Engineering Topic: Vehicle Trajectory Prediction in Autonomous Driving Systems

Zhiyuan Li

Description of the Engineering Topic:

Vehicle trajectory prediction has always been an essential component of autonomous driving systems. The main objective involves forecasting the future path of vehicles based on their historical and current states. Accurate prediction can help in path planning, collision avoidance, and enhances the overall safety and efficiency of autonomous driving technologies, making it a popular topic of exploration in the field.

Technical Problem and Difficulties

The primary technical problem is that we need to predict the future trajectory of vehicles with high accuracy and low latency. Difficulties arise from the non-linear and dynamic nature of vehicle movements, making it hard to approach from traditional methods.

In addition, one should also interpret the influence of surrounding environment and traffic, dealing with uncertain factors like human driving behaviors and sudden changes in road conditions. Moreover, since decisions need to be made in real-time, we must seek a method which has a relatively low computational load.

Literature Review and Available Approaches

The literature for trajectory prediction primarily focuses on Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks due to their effectiveness in handling sequential data.

Though both are powerful Neural Network which gives a prediction of future based on the past data, RNNs often suffer from vanishing and exploding gradient problems due to its architecture. On the other hand, LSTM, an advanced type of RNN, has been more successful in capturing long-term dependencies, making them more reliable for trajectory prediction.

Besides, the encoder-decoder architectures are also an available option for sequence-to-sequence predictions. It is able to encode sequences of vehicle states into a fixed-length context vector and decode it into future positions.

Proposed Algorithm

For this project, I have chosen to focus on the LSTM network, a proven method in sequence prediction tasks, including vehicle trajectory forecasting.

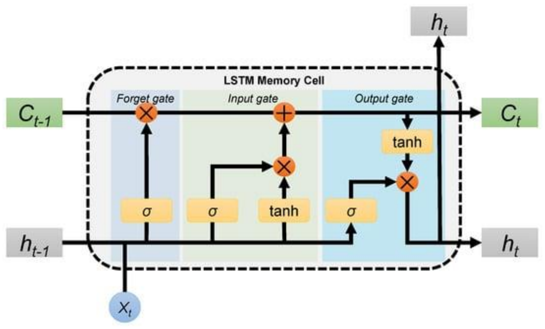

Figure 1 Illustration of basic LSTM unit.

We will be using the most classic LSTM model as is shown above. A basic LSTM is composed of forget gate which determines what percentage of long-term memory is remembered, an input gate which determines what percentage of input will contribute to long-term memory, and in the end the output gate which update the cell state, generate output for the next cell state.

As is mentioned, LSTM is immune to challenges of exploding or vanishing data by virtual of its special structure. Thus, LSTMs are adept at remembering information for long periods. The model will be designed to input a sequence of past vehicle states (position, velocity, heading) and output a sequence predicting future positions. The LSTM's ability to maintain state information makes it highly suitable for the task of predicting trajectories while considering the temporal dependencies in vehicle movements.

Code Implementation

The full python notebook has bees submitted, and Wandb link is also available below:

https://api.wandb.ai/links/zhiyuanl925/6nro4ox1

Verification of Correct Implementation

To verify my code correctly implements the algorithm, I build my LSTM based on standard LSTM model, and keeps monitoring the loss during the training process.

As is shown below, the loss functions in both training and validation decreases over training epochs and observe little oscillation or final state with high loss value. Last, the output from the model shows expected behavior, which will be presented in the Result section.

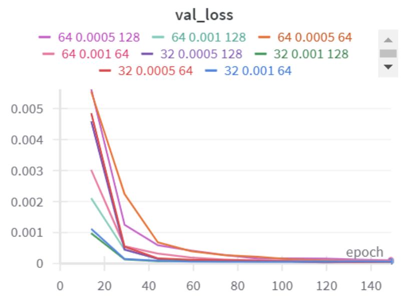

Figure 2 Validation Loss v.s. epoch

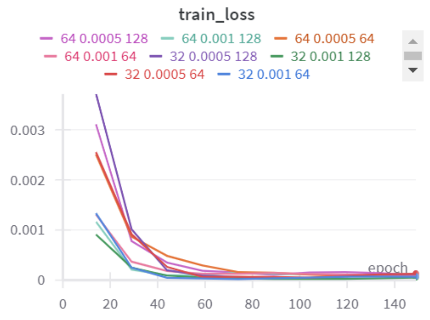

Figure 3 Training Loss v.s. epoch

Datasets, Settings, and Parameters

The project utilizes the Argoverse 2 Motion Forecasting Dataset, focusing on vehicle trajectory data. The dataset includes features such as position, velocity, and heading, grouped by unique track IDs. It should be noted that we are only predicting the trajectory of vehicle with FOCAL TRACK ID in this case, focal track represents The primary track of interest in a given scenario, it typically refers to the autonomous vehicle we are controlling. In comparison, we have SCORED TRACK which is the High-quality tracks relevant to the autonomous vehicle. We have an example illustration in the first part of submitted python notebook.

The problem is set as predicting the position_x and position_y of focal track at next time step based on the trajectory information in the past 5-time steps. For the LSTM model, sequence length was set to 5 accordingly.

The data was normalized using MinMaxScaler for better neural network performance. The network was trained with varying batch sizes (32, 64), learning rates (0.001, 0.0005), and hidden layer sizes (64, 128) to identify optimal parameters. Training was conducted over 150 epochs to ensure adequate learning while avoiding overfitting. The loss is calculated using Mean Square Error (MSE) against the actual data at the next time step.

Results and Interpretation

As we can see in previous sections, the training/validation loss keeps dropping, and so is the relative error which can be found in wandb link. Checking the parameters coordinate, we find we achieve the lowest loss when bath size = 33, hidden layer size is 128, and the learning rate is 0.0005.

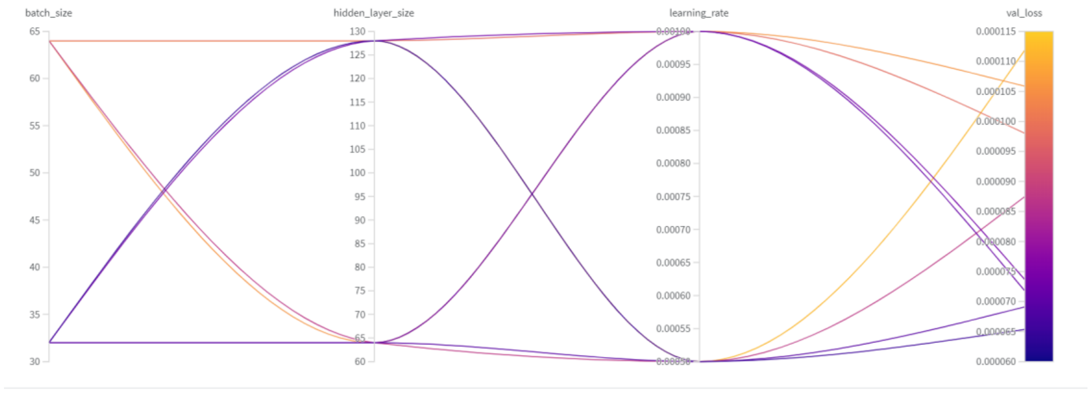

Figure 4 Parameters Coordinate

As we observed the prediction given the optimal parameters, we find most graph show satisfying result similar to figure 6, in which the actual and predicted position are relatively close to each other. However, we also encounter situations where prediction deviated from the actual data as is shown in figure. 7. In this case, the previous trajectory is far from straight line, thus our model might find it hard to make accurate predictions.

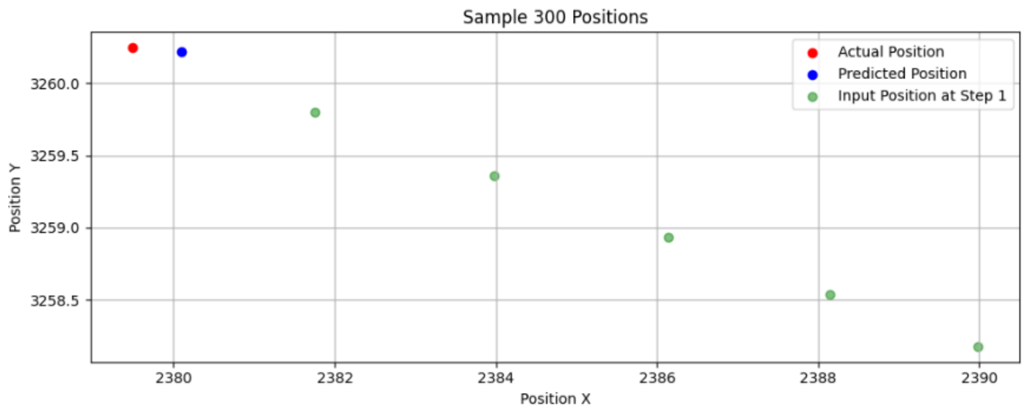

Figure 5 Prediction v.s. actual position (Sample 300)

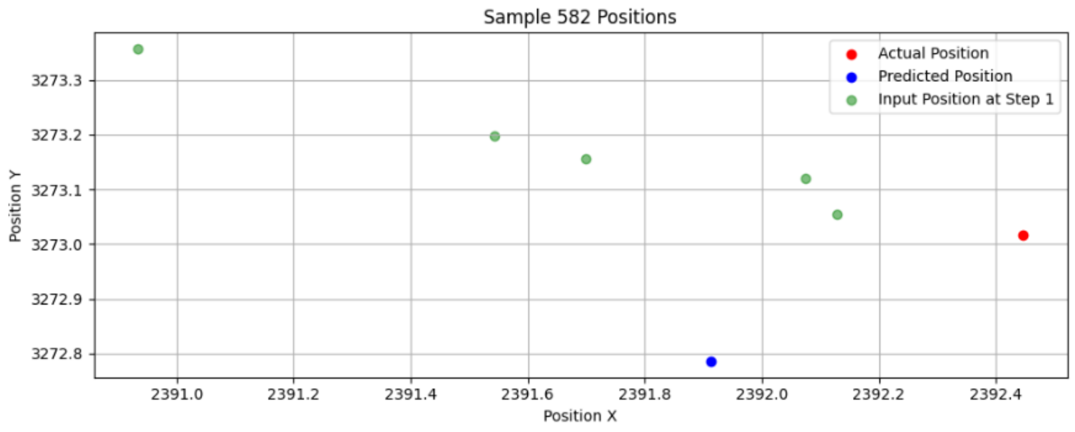

Figure 6 Prediction v.s. actual position (Sample 582)

Limitations and Improvements

As we can see, the model might struggle with erratic or unpredictable movements due to the fact that we are using past trajectory data for future predictions.

Improvements could include integrating more information such as road layouts, traffic conditions, or other environmental factors. Also, we could try more advanced models such as encoder-decoder, transformers and so forth. Additionally, further hyperparameter tuning and training with a larger variety of data could improve the robustness and accuracy of the model.

Reference

[1] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.

[2] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.