Autonomous Vehicle Motion Forecasting

Muchan Li, Sijie Liu, Zhendong Wang, Yujie Zhang

Summary

Team member:

- Muchan Li
- Sijie Liu
- Zhendong Wang
- Yujie Zhang

Overall, in order to solve the task of accurately forecasting autonomous vehicles' trajectory in the next 6 seconds using a 5 seconds position, we developed three versions of neural network models, which are **Sequence-to-Sequence Encoder-Decoder, Seq2Seq with attention, and Multi-Layer Perceptrons**.

- Our best model turned out to be the **MLP**.

In addition to learning the techniques of developing deep learning models using Pytorch, we learned a great deal from the following two perspectives:

1. Data preprocessing
    a. Especially that of proper data normalization and feature engineering.
2. Model selection and tuning
    a. Especially that of selecting model(s) that's best capable of capturing the underlying pattern of the data.

Introduction

Team Introduction - *Group 7*

Muchan Li

Data Science

Zhendong Wang

Data Science

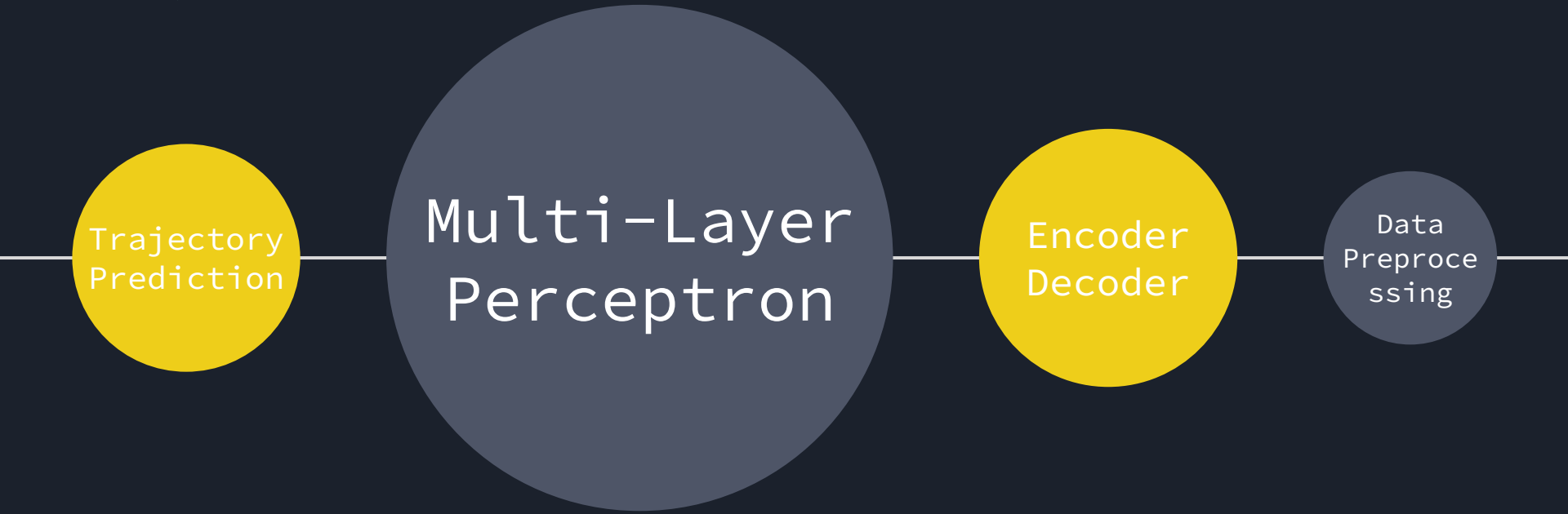
Sijie Liu

Data Science

Yujie Zhang

Data Science

Key Words


Trajectory Prediction
Multi-Layer Perceptron
Encoder Decoder
Data Preprocessing

Background Breakdown

Since “Trajectory Prediction” is one of the keywords that summarize our project, and that we’ve mention this task of trajectory/path forecast a number of times so far, we would like to formally define the problem our project and our model attempts to resolve before proceeding.

- Problem to investigate: given 5 timestamps worth of trajectory information about an autonomous vehicle deployed in one of the six cities of interest, predict the next 6 timestamps of trajectory of that vehicle.
  - Input space: vehicle’s trajectory coordinate information in 2D Cartesian coordinates (x,y). Since the data was sampled at 10 Hz (i.e. 10 samples per second timestamp), our full input set includes (x_1,y_1), (x_2,y_2), … till (x_{50},y_{50})
  - Output space: 6 timestamps worth of data in Cartesian coordinates sampled at 10 Hz in the format (x_1,y_1), (x_2,y_2), … till (x_{60},y_{60}).
- Why is this problem relevant?
  - One immediate effect of making accurate autonomous vehicle trajectory forecast is that it can allow engineers to better govern the algorithm and model that dictates autonomous driving, safeguarding both the people that are behind the wheels and the people surrounding.

Methodology

Data Processing & Engineering Tricks

Normalization

- Calculate individual trajectory minimum and trajectory maximum, normalize the input using *(input - trajectory minimum) / (trajectory maximum - trajectory minimum)*
- No need to deal with the scale -> Faster converge
- Benefit: reduce gradient's dependency on the scale/magnitude of parameters

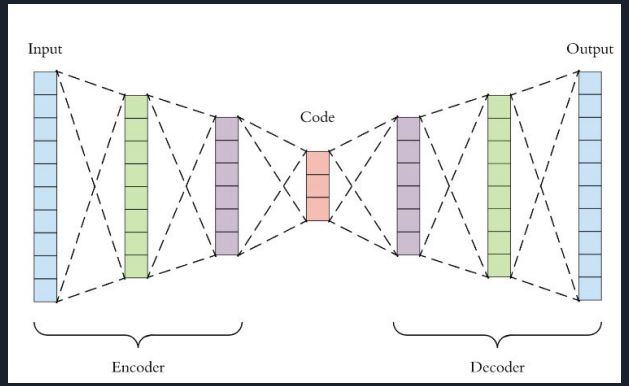
Translation

- Translate the outputs back into the original scale to generate the submission file by calculating *output * (trajectory maximum - trajectory minimum) + trajectory minimum*

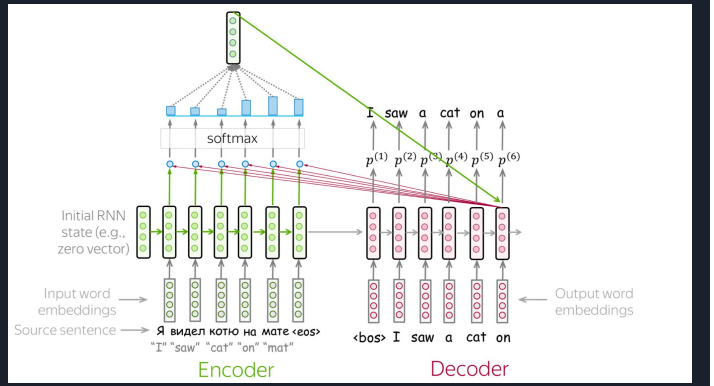
$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Deep Learning Models Explored

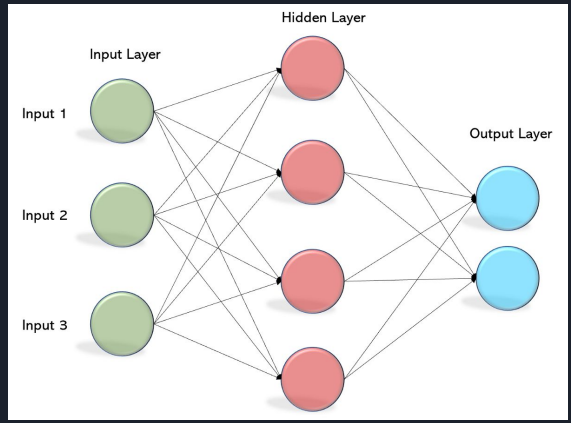
Encoder-Decoder



Seq2Seq with Attention



Multi-Layer Perceptron



Best Model Training Summary

Optimizer

- Adam

Training Steps

- For each city, we train a model and use the corresponding city's model to make prediction
- We used 80% of the training data to train the models, and validate the model's performance with the remaining 20%
- Batch size: 64
- Epochs: 10 (due to time constraints and training runtime, we didn't do much tuning to this)

Performance

- Position on private leaderboard: 27
- Private score: 61.78307

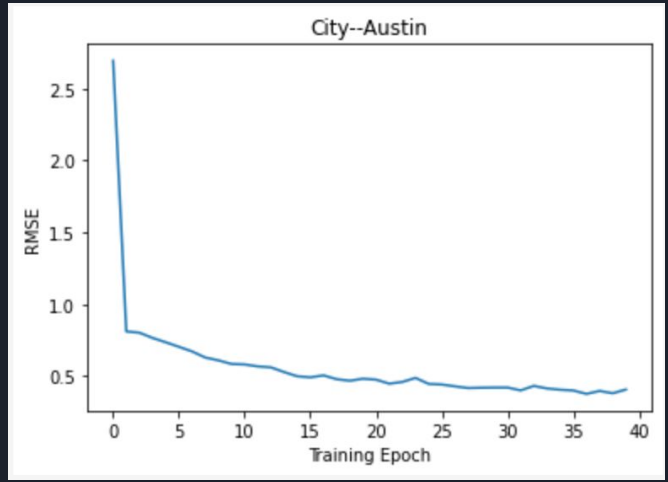
Experiments

Full Project Progression Timeline

Eventually explored and decided on MLP, which comfortably captures the majority of linear relations in the data

Experiment 1: Encoder-Decoder

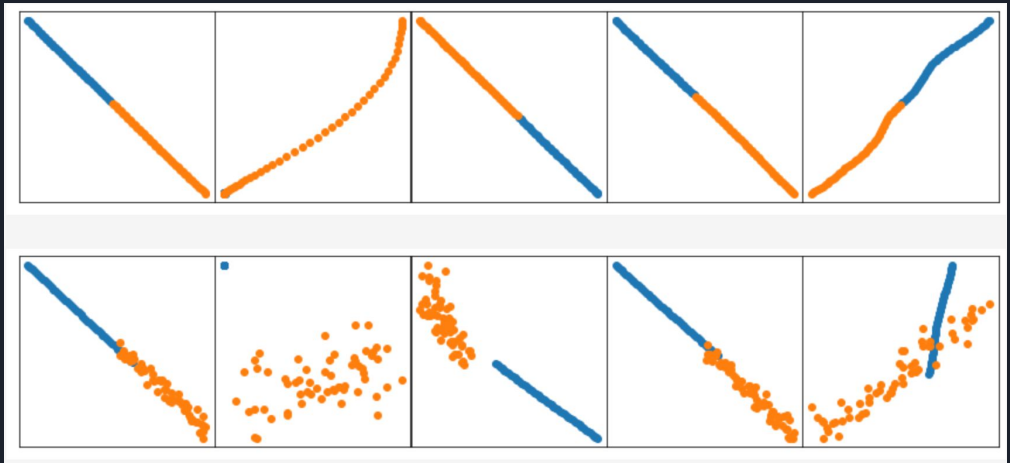
- We built our initial benchmarking model (the one we fully explored and based our milestone report off of) around the Encoder Decoder model foundation that was taught in discussion sessions.
- Tunings:
  - Number of linear layers/depth and hidden layer dimensions
    - Surprisingly, no matter how much more complex we made the encoder-decoder to be, the test performance did not improve on top of the default encoder-decoder model.
  - Batch sizes and number of epochs
    - We chose the number of epochs to be around 40 since we observe that our loss curve no longer descends significantly, take Austin as an example, as to the right.
  - Activation function
    - LogSigmoid activation proves to be optimal
  - Loss metric
    - MSEloss() proves to be the most suitable



Experiment 1 Continued

Inspect model predictions against the true trajectories:

- The 5 trajectories on the top row are truths. The bottom 5 are our model's predictions. For each plot, the blue line of scatters are the 50 pairs of input coordinates, the orange line of scatters are the 60 pairs of output trajectory coordinates.
- As we can visually tell, our model is getting good at learning and predicting linear paths (such as the prediction made on the 2nd row, 4th column, against the truth on its top) , but for the paths that are non-linear, it can only predict a blob of possible positions.

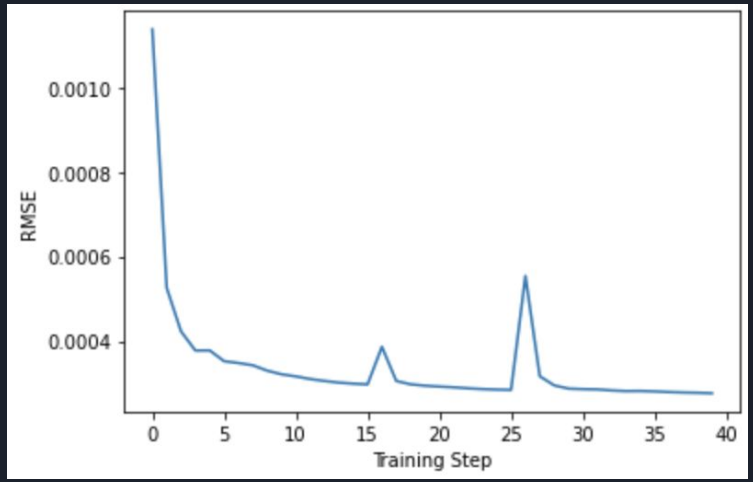


Competition Performance Update

Submission No.	Score
#1	4538.44310
#2	1034.24678
#3	1355.91321
#4	1098.29560

Experiment 2: Seq2Seq with Attention

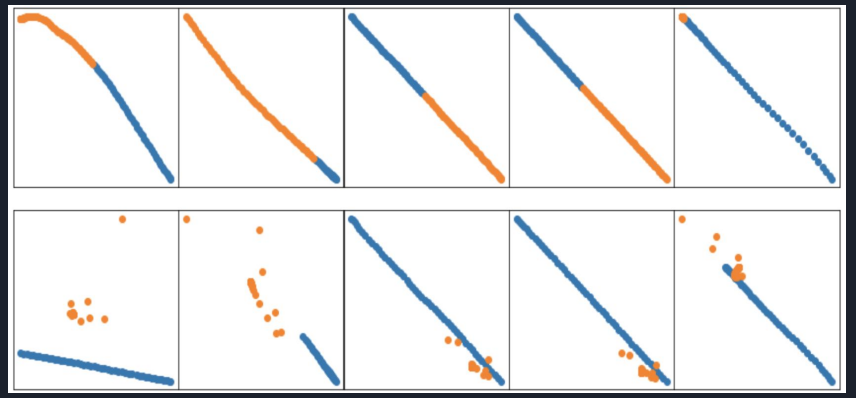
- Featured on the right is the Seq2Seq with attention model trained on "austin" dataset with a batch size of 50 and 40 epochs.
- The training loss generally follow the shape of the exponential decay. The loss magnitude is also a lot smaller. But the RMSE randomly bumps at epoch 16 and 26
- Tunings:
  - Number of hidden units in the single layer LSTM encoder and decoder
    - 128 hidden units was optimal out of the 2 others we tried.
  - Learning rate
  - Number of hidden layers (stacked LSTM)
    - However, we couldn't resolve the bugs that were thrown because of input dimensionality issue so we ended only using single hidden layer LSTM encoder and decoder



Experiment 2 Continued

Featuring the predictions made on Austin to the right:

- We now see improvement in our model's prediction in turns that it is now making more confident forecasts, as indicated by the closely-clustered blob of orange scatters on the bottom row.
- However, adding the attention mechanism along with LSTM units still don't predict sensible paths.

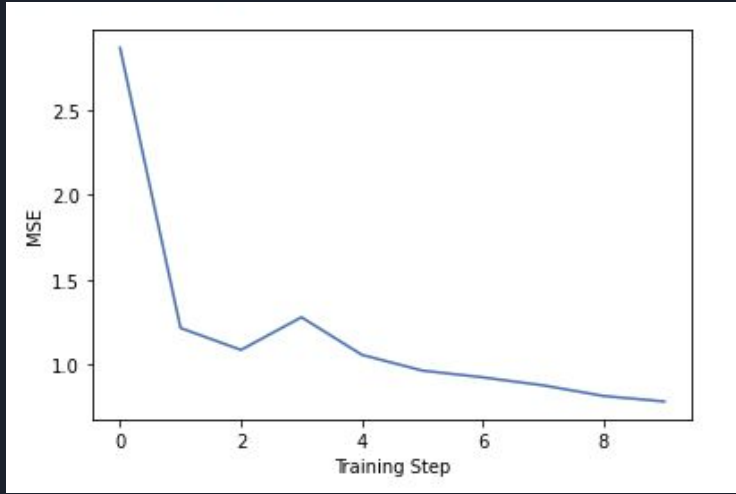


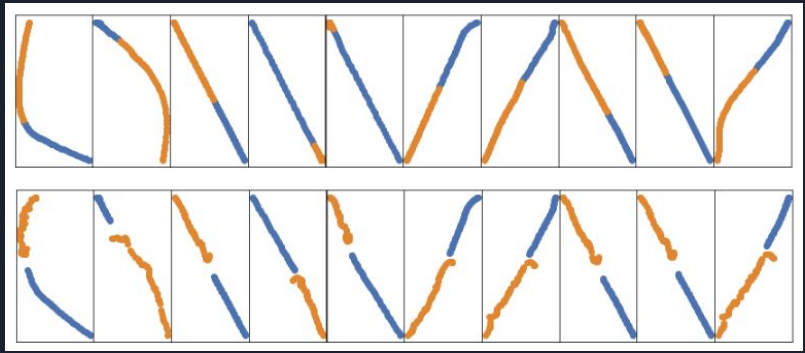
Competition Performance Update: 705.00347

Experiment 3: MLP

- MLP model trained on "austin" dataset with a batch size of 64 and 10 epochs
- The training loss followed the shape of the exponential decay
- Accurately predict the direction of the car if the the car is moving in a line
- Roughly predict the trajectory if the car makes some small turns
- There is always a small deviation between the start of the prediction and the end of the input

Competition Performance Update: 62.38054





Discussion

What have we learned

- Work as a team
- Do background research and always search on the internet to improve the models
- Be patient and try different models
- Start early so that we can have more time for the training process
- When in doubt, always helpful to return to“listening” to the data directly through exploratory data analysis.
- Deep neural network training is extremely time consuming so it’s good to build the habit of subsetting data for different training and testing purposes and experiment the models only on the training fraction.

Future Work

- Data preprocessing and engineering:
  - More rigorous feature engineering
    - Add social features such as the count of neighbors for each agent
    - Extract and build on more useful features from data, such as velocity, acceleration, etc.
- Model training:
  - Fine tune MLP model's hyperparameters
    - Number of hidden units per layer, number of layers, batch size, etc.
  - Incorporate learning rate scheduler to help speed up training and model convergence.
- Broader scope of model exploration:
  - Explore other architectures such as stacked-LSTMs, transformers, recurrent MLP.