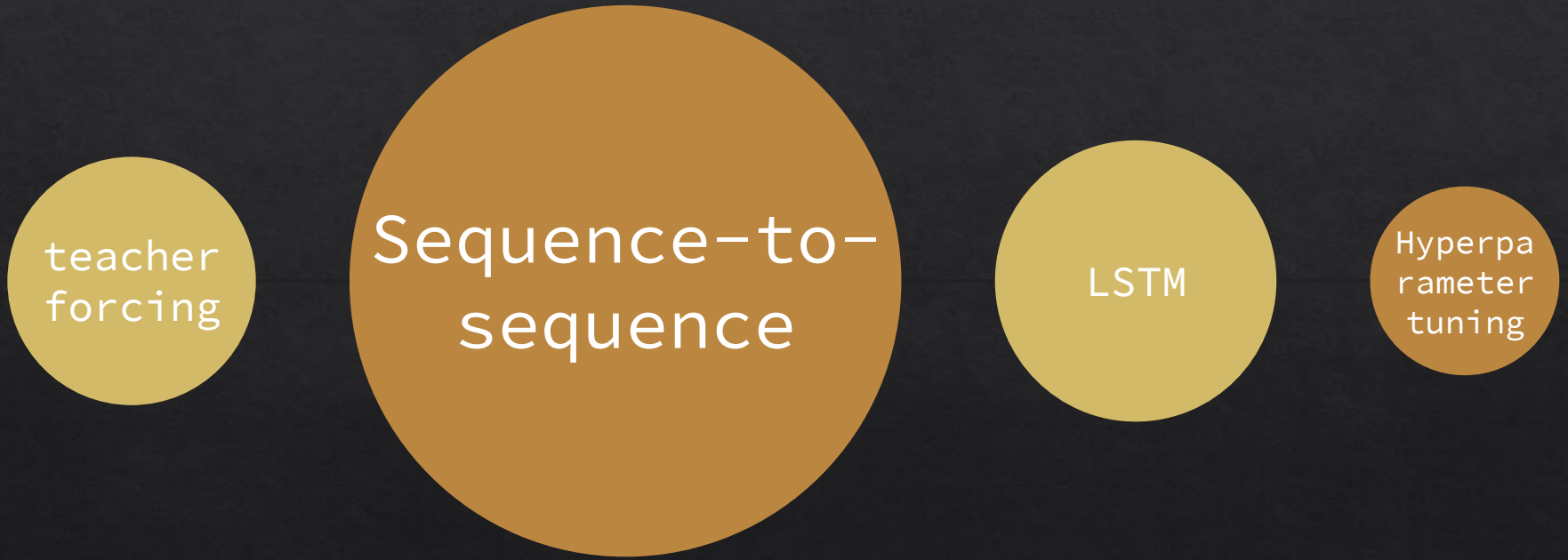
Vehicle Motion Forecasting
CSE 151B Project Final Presentation

Group TechnoCore

Summary

- Three team members: Zhongrui Cao, Jiayan Dong, and Hang Wang
- We use a Seq2seq (Encoder-Decoder) model to predict vehicle movement.
- Lesson we learned:
  - Data preprocessing is very important!
  - Experiments are very important! Certain steps in the model not only fail to improve the model, but also make the model worse.

Key Words


teacher forcing
Sequence-to-sequence
LSTM
Hyperparameter tuning

Introduction

Team Introduction

Hang Wang: Senior CS major

Jiayan Dong: Junior CS major

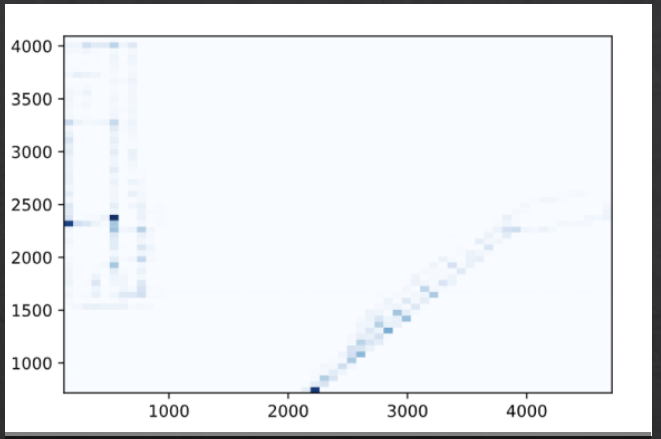
Zhongrui Cao: Junior CS major

Methodology

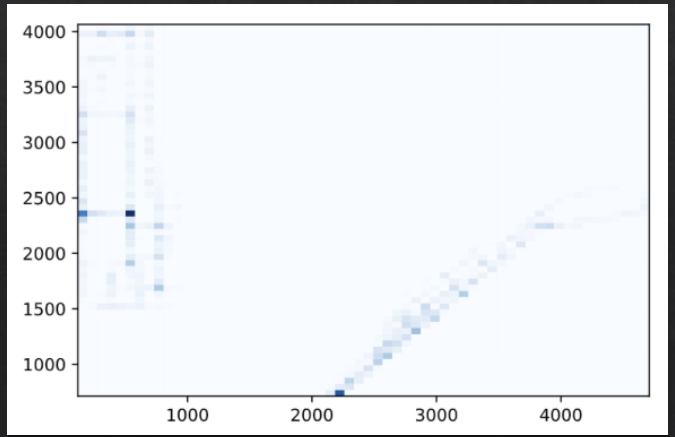
Data Processing

We only used the position information of the agent vehicle: 19 continuous 2d input positions and 30 continuous 2d output positions.

We noticed that the input and output location data are basically distributed in two different areas, which is not conducive to our prediction of location changes.



Distribution of input absolute positions.



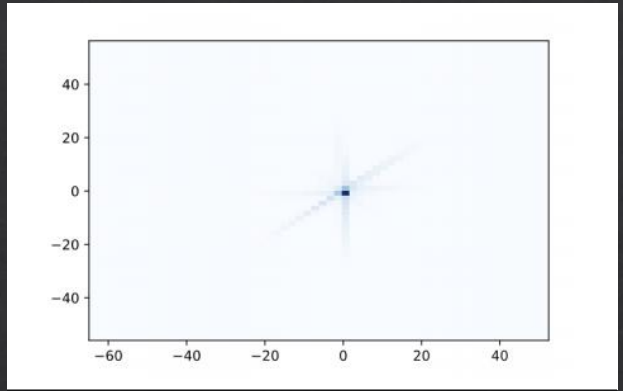
Distribution of output absolute positions.

Data Processing

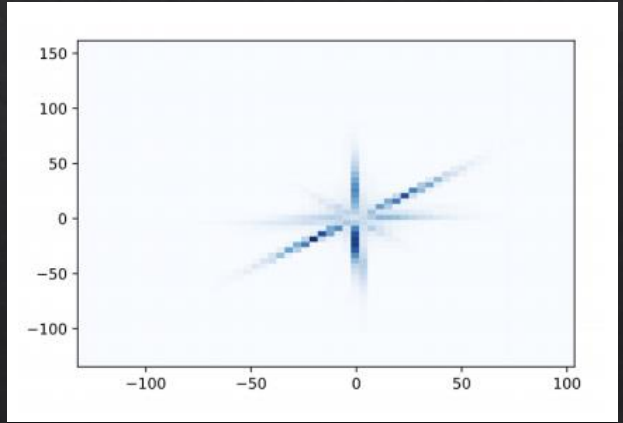
So, we preprocess the input data to convert the absolute position into a relative position (relative to the first).

The relative input positions are all concentrated around ±20, while the relative output positions are all concentrated around ±30. This is in line with our speed data.

Through experimentation, it is proved that using relative position can provide a better prediction.



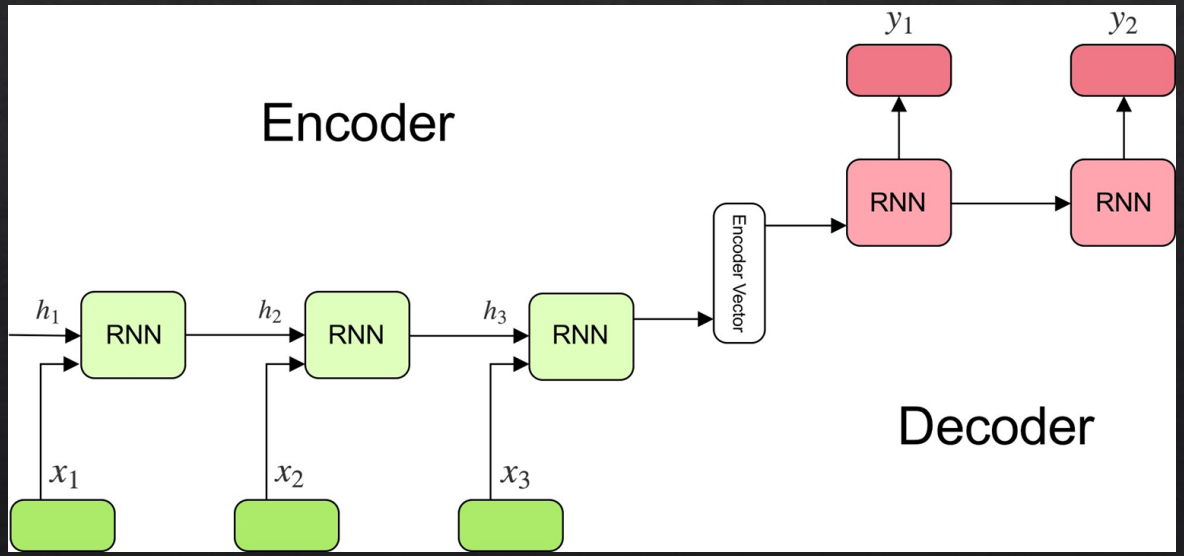
Distribution of input relative positions



Distribution of output relative positions

Deep Learning Model

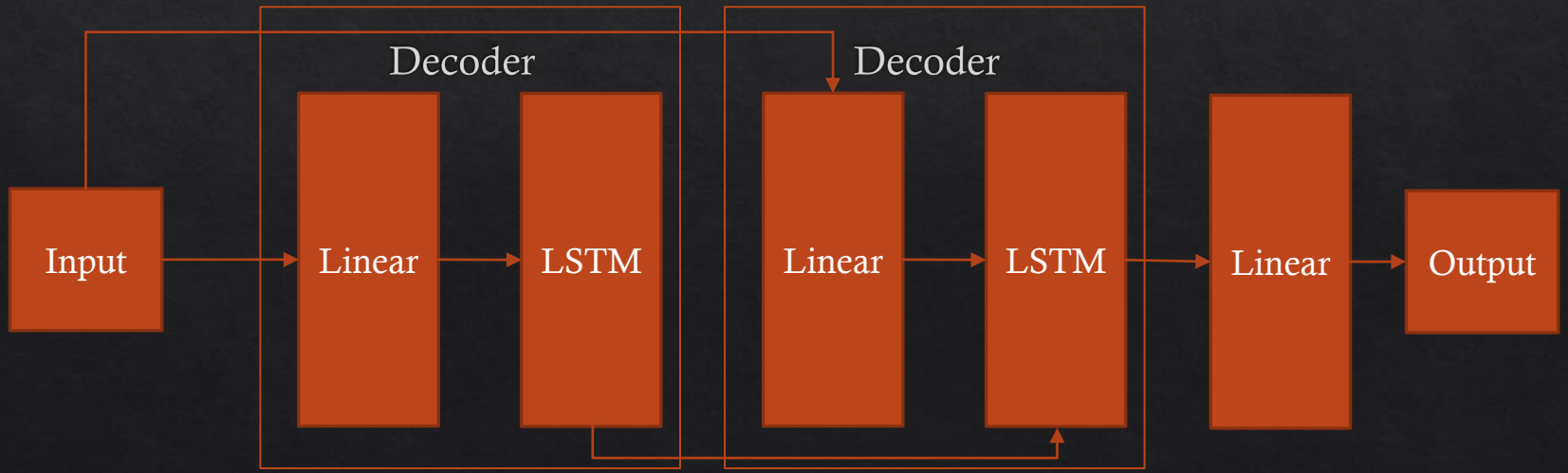
Our goal is to predict time series, so we first thought of Recurrent neural network (RNN) because it can handle time series well. Then we noticed that the input and output lengths are not the same, so we naturally thought of using Seq2Seq (Encoder-Decoder model).



Deep Learning Model

Our Encoder and Decoder both consist of a linear layer and a 4-layer LSTM. There are also a linear layer after the Decoder to output the result.

Among them, we use ReLu in the linear layer as activation function, and use dropout in LSTM to prevent overfitting.



Engineering Tricks

- The provided 205942 training data is converted into a training set and a validation set at a ratio of approximately four to one. There is also a small part of data (10,000) for quick testing.
- We used Adam optimizer to train the model. After many attempts, a learning rate of 0.001 was used.
- A high batch size is selected to speed up training.
- The training error and validation error after each epoch are also recorded to detect whether the model is under-fitting or over-fitting.
- Teacher forcing was also used to speed up training, but later it was discovered that it would reduce the accuracy of the model.

Experiments

Experiment 1 – Batch size

We used a GTX 1060 with 6GB of video memory to accelerate training.

A batch size of 2048 is the upper limit of video memory usage.

We test epoch duration in our small dataset (10,000 data).

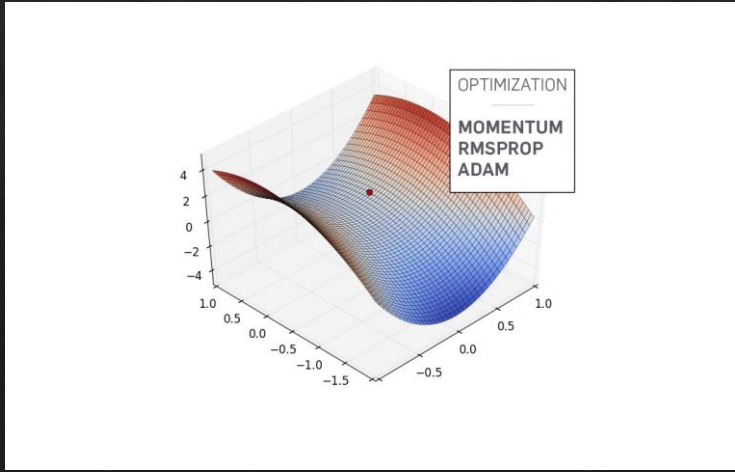
We can notice that the larger the batch size, the faster the training speed, we finally use 1024 as the batch size.

Experiment 2 – Optimizer

We tried different kinds of optimizers, including SGD, AdaGrad, RMSProp and Adam optimizers.

The final result shows that the Adam optimizer is the best choice, where the initial learning rate is set to 0.001.

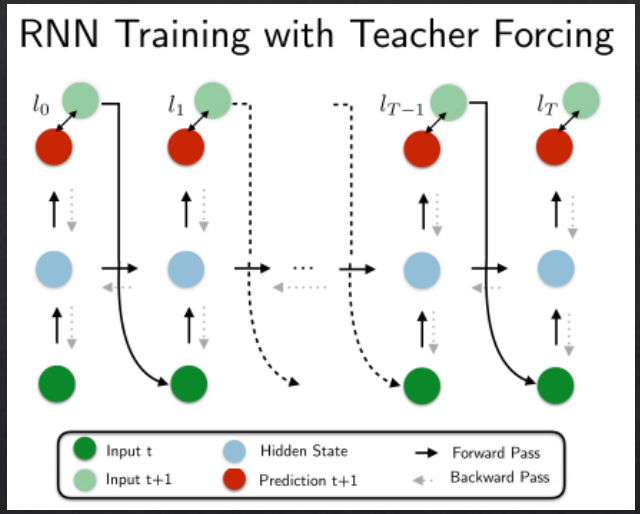
Optimizer	SGD	AdaGrad	RMSProp	Adam
Final validation error (RMSE)	8.3	8.0	7.5	7.3

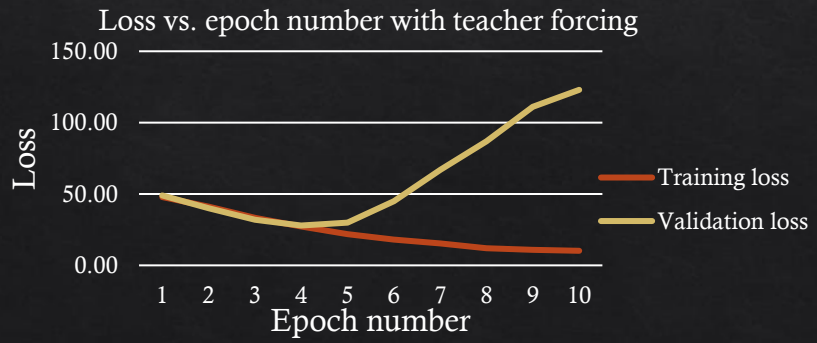


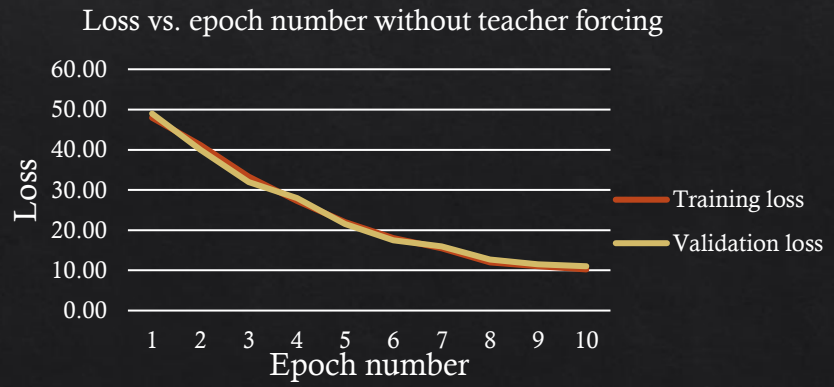
Experiment 3 – Teacher forcing

We originally thought that using teacher forcing would make our model fit better and faster, but in fact it reduces the accuracy of the model.

Teacher forcing introduces Exposure Bias, which makes our RNN model too dependent on the possibly provided real output, thereby reducing the accuracy of the actual prediction.







Discussion

What we learned

- Start Early, Start Often!
  - It takes a lot of time to read related materials, and it also takes a lot of time to train the model. Adjusting hyperparameters is even more time consuming.
- Analyze and Preprocess the data before model training
- The Seq2Seq model, LSTM and how it works and how to implement in python
- Beware of overfitting in RNN and always check validation error
- The importance of hyperparameters tuning, even small hyperparameter changes can have a big impact on the model
- Understanding the various algorithms used can effectively help identify bugs.

Future Work

- Doing more feature engineering and pre-processing, there are still information we didn’t use
- Maybe finding a fine-tuned and simpler model to avoid overfitting
- Tuning our hyperparameters further

Reference

- [1]Curow. “Curow/Vehicle-Trajectory-Prediction.” GitHub, github.com/curow/vehicle-trajectory-prediction.
- [2]“Getting Started with JAX (MLPs, CNNs amp; RNNs).” Rob’s Homepage, 16 Mar. 2020,roberttlange.github.io/posts/2020/03/blog-post-10/.
- [3]Jagjeet-Singh. “Jagjeet-Singh/Argoverse-Forecasting.” GitHub, github.com/jagjeet-singh/argoverse-forecasting.
- [4]“NLP From Scratch: Translation with a Sequence to Sequence Network and Attention ¶ .” NLP From Scratch:Translation with a Sequence to Sequence Network and Attention - PyTorch Tutorials 1.8.1+cu102 Documentation,pytorch.org/tutorials/intermediate/seq2seq t ranslation t utorial.html.

Thank you!