)time dimension, where we model the influence of past vehicle states over future states; (2)social dimension, where we model how the state of each vehicle affects others. In this report, we propose an LSTM encoder-decoder model that assigns confidence values to maneuvers being performed by vehicles. Our model outputs a multi-modal predictive distribution over future trajectories based on maneuver classes. We evaluate our model using the publicly available NGSIM US-101 and I-80 datasets and compare our approach with baseline model. Using metrics: Root of the mean squared error(RMSE) of the predicted trajectories and Negative log-likelihood(NLL) of the predictive distributions, our model is capable of generating satisfying results. Our model performs as well as the baseline model with higher efficiency. We also visualize the predictions made by the model in complex traffic scenarios.

Keywords: Multi-Vehicle; Trajectory Prediction; LSTM Encoder-Decoder; Maneuver Classes

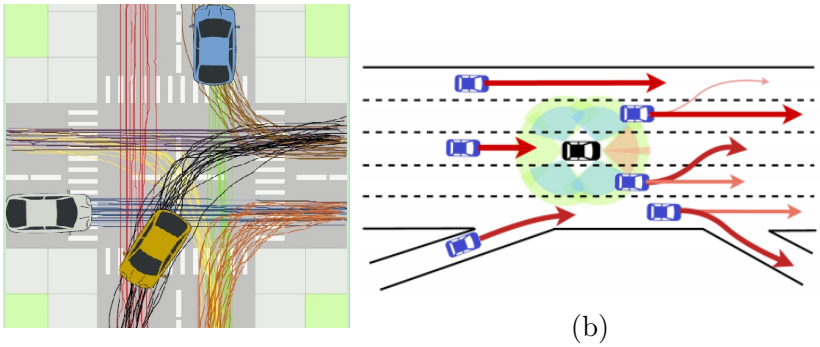
1 Background and Introduction

Our focus is the predicting future vehicles states, which is a crucial task for robot planning in real-world environments. Importantly, predicting the future of other vehicles in autonomous driving domain is vital for safe, comfortable and efficient operation. This requires the autonomous vehicle to have some ability to reason about the future motion. Accomplishing this, safely and efficiently navigate through complex traffic comprised by human drivers can be achieved. An autonomous vehicle may have the ability to take initiative, such as deciding when to change lanes, overtake another vehicle, or slowing down to allow other vehicles to merge. This can be seen in existing tactical path planning algorithms [1] [2] [3], which depend on reliable estimation of future trajectories of surrounding vehicles.

To achieve safe planning of autonomous driving vehicles, two aspects are required: forecasting accurate future trajectories of ego vehicle and surrounding vehicles. However, multi-agent trajectory forecasting is challenging since the human drivers are highly multimodal, dynamic, and variable. Here, "multimodal" refers to the possibility of many highly-distinct future behaviors; "dynamic" refers to the ability of vehicles to change directions or velocities; and "variable" refers to the fact that in any situation there can be a different number of vehicles, meaning any multi-agent model needs to be able to handle the inputs including the information of many vehicles. Besides, the behavioral influence of an agent on others, is a complex process. Therefore, let us separate the problem into two aspects: (1) time dimension, where we model how past vehicles states (positions and velocities) influence future vehicles states; (2) social dimension, where we model how each vehicles' state affects the state other vehicles.

For the time dimension, intuitively speaking, it is to infer the future trajectory based on the history information. This is a temporal problem. Approaches like [4] [5] [6]

first use temporal models (e.g., LSTM [7] or Transformers [8]) to summarize trajectory features over time for each vehicle. These approaches formulate trajectory prediction as a regression problem by minimizing the error between predicted and true trajectories in a training dataset. A pitfall for regression based approaches is the inherent multi-modality of driver behavior. A human driver can make one of many decisions under the same traffic circumstances. For example, a driver approaching their leading vehicle at a faster speed could either slow down, or change lane and accelerate to overtake. Regression based approaches tend to output the average of these multiple possibilities. However, the average value may not be the best prediction. Mean value cannot address the multi-modal nature of predictions. As shown in Fig.1(a)(b), vehicles have many choices when deciding their future positions



(a) (b)

Figure 1

For the social dimension, methods for social interaction modeling can be categorized based on how they model the time and social dimensions. Graph neural networks (GNNs) [9] [10] are often employed as the social models for interaction modeling. Approaches like [11] [12] first use social models to produce social features of each vehicle at each independent timestep and then apply temporal models to summarize the social features over time for each vehicle. The common characteristic of these prior works is that they model the time and social dimensions separately. This is not very optimal because it prevents an vehicle's feature at one time from directly

interacting with another vehicle's feature at a different time, leading to the limitation on the model's ability to capture long-range dependencies.

Considering the problems of these two dimensions, we should consider more than average value of future trajectories and consider the social interactions. So it is important to learn drivers driving pattern, which means how they drive. Our target is to find out the future position distribution. To accomplish this, we present a framework for modeling multi-modal, dynamic, and variable multi-vehicle scenarios. It efficiently models the multi-modal prediction. Following the success of Long-Short Term Memory (LSTM) networks in modeling non-linear temporal dependencies in sequence learning and generation tasks [5] [13] [14], we build up an LSTM Encoder-Decoder. Moreover, we use maneuvers for multi-modal trajectory prediction, by learning a model that assigns probabilities for different maneuver classes, and outputs maneuver specific predictions for each maneuver class. Our project is inspired by the work of Dr.Nachiket Deo and Prof.Mohan M. Trivedi. We use their model as the baseline model [15] and make some innovations. Surprisingly, we get amazing results.

2 Challenges and Related Work

In this section, we will discuss the most relevant parts of our approach and what have been done in these areas. Then, I will talk about which related work inspire us and what idea we get from that work. In this way, the design and development of our work can be understood easily.

2.1 Maneuver Classification

To improve the prediction accuracy of the model, making maneuver based prediction is a good choice. This design make classification of vehicle motion into semantically

interpretable maneuver classes. This method has been extensively addressed in both advanced driver assistance systems as well as naturalistic drive studies. Here is some work that use the recognized maneuvers to make better predictions of future trajectories. [16] [17] [18] These approaches usually involve a maneuver recognition module for classifying maneuvers and maneuver specific trajectory prediction modules. Maneuver recognition modules are typically classifiers that use past positions and motion states of the vehicles and context cues as features. And its output is the future locations of the vehicle given its maneuver class. Heuristic based classifiers [16], Bayesian networks [2], maneuver specific motion models [2], Gaussian processes [17], and hidden Markov model [19] have been widely used. It is widely acknowledged that using maneuver based predicting method can improve the model performance.

The hidden Markov model inspire us the most. [19] Hidden Markov models (HMMs) have previously been used for maneuver recognition due to their ability to capture the spatial and temporal variability of trajectories. HMMs can be thought of as combining two stochastic models, an underlying Markov chain of states characterized by state transition probabilities and an emission probability distribution over the feature space for each state. The transition probabilities model the temporal variability of trajectories while the emission probabilities model the spatial variability, making HMMs a viable approach for maneuver recognition.

The process is that for each maneuver, a separate HMM which is with a left-right topology with only self transitions and transitions to the next state, will be trained. The x and y ground plane co-ordinates and instantaneous velocities in the x and y direction are used as features for training the HMMs. For a car i, the HMM for maneuver k, the log likelihood output is:

$$L_k^i = log(P(\mathbf{x}_h^i, \mathbf{y}_h^i, \mathbf{v}_x^i h, \mathbf{x}_y^i h | m^i = k; \theta_k)) \tag{1}$$

The state emission probabilities are modeled as mixtures of Gaussians with diagonal covariances. And the parameter θ of the HMMs (including the state transition probabilities and the means, variances and weights of the mixture components) are estimated by Baum-Welch algorithm. [20] This method give us a hint on we can predict the future position distributions of vehicles using predefined maneuver classes in supervised learning.

2.2 Social Interaction

As discussed above, the social dimension is a very important point to be considered about in this topic. We want to make our model more compact and let our model be able to learn the dependencies between a vehicle at one time and another vehicle at a different time. So we seek for a method that can model both the time and social dimensions simultaneously, allowing direct feature interaction across time and agents.

There are some works that can achieve this goal, using Transformer [21], Conditional Variational Autoencoders(CVAEs) [22] and so on. The most common characteristic of these methods, are their way to organize the input data.

$$Input : \mathbf{X}_{1,\dots,N}^{t+1:t+T} \tag{2}$$

As we can see, the history trajectory of different vehicles are concatenated to form the input. Since this is a supervised learning task, the ground truth are formed in the same way.

$$Output : \mathbf{Y}_{1,\dots,N}^{t+1:t+T} \tag{3}$$

With reference to this idea, we designed our own input and ground truth.

3 Problem Formulation

Our Task: This is a supervised learning problem. Given data set $\mathbf{X} = [\mathbf{x}^{(t-T)}, ..., \mathbf{x}^{(t)}]$ and $\mathbf{Y} = [\mathbf{y}^{(t+1)}, ..., \mathbf{y}^{(t+T_f)}]$, We want to find out the future position distribution. History trajectory X is the input data, and future trajectory Y is the ground truth. Considering how to model the distribution, a bivariate Gaussian distribution is appropriate because X and Y are two orthogonal directions.

$$f(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x-\mu_X}{\sigma_X}\right)^2 - 2\rho\left(\frac{x-\mu_X}{\sigma_X}\right)\left(\frac{y-\mu_Y}{\sigma_Y}\right) + \left(\frac{y-\mu_Y}{\sigma_Y}\right)^2\right]\right)$$

In this formula, θ represent the point in the parameter space. It has five dimensions. $\theta = (\sigma_x, \sigma + y_{,x}\ _{,y}\ , \rho)$ These five dimensions represent the means for x and y, variance for x and y, and ρ. Our task is to decide the value of $\sigma_x, \sigma + y_{,x}\ _{,y}\ , \rho$ so that we can decide the parameter θ and decided the Gaussian Distribution.

Methodology: The learning process can be summarized as a Maximum Likelihood process:

$$\theta = argmax_\theta P(Y|X) \tag{4}$$

This equation means choosing the theta that can maximize the joint distribution of P(Y|X). Then, apply negative log to 6, we get:

$$\theta = argmin_\theta[-log(P(Y|X))] \tag{5}$$

Based on the structure of multi-lane freeways, 6 maneuver class are adopted to help the prediction process. The process is: Firstly, estimate m_i based on $\mathbf{X}$ and get $P(m_i|X)$. Secondly, based on m_i and $\mathbf{X}$, estimate $\mathbf{Y}$, and then get the maneuver dependent future distributions $P(Y|m_i, X)$. Using the chain rule, the product of

$P(Y|m_i, X)$ and $P(m_i|X)$ is the joint distribution $P(Y|X)$:

$$\theta = argmin_\theta[-log[\sum_i P(Y|m_i, X)P(m_i|X)]] \tag{6}$$

So far, the learning process can be modeled. The next step is how to model the computation process of $P(Y|m_i, X)$ and $P(m_i|X)$. We choose Neural Network here.

4 Baseline Model

Reference to the work of Dr.Nachiket Deo and Prof.Mohan M. Trivedi [15], we design our prediction model. Before the introduction to our work, it is necessary to discuss their model. In this way, we can show what our innovations are.

4.1 Inputs and Outputs

The main components of this model is an LSTM Encoder-Decoder and an LSTM classifier. The input of the model is the tensor of track histories:

$$\mathbf{X} = [\mathbf{x}^{(t-t_h)}, ..., \mathbf{x}^{(t-1)}, \mathbf{x}^{(t)}] \tag{7}$$

where,

$$\mathbf{X}^{(t)} = [x_0^{(t)}, y_0^{(t)}, x_1^{(t)}, y_1^{(t)}, ..., x_6^{(t)}, y_6^{(t)}] \tag{8}$$

are the x and y co-ordinates at time t of the vehicle being predicted and six surrounding vehicles. The ground truth is:

$$\mathbf{Y} = [\mathbf{y}^{(t+1)}, ..., \mathbf{y}^{(t+t_f)}] \tag{9}$$

where,

$$\mathbf{y}^{(t)} = [x_0^{(t)}, y_0^{(t)}] \tag{10}$$

The figure below shows the The co-ordinate system used for trajectory prediction. The final output is the parameters of a probability distribution over future predictions.

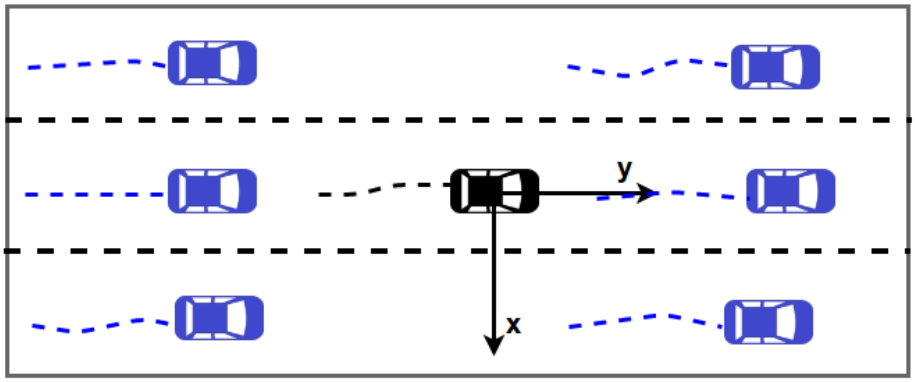


Figure 2: The vehicle being predicted is shown in black, neighboring vehicles considered are shown in blue.

4.2 Model Structure

Three lateral and two longitudinal maneuver classes are predefined, as shown in fig. 3.

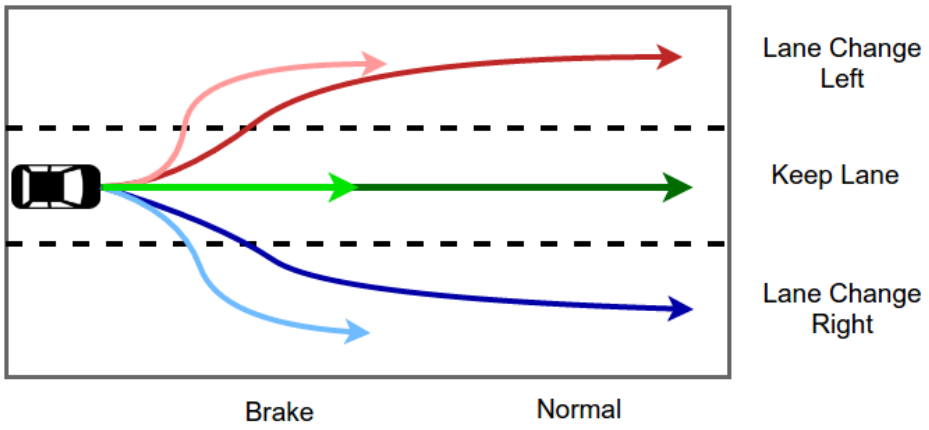


Figure 3: The lateral maneuvers consist of left and right lane changes and a lane keeping maneuver. The longitudinal maneuvers are split into normal driving and braking.

These six maneuvers help us in predicting. Our model will output a Maneuver dependent predictions. An encoder-decoder framework is used. The encoder LSTM takes the input for the past t_h frames frame by frame. The hidden state vector of the encoder is updated at each time step based on the hidden state at the previous time step and the input frame of vehicle locations at the current time step. The output of encoder, context vector is appended with maneuver encodings of the lateral and longitudinal maneuver classes. This context vector is then used by the decoder LSTM as input. At each time step, for t_f frames into the future, the decoder LSTM state is updated based on the encoded context vector and LSTM state at the previous instant. The decoder outputs at each time step, a 5-D vector $\theta(t)$ corresponding to the parameters of a bivariate Gaussian distribution. The trajectory encoder-decoder is trained using NLL loss. The LSTM classifier is a normal LSTM using the same input as the encoder-decoder LSTM. The classifier is trained to minimize the the sum of cross-entropy losses of the predicted and ground truth lateral and longitudinal maneuver classes and can output the lateral and longitudinal probability separately. The proposed model are shown in fig.4.

5 Our Model Design

5.1 Innovations

The baseline model uses three LSTM networks and separates the computation of $P(Y|m_i, X)$ and $P(m_i|X)$ into 2 processes. However, we think that $P(Y|m_i, X)$ and $P(m_i|X)$ share the same input information. They computation process can be merged. Therefore, we cancel the classifier LSTM and use the Encoder LSTM to accomplish the classifying task. This innovation significantly improves model's efficiency. Additionally, we also change some parameters of the model.

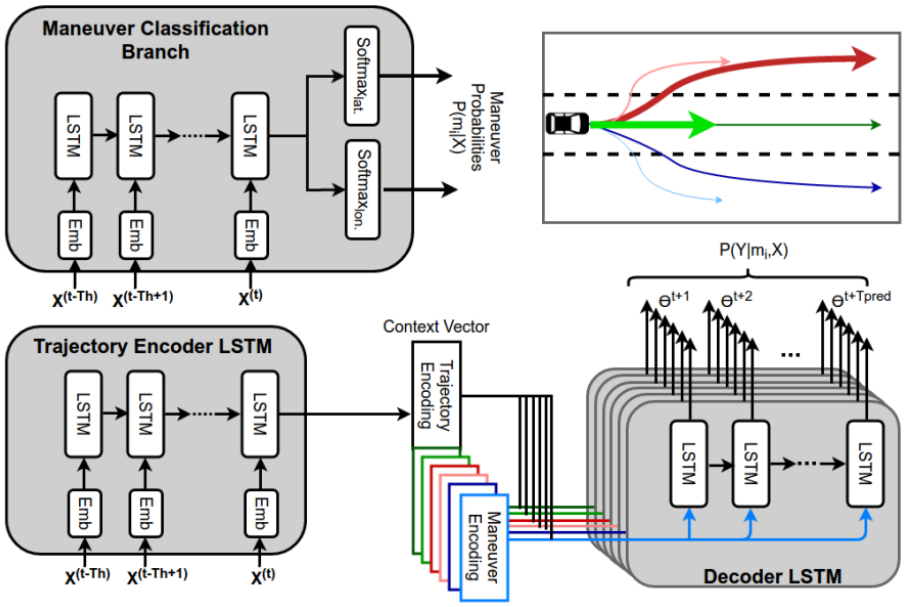


Figure 4: **Baseline Model:** Three LSTM networks are included in this figure. The encoder LSTM encodes the track histories and relative positions of the vehicle being predicted and its adjacent vehicles in a concatenated tensor which is the input. Concatenate the encoder output with Maneuver Encoding and send it to the Decoder. The decoder LSTM generates maneuver specific future distributions of vehicle positions at each time step. Besides, the LSTM based classification use the same input as the encoder-decoder and output the probabilities of maneuver classes.

5.2 Model Implementation details

Our model's input and output have the same structure as the baseline model. But we merge the classifier with the encoder. Instead of using input data to LSTM, we directly pass the context vector to Softmax layer to generate maneuver probability. Model structure is shown in fig.5

Embedded Layer: Before $\mathbf{X}$ sequence is inputted to the LSTM network, it should go through the embedded layer. The embedded using a 64-unit fully-connected layer. Since our input data is the x y co-ordinates of ego vehicle and six surrounding vehicles: $\mathbf{X}^{(t)} = [x_0^{(t)}, y_0^{(t)}, x_1^{(t)}, y_1^{(t)}, ..., x_6^{(t)}, y_6^{(t)}]$. So the input dimension is $D = 14$. After the embedded, it will be changed to a tensor with shape: $[batchsize, timelength, featuresdimension] = [batchsize, 16, 64]$. The activation function of Emb is Leaky Rectified Linear Unit

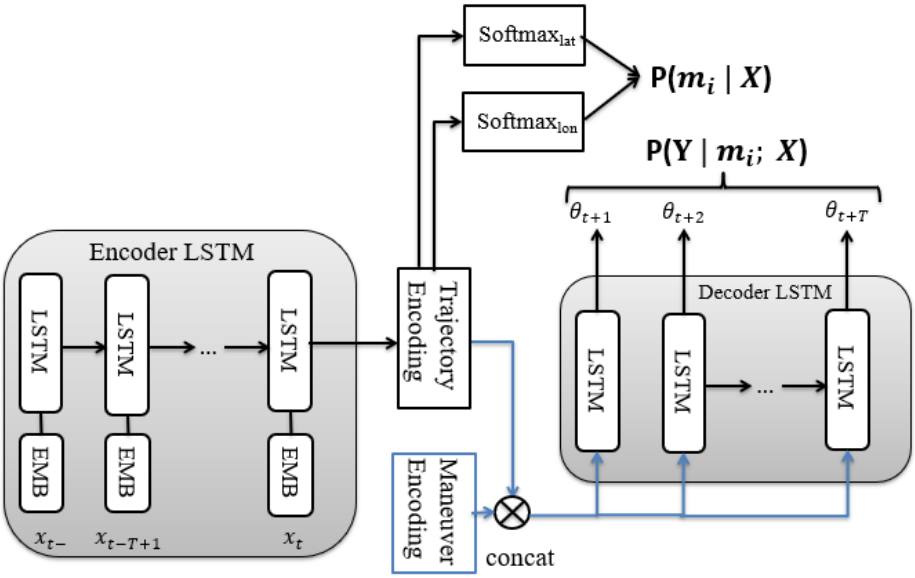


Figure 5: **New Model:** Input is the **X** sequence composed of ego cars and its surroundings. Emb is fully connected layers. Output of the encoder will pass 2 softmax layers and ouput 2 probabilities: lateral and longitudinal. Their product is the P(mi,x). Based on the output manuveur, we concat the one-hot encoding of m_i with encoder's output and pass it to decoder LSTM.

(leaky ReLU) with $\alpha = 0.1$. Leaky ReLU:

$$R(x) = \begin{cases} x & x > 0 \\ \alpha x & x <= 0 \end{cases} \tag{11}$$

LeakyRelu is a variant of ReLU. Instead of being 0 when z<0, a leaky ReLU allows a small, non-zero, constant gradient α. By having this small negative slope, leaky ReLU [23] is useful in fixing "dying ReLU" [24] problem: when most of the inputs to ReLU neurons are in the negative range, many ReLU neurons only output values of 0. This will cause the gradients fail to flow during backpropagation, and the weights are not updated. [25] To avoid this, we use leaky ReLU.

$P(Y|m_i; X)$**:** The output of encoder-decoder can represent the $P(Y|m_i; X)$. We use LSTM with 128 units for the encoder and decoder. Encoder output its final state, which is in the shape of [batch size, 1, 128]. 1 means only one layer of LSTM. Since

it is the fnial state, we expect that it have encoded the information about the track histories and relative positions of the total 7 vehicles. So we call this final state a context vector. We append the context vector with a one-hot vector corresponding to the lateral maneuver class and a one-hot vector corresponding to the longitudinal maneuver class. This concatenated tensor is in the shape of [batch size, 1, 134]. It is used as the input of the decoder LSTM. In this way, the decoder will generate the particular probability based on the specific maneuver class. Therefore, this output can represent the $P(Y|m_i; X)$. The loss of it is the negative log likelihood loss (NLL loss) for the ground truth future locations of vehicles under the predicted trajectory distribution.

$P(m_i|X)$**:** The context vector is also passed to the Softmax layer. We use two Softmax layer here to separately compute the lateral and longitudinal probability. Lateral means changing right, left or keep the lane. Longitudinal means keep velocity or brake. So the output are in the shape of [1, 3] and [1, 2], both of which are one-hot encoding.The product of this two output vector determines the maneuver class m_i.

$$softmax(x)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{12}$$

where K is the total number of classes, $K_lat = 3$, $K_lon = 2$. From Softmax form we know that, the element in output vector are probabilities. The biggest probability means the specific maneuver class this trajectory is. Then, we compute the sum of cross-entropy losses of the predicted and ground truth lateral and longitudinal maneuver classes. Sum these two cross-entropy with the NLL loss of $P(Y|m_i; X)$ and train to minimize it.

Our model are trained using Adam [26] with learning rate of 0.001. Batch size is set to be 128. The models are implemented using PyTorch. [27]

6 My work in This Project

Our work in this project can be divided into 5 parts: survey on topic, reproduce the baseline model, build a new model, train the model, and prepare for the presentation.

First of all, I proposed this topic to my partner, Xiaoying and find some relevant paper in this field. I have read most of them, chose 5 of them which I think interesting, and sent them to Xiaoying. Then, we discussed over these five papers and choose the best suitable one as our baseline model [15]. Moreover, we together discussed the improvement ideas and finally determined the most suitable and practical one.

Secondly, due to the author of this paper did not provide his source code and data set, I take the responsibility on reproducing the original model and generate data set. The publicly available NGSIM US-101 [28] and I80 [28] datasets were used for our experiments. The raw data on the NGSIM website are in the form of .csv with many columns. Each column represent a feature. I chose the data column that we need and encapsulated it into the form of PyTorch Dataset which is convenient for training process. Moreover, I built up the baseline model using PyTorch referring to the description and parameters in the paper.

And then, I trained the baseline model as well as our own model on servers. During the training process, I spilt the original dataset into 3 subsets: TrainSet, TestSet and ValidationSet. TrainSet is used for computing train loss and to back propagate and renew the parameters. After certain epoch, I use ValidationSet to compute the test loss. When test loss has converged, the best model is saved. TestSet is used when evaluation and prediction. Moreover, I recorded the training time for each epoch, training and testing loss and prediction accuracy using RMSE and NLL. These results will be discussed in next section.

Several days before the presentation day, we made PPT together. I took in charge of three parts of PPT: Background Introduction, Related Work, and Description to Our Innovative Approach.

Experiments and Results will be presented in following pages in section7.

7 Experiments and Results

Data Prepare: Our experiments use NGSIM US-101 [28] and I80 [28] datasets. Each dataset consists of trajectories of real freeway traffic captured at 10Hz over a time span of 45 minutes. The datasets provides the co-ordinate of vehicles projected to a local co-ordinate system. We spilt the trajectories into segments of 8s, where we use 3s of track history and a 5s prediction horizon. Within the 8s segments, the data are sampled at the dataset sampling rate of 10Hz. Moreover, we then downsample each segment by a factor of 2 before using it as the input data, in order to reduce the model's computation complexity. After preparing the dataset, We spilt it into three sets, train set, test set and validation set.

Evaluation Metrics: We use two metrics: root of the mean squared error (RMSE) and Negative log-likelihood (NLL). RMSE is between the predicted trajectories with respect to the true future trajectories, over a prediction of 5 seconds. It helps us intuitively determine whether the predicted trajectory is accurate or not. Since our model generates a bivariate Gaussian distributions, the means of the Gaussian components are used for RMSE calculation.

RMSE has limitations on evaluating multi-modal predictions. Because RMSE is biased in favor of models that average the modes, but this average may not represent a good prediction. To address this limitation, we adopt the negative log-likelihood (NLL) of the true trajectories under the predictive distributions generated by the models. NLL cannot show the prediction result as intuitive as RMSE, but NLL give us better distributions.

Results Comparing and Analysis: To judge whether our work has been successful or not, it is vital to compare evaluation results with baseline model and other

famous model. So we choose two models using LSTM network to compare with, M-LSTM(Baseline Model) [15] and CS-LSTM [29].

Table 1: **Results:** Root mean squared prediction error (RMSE) and negative log-likelihood (NLL) values over a 5 second prediction horizon for the models being compared.

Metrics	Prediction Horizon(s)	M-LSTM	CS-LSTM	Our Model
RMSE	1	0.61	0.62	0.59
	2	1.34	1.29	1.36
	3	2.28	2.13	2.26
	4	3.21	3.20	3.16
	5	4.62	4.52	4.62
NLL	1	0.49	0.58	0.45
	2	1.93	2.14	1.93
	3	3.02	3.03	3.15
	4	3.65	3.68	3.64
	5	4.18	4.22	4.18

Table1 shows the RMS and NLL values of prediction error for the models being compared. Based on the trend of the error values associated with time evolution, we can find that as the prediction horizon increases, the RMSE and NLL errors of these three models increase monotonically. This is explicable and natural. Because when calculating RMSE, we use the means value of the bivariate Gaussian Distribution. But the actual value is less likely to be the average value. So each time we predict, there are some errors accumulated. After a period of time, the RMSE will accumulate to a relatively big value. NLL loss increases as the time passing for the reason that what we predict is a distribution at that time. The distribution indicates many possible future trajectories and provides us with a possible future driving direction. Since our ground truth trajectories is a single trajectory, the difference between prediction and

target certainly occurs. Besides, as the prediction time increases, the uncertainty for prediction increase. For this reason, NLL loss increase following the time evolvement. In fact, our model actually performs quite well. The lateral maneuver classes prediction accuracy is 0.97, and the longitude maneuver classes prediction accuracy is 0.89. From the accuracy of Maneuver Classes prediction, we can say our model works.

Compared with each other, these three models using LSTM shows nearly the same performance. Consider about the increasing value of these two kinds of error, our model performs well using NLL and CS-LSTM performs well using RMSE. However, our model has less training time, which means our model has higher efficiency. Tab2 shows the detailed computation time of different models.

Table 2: **Training Time:** Training time over 5 epoch for our model and M-LSTM. Both of the model were trained on NVIDIA Titan RTX

Model	Epoch	Time(s)
Our Model	1	2194
	2	2023
	3	2177
	4	2034
	5	2023
M-LSTM	1	11645
	2	11624
	3	11723
	4	11643
	5	11664

Table2 shows that our model reduce training time by 82.8 percent, which means it has much higher computational efficiency.

Visualizing Results: Using the mean values of prediction distribution, the same as the values used to compute RMSE, we draw the future trajectory of vehicle: Fig.6

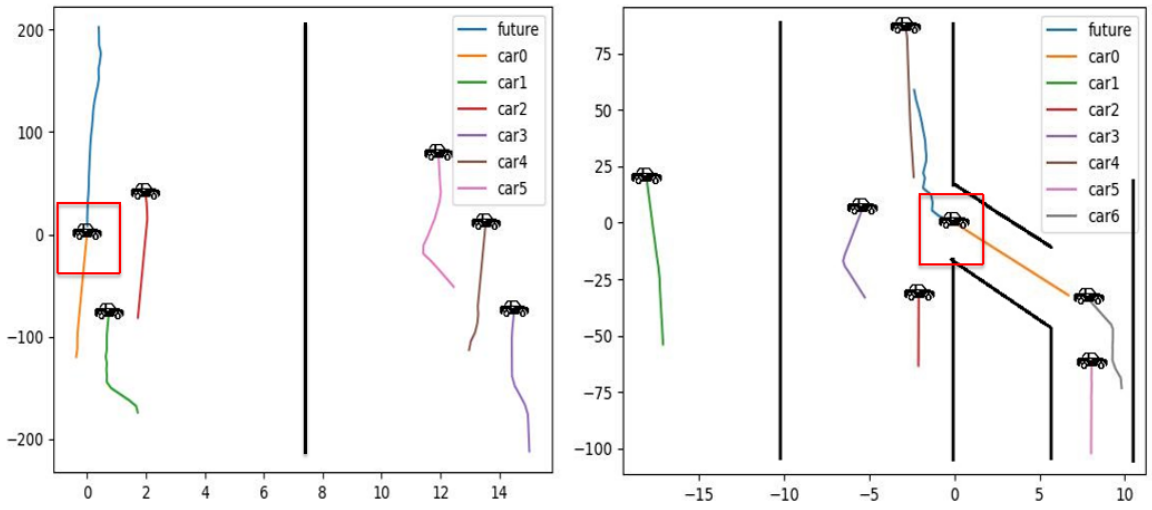


Figure 6

In Fig.6, the vehicles in the red rectangle are the vehicles being predicted. The two blue lines are the predicted future trajectory of them. Other vehicles are the surrounding which have tight social interactions with them. In Fig.6 we can see that the predicted trajectory is very reasonable.

8 Conclusions

In this project, we present a LSTM based trajectory prediction model. We consider about the time dimesion and social interaction dimension challenges when designing it. The experiments show the competitive results on multi-modal trajectory prediction with high efficiency. I hope this work can provide a common comparison point for future deterministic regressors and generative models in the field of multi-vehicle trajectory modeling.

9 Future Work

A key future direction is to renew present single-direction LSTM network to a bi-direction LSTM network. Since single-direction network only takes into account information from the previous moments, the bi-direction LSTM network can take into account the information from both previous and later time. Another way to improve is that we should compare our model with more baseline models, not only those LSTM based. In this way, we may find some other ideas in solving this problem.

10 Course Feedback

This is definitely a great course. Generally, the course are organized very well. From traditional pattern recognition algorithm to the present popular methods, the course allows students to grasp the core ideas of pattern recognition. Each lesson is a lecture about certain relevant fields in pattern recognition and each algorithm is described mathematically in detail by Prof.Wu. This helps students build up a high understanding of these algorithms. However, I think the hand-on experience is also important in learning new things.

From my perspective, students can get deeper understanding of these pattern recognition algorithms if they can apply these knowledge to practical problems, in addition to only learning the theory. Finishing tasks is always the most efficient and effective way to get familiar with the knowledge. These tasks can be organized into the form of homework or mini-projects.

EE433 is really a great course. Thanks very much for Prof.Wu's instruction and dedication through the quarter.

References

[1] J. Nilsson, J. Silvlin, M. Brannstrom, E. Coelingh, and J. Fredriksson, "If, when, and how to perform lane change maneuvers on highways", *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 4, pp. 68–78, 2016.

[2] M. Schreier, V. Willert, and J. Adamy, "Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems", in *17th international ieee conference on intelligent transportation systems (ITSC)*, IEEE, 2014, pp. 334–341.

[3] S. Ulbrich and M. Maurer, "Towards tactical lane change behavior planning for automated vehicles", in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2015, pp. 989–995.

[4] V. Kosaraju, A. Sadeghian, R. Martın-Martın, I. Reid, H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks", *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[5] 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*, 2016, pp. 961–971.

[6] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks", *arXiv preprint arXiv:1609.02907*, 2016.

[10] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks", *arXiv preprint arXiv:1511.05493*, 2015.

[11] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data", in *European Conference on Computer Vision*, Springer, 2020, pp. 683–700.

[12] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds", in *2018 IEEE international Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 4601–4607.

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation", *arXiv preprint arXiv:1406.1078*, 2014.

[14] A. Graves, "Generating sequences with recurrent neural networks", *arXiv preprint arXiv:1308.0850*, 2013.

[15] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms", in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1179–1184.

[16] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition", in *2013 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2013, pp. 4363–4369.

[17] C. Laugier, I. E. Paromtchik, M. Perrollaz, *et al.*, "Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety", *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 4, pp. 4–19, 2011.

[18] J. Schlechtriemen, F. Wirthmueller, A. Wedel, G. Breuel, and K.-D. Kuhnert, "When will it change the lane? a probabilistic regression approach for rarely occurring events", in *2015 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2015, pp. 1373–1379.

[19] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? a unified framework for maneuver classification and motion prediction", *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 129–140, 2018.

[20] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains", *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[21] Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani, "Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9813–9823.

[22] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2375–2384.

[23] T. Laurent and J. Brecht, "The multilinear structure of relu networks", in *International conference on machine learning*, PMLR, 2018, pp. 2908–2916.

[24] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis, "Dying relu and initialization: Theory and numerical examples", *arXiv preprint arXiv:1903.06733*, 2019.

[25] C. Cui and T. Fearn, "Modern practical convolutional neural networks for multivariate regression: Applications to nir calibration", *Chemometrics and Intelligent Laboratory Systems*, vol. 182, pp. 9–20, 2018.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, and E. Yang, "Zd facebook, ai research, z. lin, a. desmaison, l. antiga, o. srl, and a. lerer,"automatic differentiation in pytorch,"", *Neural Information Processing Systems, Tech. Rep*, 2017.

[28] J. Colyar and J. Halkias, "Us highway 101 dataset", *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, pp. 27–69, 2007.

[29] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468–1476.