CSE151B Project Fianl Report

Steven Kan*
Department of Computer Science
University of California, San Diego
San Diego, SD 92122
s1kan@ucsd.edu

Li-An Wu
Department of Computer Science
University of California, San Diego
San Diego, SD 92122
l6wu@ucsd.edu

Bowen Shi
Department of Computer Science
University of California, San Diego
San Diego, SD 92122
yashi@ucsd.edu

Zhiyang Xu
Department of Computer Science
University of California, San Diego
San Diego, SD 92122
zhx011@ucsd.edu

GithubLink:https://github.com/WaWaWu0808/CSE151B_Project_Team39

1 Task Description and Background

1.1 Problem A

Describe in your own words what the deep learning task is and why it is important. Provide some real-world examples where solving this task can impact on our daily life and the society.

- In this project, the task is regression. We want to predict the future three seconds positions of a tracked vehicle given initial two seconds. This task is important because motion forecasting can be used on vehicles to predict their position and potentially increase the safety.

- Some real-world examples that this task can have great impacts on:
  - self autonomous vehicles: in order to achieve self autonomous vehicle, predictions of surrounding vehicles would be tremendously useful such that target vehicle can make the right judgements to avoid collisions.
  - maps: if maps have the data on accurate predicted positions of vehicles, they could provide drivers the most optimal route which would save people a lot of time and frustrations during rush hours.

1.2 Problem B

Use Google Scholar or other internet resources to research on this task. What type of methods have been examined before? Include some references and discuss their ideas in a few sentences. You can use Bibtex to manage your bibliography.

- In source (1) section 5.3, we see that K Nearest Neighbor and Lstm Encoder-Decoder have been tried.
  - For K nearest Neighbor, trajectories are queried by (x^t, y^t) for $t = 1, ..., Tobs$. To make K predictions, we performed a lookup for K Nearest Neighbors.

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

Preprint. Under review.

* For the Lstm encoder-decoder, the input is (x^t, y^t) for $t = 1, ..., Tobs$ and output is (x^t, y^t) for $t = Tobs + 1, ..., Tpred$. This is limited to one prediction because we used a deterministic model.

- In source (2), Guo Xie and his team conducted motion trajectory prediction using CNN and LSTM. Specifically, they had the idea of fusing two neural networks of a convolutional neural network (CNN) and a long short-term memory network (LSTM). Furthermore, they used box plot eliminate abnoraml values then merged CNN space expansion and LSTM time expansion for predictions. At the end of their experiment, they concluded that CNN-LSTM method is more accurate and features a shorter time cost which means they successfully kept the benefits of both CNN and LSTM.

1.3 Problem C

Define the input and output in mathematical language and formulate your prediction task. From the abstraction, do you think your model can potentially solve other tasks beyond this project? If so, list some examples and explain your rationale.

- In this case, the input is a batch of past two seconds positions. The input x is a tensor of size (batchsize, 19 , 2)

- In this case, the output f(x|w) is the batch of predicted positions for the future three seconds. The output x is a tensor of size (batchsize, 30, 2)

- Yes, from the abstraction, we do think the model can potentially solve the task. In the project, we tried out the multi-layer perceptron and Lstm encoder-decoer. For the MLP, although we are asking a direct relationship between past positions and future positions, we believe the weights between each layer will still learn some pattern. For the Lstm encoder-decoder, we believe the encoder part can learn the patter of the past input and generate

2 Exploratory Data Analysis

2.1 Problem A

Run the provided Jupyter notebook for loading the data. Describe the details of this data set. Your description should answer the following questions:

- What is the train/test data size?
  - The train size is 205942, and the test data size is 3200.

- How many dimensions of inputs/outputs in the raw data?
  - The dimensions for inputs in the raw data is $19 * 4 * 60$.

  - The dimension of the outputs in the raw data is $30 * 4 * 60$.

- What are the meanings of these input/output dimensions?
  - For the dimensions of inputs, 19 represents the time steps in 1.9 seconds, 4 represents the position and velocity of x and y, and 60 represents all the sampled vehicles (including dummies).

  - For the dimensions of outputs, 30 represents the 30 time steps in 3 seconds, 4 represents the position and velocity of x and y, and 60 corresponds to all the input vehicles (including dummies).

- What does one data sample look like?

  - input positions: [566.92871094 567.08374023 566.95080566 566.98754883 567.0489502 567.12310791 567.05877686 567.15899658 567.22442627 567.27709961 567.28881836 567.30480957 567.34729004 567.36517334 567.44970703 567.5020752 567.48840332 567.5345459 567.57092285]

  - output positions: [567.57348633 567.65936279 567.79882812 567.83984375 567.88226318 567.9029541 567.9072876 567.93145752 567.98901367 567.99414062 568.01226807 568.03735352 568.07861328 568.05999756 568.10357666 568.11450195 568.08642578 568.09991455 568.13446045 568.14654541 568.09283447 568.10559082 568.10858154 568.12786865 568.17224121 568.07336426 568.3213501 568.24176025 568.33892822 568.28613281]

2.2 Problem B

Perform statistical analysis to understand the properties of the data. Your analysis should at least answer the following questions.

- what is the distribution of input positions/velocity (magnitude) for all agents?

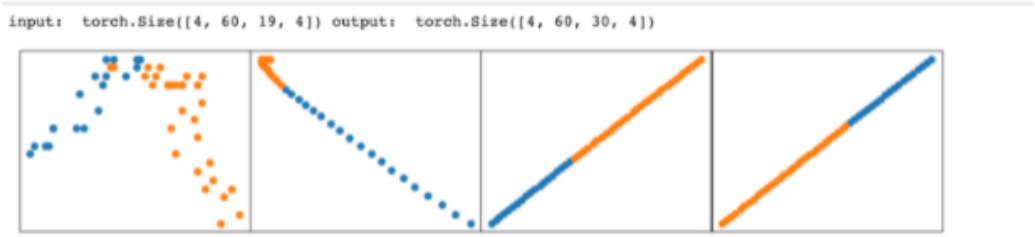


Figure 1: Scatter plots of a 4 random target samples

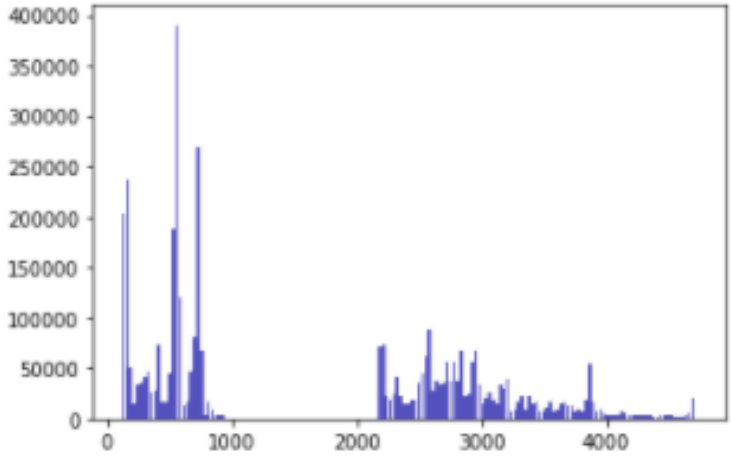


Figure 2: Histogram of pin x distribution

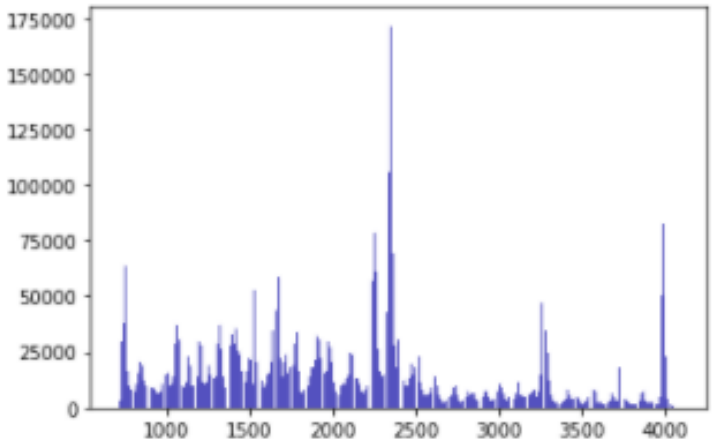


Figure 3: Histogram of pin y distribution

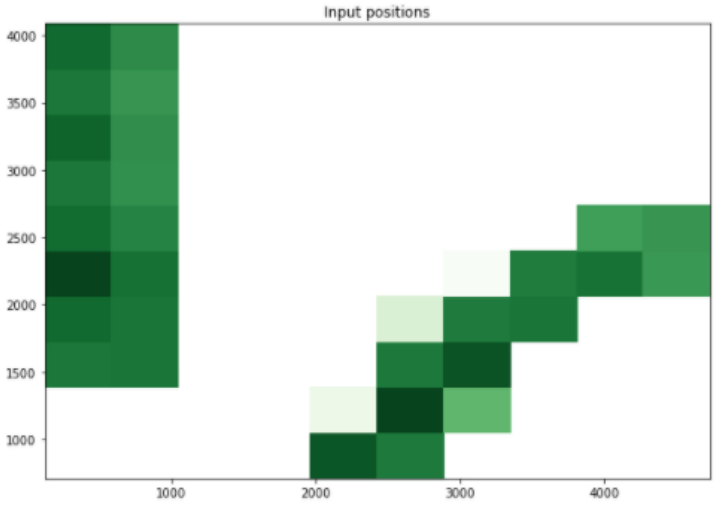


Figure 4: 2D Histogram of x and y input positions for target agents.

- In figure 1, the orange dots represent the output positions and blue dots represents input positions. We use these simple scatter plots to have a general idea of the positions of input and output

- Figure 2 and 3 shows the distribution of x and y separately for all target agents.

- In figure 4, we used a 2D histogram to plot positions of target agents. We will further analyze the relationship between input and output using figure 4 and 7.

- what is the distribution of output positions/velocity (magnitude) for all agents?

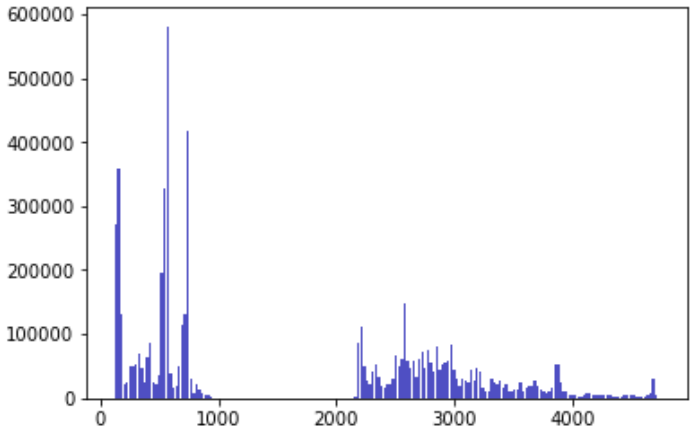


Figure 5: Histogram of pout x distribution

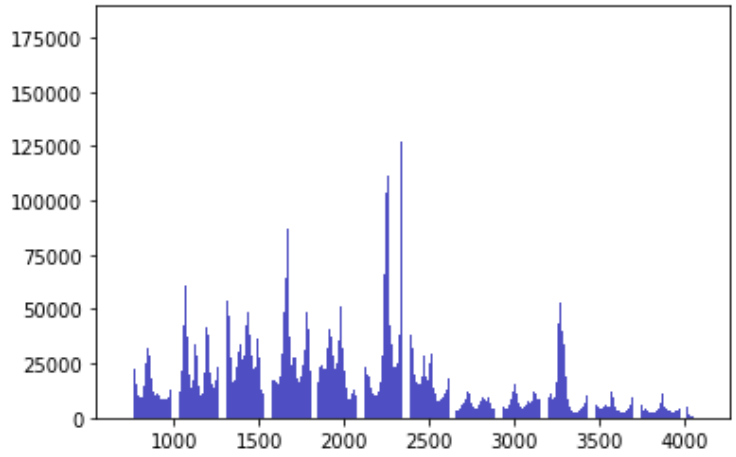


Figure 6: Histogram of pout y distribution

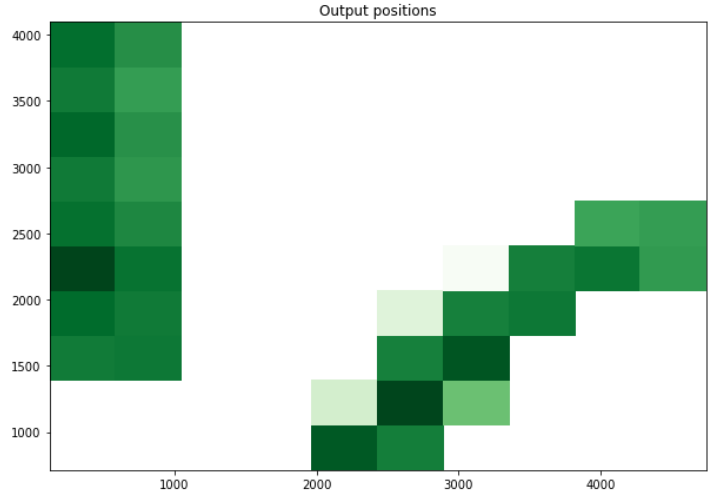


Figure 7: 2D Histogram of x and y output positions for target agents.

– In figure 7, we used 2D histogram to plot the output positions of target agents. As shown in figure 6 and 7, the darker the bin, the more concentrated data points are. We can see that there are more points lie in larger (x,y) in output than input. This behavior is expected as vehicles travel a little bit further with respected velocity in output. Moreover, we can also have a general idea of the directions of velocity such

that majority of the velocities are in positive direction.

- what is the distribution of positions/velocity (magnitude) for the target agent?

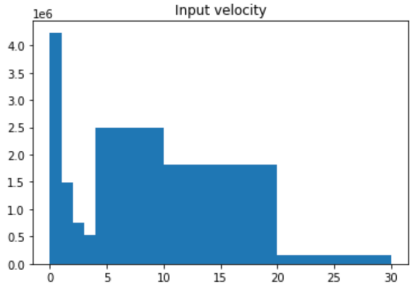


Figure 8: Distribution of input velocity (magnitude) of all agents

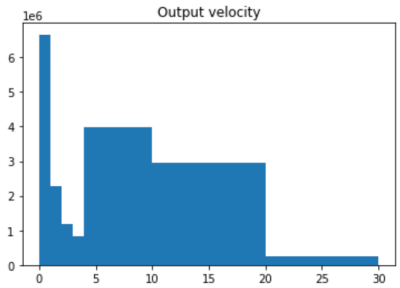


Figure 9: Distribution of output velocity (magnitude) of all agents

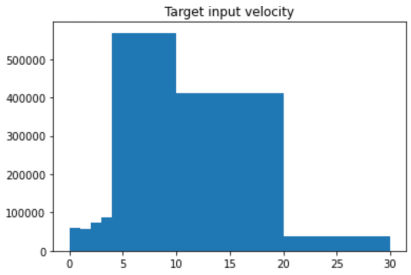


Figure 10: Distribution of input velocity (magnitude) of target agent

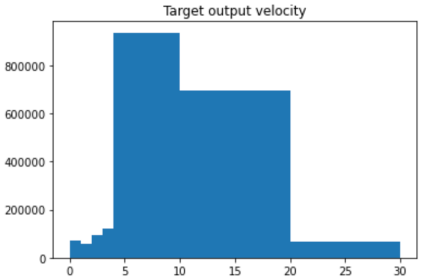


Figure 11: Distribution of output velocity (magnitude) of target agent

  - Figure 8 and 9 show the distributions of overall velocity for input and output. As we can see, velocities do not change much from input to output. Hence, we can suspect that most vehicles are not accelerating. This aspect is important for our prediction because now we can exclude accelerations from our model which might help with reducing dimensions for our models.

  - Figure 10 and 11 show the distributions of velocity for all target input and output. In most cases, they are pretty similar to the overall distribution which lie between 5 to 20 magnitude. In our model, we will be using target positions and velocities to predict future positions.

2.3 Problem C

Process the data to prepare for the prediction task. Describe the steps that you have taken to process the data. Your description should at least answer the following questions.

- Did you use any feature engineering? If yes, how did you design your features? Explain your rationale.

  - In our most optimal model, linear regression, we did not use any feature engineering. We simply mapped input positions to output positions using fully connected layers with dropout and relu activation. We tried using feature engineering in our LSTM model, however, we could not achieve the same result as linear regression model. In our LSTM model, we tried to incorporate velocities into the model because we thought by adapting different velocities would give us better results.

- How did you normalize your data? Why did you choose this normalization scheme?

- We did not normalize our data because for linear regression model we did not find the need to normalize our data and thought it could potentially give us less accurate results due to the normalization.

- Did you use the lane information provided in the dataset. If yes, how did you exploit this information.
  - In our data processing, we did try to exploit lane information by simply create a new array corresponding to each scene. However, for both linear and LSTM model, we found it difficult to incorporate this feature into our model.

3 Deep Learning Model

3.1 Problem A

Describe the deep learning pipeline for your prediction task and answer the following questions.

- What are the input/output that you end up using for prediction after pre-processing?

  In data visualization, we explore the distribution of x position and y position are significantly different.
  We train the position x model and position y model separately and combine the result at last.
  Thus, the input and output are also separated by x and y. we have 4 different types of input and output after pre-processing

  - 1. For MLP experiment 1:
    - The input is a flattened of p_in(19) and v_in(19) for x and y model (1 x 38).
    - The output is flattened of p_out(30) for x and y model (1 x 30).
  - 2. For MLP experiment 2:
    - The input is a flattened of p_in(19), v_in(19), and city(1) for x and y model (1 x 39).
    - The output is flattened of p_out(30) for x and y model (1 x 30).
  - 3. For Lstm encoder-decoder experiment 1:
    - The input is the sequence on past 19 seconds of positions with shape of (19 x 1) and format of (p_in(t)) for both x and y model.
    - The input is the sequence on past 30 seconds of positions with shape of (30 x 1) and format of (p_out(t)) fr both x and y model.
  - 4. For Lstm encoder-decoder experiment 2:
    - The input is the sequence on past 19 seconds of positions and velocity with shape of (19 x 2) and format of (p_in(t), v_in(t))
    - The input is the sequence on past 30 seconds of positions with shape of (30 x 2) and format of (p_in(t), v_in(t))

- What is your loss function? If you have multiple alternatives, discuss your ideas and observations.

  - Our loss function has always been the RMSE throughout the project. This is because we think using the same loss function as the Kaggle project would help us converge to the expected result.

- How did you decide on the deep learning model? Explain the rationale given the input/output.

* For MLP, we think that MLP can basically solve any nonlinear problem with enough capacity. It can also act as a good baseline model for the project

* For Lstm encoder-decoder, we think the encoder and decoder mechanism resembles the structure to our goal to predict the future timestamp based on past timestamp. Thus think it would be a good fit to capture the time sequence feature.

3.2 Problem B

Describe all the models you have tried to make predictions. You should always start with simple models (such as Linear Regression) and gradually in-crease the complexity of your model.

- Use an itemized list to briefly summarize each of the models, their architecture, and parameters, and provide the correct reference if possible.

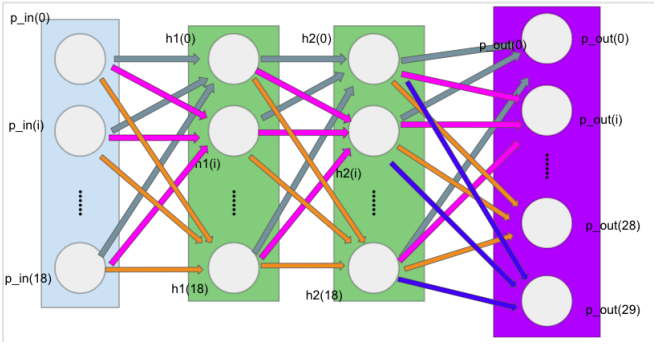
  * In data visualization, we explore the distribution of x position and y position are significantly different.
    Thus we train the position x model and position y model separately and combine the result at last.

  * 1. The first model we use is the MLP. The architecture contains 2 hidden Layers; each hidden layer contains 512 neurons and activation function of ReLu; the output layer with activation function of Linear.

  * 2. The second model we use is the lstm encoder-decoder. The architecture has the encoder part and the decoder part that both use Lstm as thier internal unit.
    In the model, it has one layer of one layer of lstm for encoder and one layer of lstm for decoder. Both Lstm layer have hidden unit of 256 and activation function of Relu. At last, there is one layer of fully connected layer with number of input tokens unit and activation function of Linear.

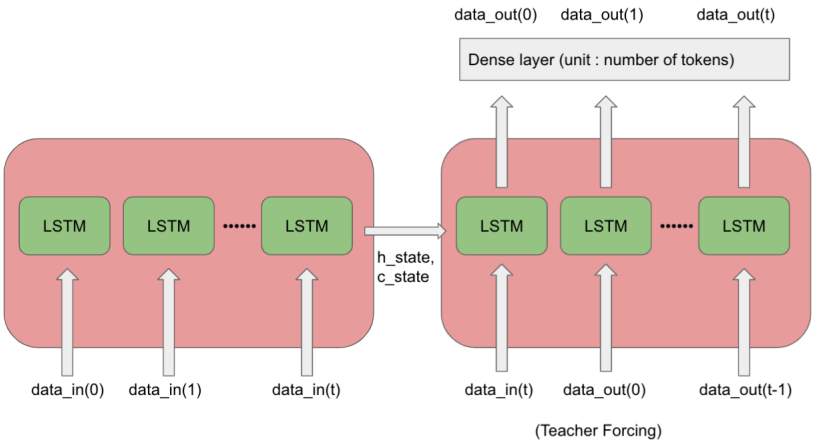
- If you end up designing your own model architecture, include a picture/sketch of your model architecture. Explain why you choose such a model.

  * For MLP model it looks like



We think that MLP can basically solve any nonlinear problem with enough capacity. It can also act as a good baseline model for the project.

* For the Lstm encoder-decoder models it looks like



We think the encoder and decoder mechanism resembles the structure to our goal to predict the future timestamp based on past timestamp.

– Describe different regularization techniques that you have used such as dropout and max-pooling in your model.

* we had attempted to try dropout and batch normalization methods but the training and validation loss have been significantly high during training. Thus we believed that regularization wouldn't be a good fit the data set and we exclude the regularization for the rest of the experiments.

4 Experiments Design

4.1 Problem A

Describe how you set up the training and testing design for deep learning. Answer the following questions:

– What computational platform/GPU did you use for training/testing?

* We are using UCSD dataHub which supports 8 CPU, 1 GPU, and 16GB RAM. If GPU is available, we will be using GPU to train our model. Although modern CPU with multiple cores is capable in parallel computing, GPU in this case would be more superior with hundreds and thousands of cores. Specifically, we will be using Compute Unified Device Architecture (CUDA) which is only supported by Nvidia GPUs.

– How did you split your training and validation set?

* We first load the file paths of each pickle file, and there are 205942 pickle files. Then we shuffle those pickle file paths and split the training and validation with ratio of 9:1. Thus, we have 185300 training pickle files and 20500 validating pickle files ready for the generator.

– What is your optimizer? How did you tune your learning rate, learning rate decay,momentum and other parameters?

* The optimizer we are currently using is ADAM, which can be seen as the combination of RMSprop and Stochastic Gradient Descent(SDG) with momentum.

  In source (3) and (4), I summarize the following terms:

- The exponential decay rate(beta1) for first moment estimation is defaulted to 0.9. The exponential decay rate(beta2) for second moment estimation is defaulted to 0.999.

- The epsilon is defaulted to 1e-8

- The exponentially decaying average of past squared gradients v_t is initialized with a vector of 0's

- The exponentially decaying average of past gradients m_t is initialized with a vector of 0's

- Finally, the initial learning rate is 0.001 by default, and the learning rate is adapted based on the following formula in each updated weight parameters.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \rightarrow \text{(mean)},$$

$$v_t = \beta_1 v_{t-1} + (1-\beta_2) g_t^2 \rightarrow \text{(uncentered variance)},$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t},$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t},$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t,$$

where $\beta 1 = 0.9, \beta 2 = 0.999, \varepsilon = 10-8$.

– How did you make multi-step (30 step) prediction for each target agent?

Since we train the x position model and y position model separately, each model is only predicting either x position of y position.

* For MLP, we extract the p_in, v_in from x, y, and city from the testing target argent as input feature. Since both of p_in and v_in have 19 dimensions, we have total 39 dimensions in our input feature. Then shape of flattened result will be (1x39) which will be the input dimensions to our MLP model. By the mechanism of MLP, the output layer will generate the output 30 dimensions for p_out.

* For Lstm encoder-decoder, we also tried two types of experiment. For the first experiment, we turn the past 19 positions of either x or y positions into sequence of 19 x 1 as the input of the model and receive a sequence of future 30 positions as the output.

  For the second experiment, we add the input velocity of either x or y direction as a part of the input and output. Thus the input sequence become 19 x 2 and the output sequence become 30 x 2. At last, we extract the output position part as our prediction.

– How many epoch did you use? What is your batch-size? How long does it take to train your model for one epoch (going through the entire training data set once)?

* For MLP experiement 1, the epoch is 5, the batch size is 20. One epoch takes about 300s for both x and y models.

* For MLP experiemnt 2, the epoch is 50, the batch size is 20. One epoch takes about 300s for both x and y models..

* For Lstm encoder-decoder experiment 1, the epoch is 10, the batch size is 25, one epoch takes about 365 seconds for both x and y models.

* For Lstm encoder-decoder experiment 2, the epoch is 10, the batch size is 25, one epoch takes about 620 seconds for both x and y models.

5 Experiment Results

5.1 Problem A

Select a few representative models of yours and report the following results by comparing different models.

- Use a table to compare the performances of different model designs. What conclusions can you draw from this table.

  * Conclusion: MLP in experiment 2 works better comparing to other experiment models. We realize that model with more complexity doesn't necessary guarantee higher accuracy.

	Experiment_1	Experiment_2	Experiment_3	Experiment_4
Number of epoch	5	50	10	10
Training time	125 minutes	10 hrs	3650 s	6200 s
Train loss for X	5.3422	4.3841	21.6393	3.3123
Train loss for Y	5.2341	5.8433	34.1154	4.5011
Validation loss for X	4.1283	2.5161	13.3630	2.6585
Validation loss for Y	2.6652	3.2574	25.8826	4.5748
Testing loss	4.14361	3.46098	53.12467	6.39968

- Provide an estimate of training time (in flops or minutes) for different models. What did you do to improve the training speed?

  * 1. For the MLP model, the average training time of 1 epoch is about 5 minutes for both x and y model. We did increase the batch size to reduce the training time. Also, we did reduce the number of hidden neurons to decrease the training time and to reduce unnecessary weights.

  * 2. For the Lstm encoder-decoder models, the training time of each experiment is about 6 10 minutes. The sequential models iare hard to utilize parallel processing. Thus, we did increase the batch size to reduce the training time.

- Count and report the number of parameters in different models.

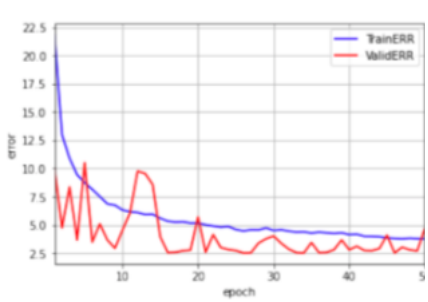
  * For experiment 1, the total number of parameters for each x and y model is 298,014(without city).
  * For experiment 2, the total number of parameters for each x and y model is 298,526(with city).
  * For experiment 3, the total number of parameter for each of x and y model is 528,641
  * For experiment 4, the total number of parameter for each of x and y model is 530,946.

5.2 Problem B

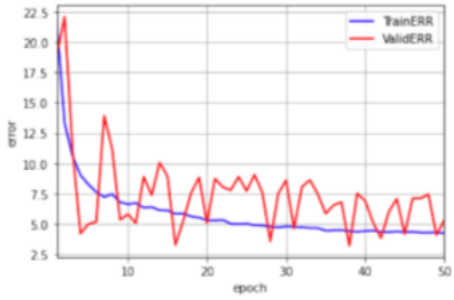
Play with different designs of your model and experiments and report the following for your best-performing design:

- Visualize the training/validation loss (RMSE) value over training steps (You should expect to see an exponential decay).

- For experiment:



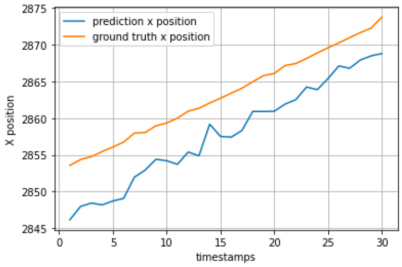
X training and validation



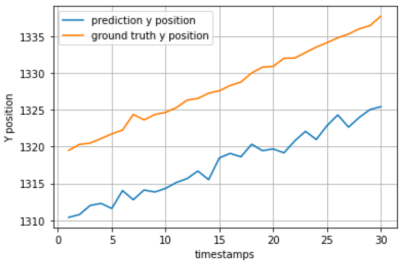
Y training and validation

- Randomly sample a few training samples after the training has finished. Visualize the ground truth and your predictions.

For sample 1:

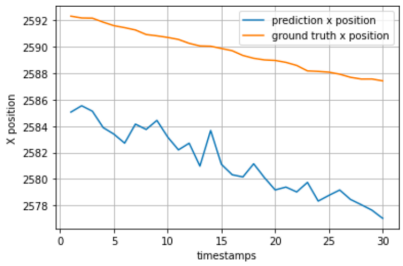


X prediction and ground truth

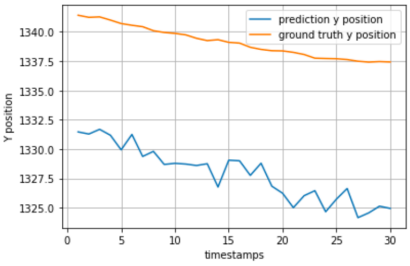


Y prediction and ground truth

For sample 2:

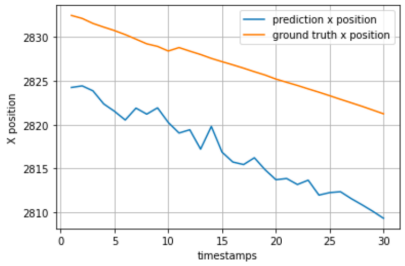


X prediction and ground truth

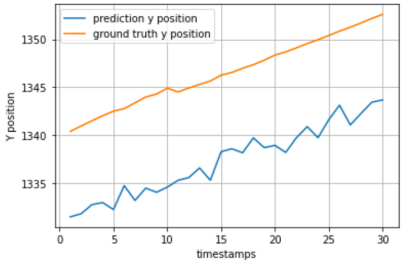


Y prediction and ground truth

For sample 3:



X prediction and ground truth



Y prediction and ground truth

- our current ranking on the leader board and your final test RMSE.
  - Our current ranking is 40th on Kaggle

6 Discussion

6.1 Problem A

Analyze the results and identify the lessons/issues that you have learned so far. Briefly answering the following questions.

- What do you think is the most effective feature engineering strategy?
  - We think the most effective feature engineering strategy is to visualize and explore the data first, then we can determine which feature is necessary.
- What techniques (data visualization/model design/hyper-parameter tuning) did you find most helpful in improving your score?
  - In the project, we find the data visualization most useful as we found out that distribution of positions x and positions y differ a lot.
- What was your biggest bottle-neck in this project?
  - The biggest bottle-neck in this project is the code implementation of the model.
- How would you advise a deep learning beginner in terms of designing deep learning models for similar prediction tasks.
  - I would advise him to do more research and reference other peoples' experience, including code implementation and model structure.
- If you had more resources, what other ideas would you like to explore?
  - If I have more resources, I would try to build a larger model that has more capacity thus we can input more feature for our prediction.

References

[1] Ming-Fang Chang, John Lambert,and Patsorn Sangkloy. *Argoverse: 3D Tracking and Forecasting with Rich Maps*.
`https://arxiv.org/abs/1911.02620`

[2] Guo Xie, Anqi Shangguan, Rong Fei, Wenjiang Ji, Weigang Ma Xinhong Hei. *Motion trajectory prediction based on a CNN-LSTM sequential model*.
`https://link.springer.com/article/10.1007/s11432-019-2761-y`

[3] SEBASTIAN RUDER. *An overview of gradient descent optimization algorithms*. https://ruder.io/optimizing-gradient-descent/

[4] Chollet, François and others. *Keras*. https://keras.io/api/optimizers/adam/