CSE151B Project Milestone: ML Noobs Group

Jeffrey Feng || Dennis Wu || TaiHei Tsin || Yujian He
A15507377 || A15537107 || A15541127 || A15386248

Abstract

This is the mile stone report for CSE151B deep learning project Autonomous vehicles(AV), which are expected to dramatically redefine the future of transportation.

1 Task Description and Exploratory Analysis

1.1 Describe in your own words what the task is and why it is important. Define the input and output in mathematical language and formulate your prediction task. Refer to week 1 lectures if you need help.

We are given the first 19 times (2 second) steps of vehicles' information to predict the next 30 time steps (3 seconds) of the corresponding target vehicles. Being able to predict the next 3 seconds of object in autonomous vehicles is important as it can avoid the car accident by auto breaking the vehicles. Moreover, it helps to keep track of traffic flow of city and prevent bad traffic.

if we only use p in as our feature in our model, we have an input matrix $M \in R^{60*19*2}$ for every scene in train/test data, and we need to output a matrix $N \in R^{60*30*2}$ including all the tracks for each scene. The final prediction for each target will be $T \in R^{30*2}$ in the test set.

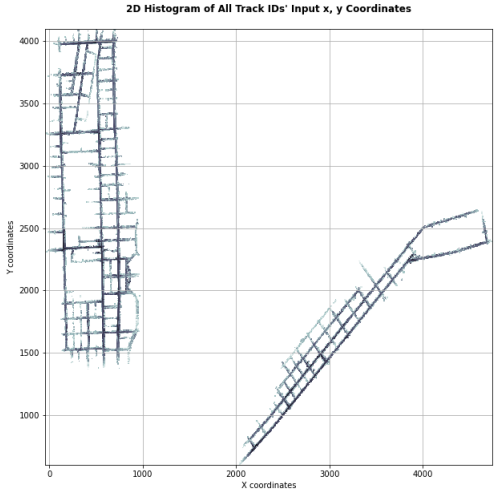
1.2 Run the provided Juypter Notebook for load the data.

1.2.1 what is the train/test data size, how many dimensions of inputs/outputs

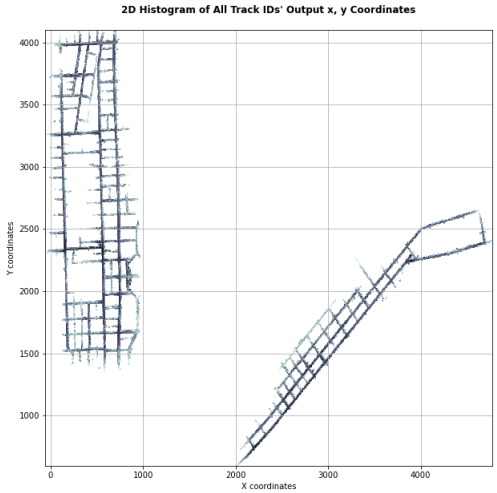
We perform train/validation split in our train data size of 80% and 20% because the given validation set has no label. Thus the train set dimensions is 164754 * 60 * 19 * 2, the validation set dimension is 41188 * 60 * 19 * 2, and the test set dimension is 3200 * 60 * 19 * 2 if we account for all the agents in the scenes.

The output may as well be 3200 * 19 * 2 as we are interested in one targeted vehicle for each scene, for the final submission of this project.

1.2.2 what is the distribution of input positions for all agents (hint: use histogram)

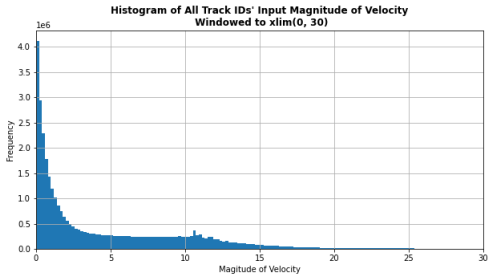


1.2.3 what is the distribution of output positions for all agents (hint: use histogram)

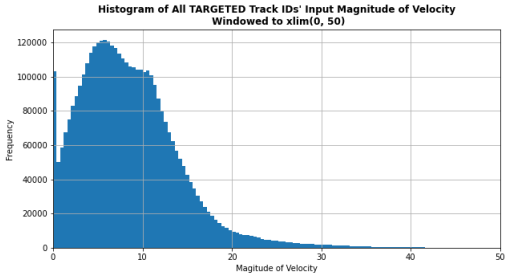


1.2.4 what is the distribution of velocity (magnitude) of all agents and the target agent

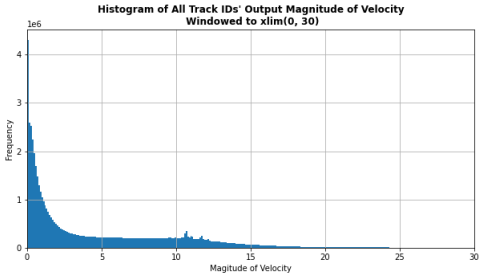
The distribution of velocity input as magnitude of all agents is: (Notes that we limit the x range to (0, 30) to show the curve of the distribution better.



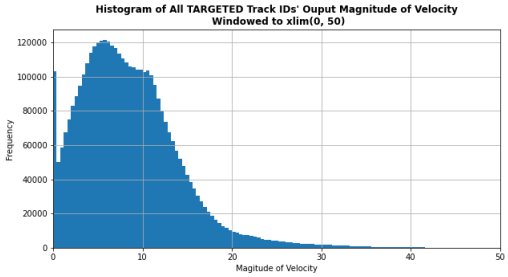
The distribution of velocity input as magnitude of just targeted agents is: (Notes that we limit the x range to (0, 50) to show the curve of the distribution better.



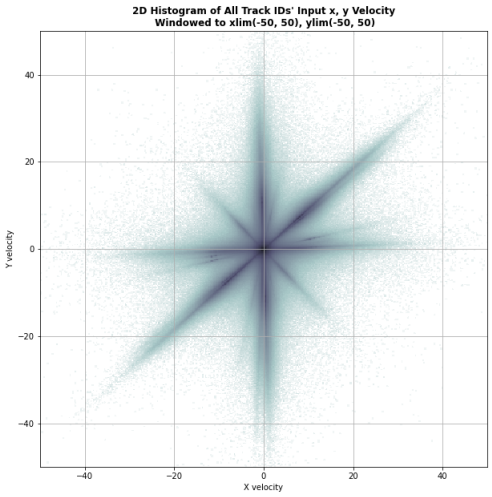
The distribution of velocity output as magnitude of all agents is: (Notes that we limit the x range to (0, 30) to show the curve of the distribution better.



The distribution of velocity output as magnitude of just targeted agents is: (Notes that we limit the x range to (0, 50) to show the curve of the distribution better.



If we do not use magnitude, but the vector of the velocity, for the input velocity of all agents, we will get visuals like the following:



2 Deep Learning Model and Experiment Design

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

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

Schools DSMLP server (datahub), we have 1 GPU, 8 CPU, and 16 G RAM
Local setup: 1 GPU, 32 Gb RAM

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

We used ADAM (torch.optim.Adam) with a learning rate of 0.001. We used two methods In order to tune our learning rate, we used a part of the training dataset as a validation set. We also set our learning rate decay as 0.0005. We wrap the ADAM optimizer in a scheduler torch.optim.lr scheduler.StepLR and set the parameter setp size to be 3. We also tried incorporating validation sets into our training, actively tuning the learning rate while training with the validation set.

2.1.3 How did you make multistep (30 step) prediction for each target agent?

We used a RNN approach. Taking advantage of LSTM, we are able to use the information of the previous layer to further predict the later frame information for 30 timesteps while updating the hidden states H and current state C.

2.1.4 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)?

We used around 10 epochs for our models. Our batch size is 128. For one epoch, our model takes around 700 seconds.

2.2 Describe the models you have tried to make predictions. You should always start with simple models (such as Linear Regression) and gradually increase the complexity of your model. Include pictures / sketch of your model architecture if that helps. You can also use mathematical equations to explain your prediction logic.

3 Experiment Results and Future Work

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

The simple baseline model that we designed is a linear regression model. We are inspired by the idea of auto regression and teacher-forcing: since we are predicting multiple time points in the future, each prediction will have a relationship with a previous prediction. We decide to create n * 2 Linear Regression models in Scikit-learn, n = number of time points that need to be predicted. Since we are predicting the next 3 seconds (30 time points), and we have both x and y coordinates to predict, we would create 60 linear regression models.

How we train and test the models are the following: for the x output predictions, we used all the 30 inputs of x coordinates as features, the first output of x coordinates in the ground truth in the train data as the label, and we generate parameters W for the first 19 x inputs to predict the first output in the test dataset. In the training set, we can always access the ground truth output of both x and y, but in the testing set, we do not have the output, thus we will use the prediction of the previous linear regression models to append to the feature matrix to create a new feature matrix with more input.

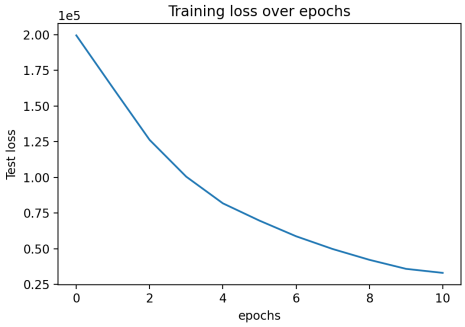
For testing the performance of different design of linear regression models, we used the validation set to show that the accuracy dramatically increased when we use the idea of auto regression: if we just predict each output point from the 19 inputs points, the accuracy for the later time points decreased, and the result is much worse than the auto-regression result.

The prediction workflows on the test set goes like this: we used the 19 input x, 19 input y, in total 38 features to predict the first x output, using the first linear regression classifier we have made from the training set. Then, we used 20 input x (19 input from the real input, and 1 output from the prediction), 19 input y, in total 39 features to predict the first y output. Then, we used 20 input x, and 20 input y, (40 features) to predict the second x output.

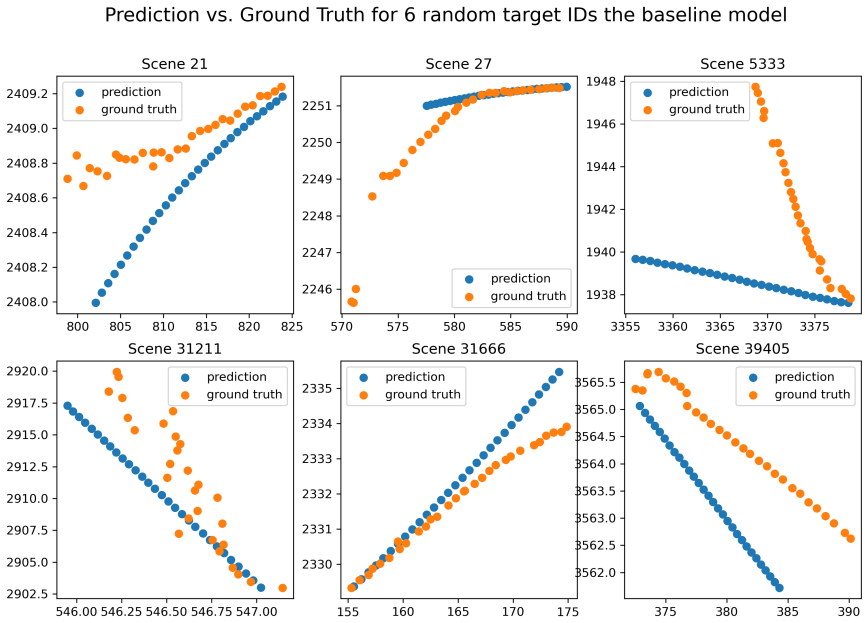
The accuracy of the model is surprisingly well on the test set, achieved leaderboard position 6 as of 5/16 with RMSE of 2.52598.

The second model we tried is a Long Short Term Memory (LSTM) model. Our logic is that the input would contain 60 * 4 features, which can be seen as the number of cars multiplied with the both parameters p_in and v_in. Then we set up the LSTM model to have 3 hidden layers and 2048 hidden dimensions. After the LSTM model, we obtain the last hidden cell state and input into a linear layer, outputting to a linear layer. Eventually we select the last 30 timesteps of the output of the linear layer and update the loss comparing to the ground truth.

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



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



3.1.3 Your current ranking on the leaderboard and your test RMSE.

Our current position on the leaderboard is 6, with RMSE of 2.52598.