# CSE 151B Milestone Report

**Robert Dunn**
rdunn@ucsd.edu
github.com/rdunnUCSD/cse151b_project

## 1 Task description and exploratory analysis

### 1.1 The task and its importance

The goal of this task is to predict three seconds of movement for an autonomous vehicle given a two second observation of the vehicle as well as other objects present in the scene. The importance of this task is a result of change in our society from manually driven vehicles to the autonomous vehicles that are being used in this task. As these vehicles are not being driven with manual input, it is necessary to the safety of the individuals both inside and outside of the vehicle to have accurate movements as to avoid collisions or other potential dangers that we currently face on the road.

We can think about this task as a function of the inputs that output our predictions. For my own model, this means that the inputs are the path of the target vehicle and the output is the predicted path of the target vehicle. Therefore, we can formulate this as the following:

$$f(x) = b$$

such that $x$ is the initial tensor path of the vehicle and $b$ is the tensor representing the predicted path. Furthermore, $f$ represents the transformation applied on $x$ in order to reshape it into $b$.

### 1.2 Exploratory analysis of the data

For this task, there are two data sets, a train set and a test set.The train set is built of 205,942 observations while the test set contains 3,200 observations. The difference between the observations in the train and test sets is that the train sets contain both an initial two second observation of the scene as well as the three following seconds, while the test set only contains the initial two second observation. Both the train and test data sets contain the following fields:

- 'city': A three character string detailing the city the observations were taken place.
- 'lane': A (N, 3) array of floats detailing the positions of the lanes in a given scene. In this field, 'N' is equal to the number of lanes which is different depending on the scene.
- 'lane_norm': A (N, 3) array of floats detailing the direction of the lanes in a given scene. In this field, 'N' is equal to the number of lanes which is different depending on the scene. 'N' in this field is always equal to 'N' in the 'lane' field.
- 'scene_idx': A nonzero integer representing the identification number for the scene.
- 'agent_id': A string that represents the identification for the autonomous vehicle.
- 'car_mask': A (60, 1) array of 1's and 0's detailing if an object is being tracked in the nth entry in which a 1 represents an object being tracked and a 0 represents no object being tracked.
- 'track_id': A (60, 30, 1) array containing the identification for all objects being tracked in the scene. For each of the 60 entries, the 30 values all contain the same identification of the given scene.

Preprint. Under review.

- 'p_in': A (60, 19, 2) array of floats detailing the x and y coordinates of each object being tracked over nineteen time intervals. Each object is identified such that the index of the object correlates to the same index in the 'track_id' field.
- 'v_in': A (60, 19, 2) array of floats detailing the x and y velocities of each object being tracked over nineteen 0.1 second time intervals. Each object is identified in the same way as 'x_in'.

The following fields are only contained in the train set:

- 'p_out': A (60, 30, 2) array of floats detailing the x and y coordinates of each object being tracked for the following thirty 0.1 second time intervals immediately after the initial observations found in 'p_in'. Each object is identified in the same way as 'x_in'.
- 'v_out': A (60, 30, 2) array of floats detailing the x and y velocities of each object being tracked for the following thirty 0.1 second time intervals immediately after the initial observations found in 'v_in'. Each object is identified in the same way as 'x_in'.

When looking at the distribution of input positions as shown in Figure 1, we can see that the inputs fall mostly into two different groupings. The first being approximately within the x-range of [0, 1000] and y-range of [1500, 4000]. The second major group is seen to be the diagonal space from the approximate coordinates of (2000, 500) to (4500, 2700). Comparing this to the distribution of output positions as shown in Figure 2, we can see that most of the data is also in the same two major groupings. From this, we can see that it is likely that most vehicles drove along either of the paths that make up these two groupings.

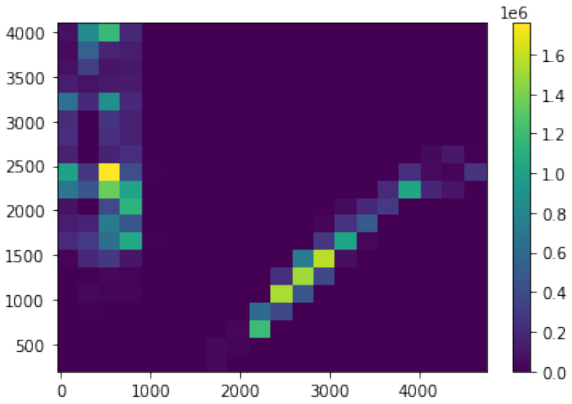


Figure 1: Distribution of input positions

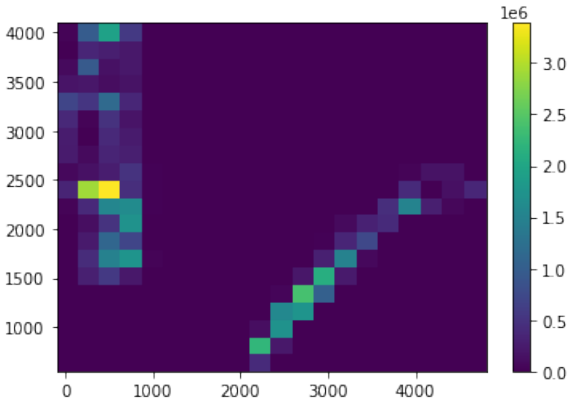


Figure 2: Distribution of output positions

On the other hand, if we look at the distribution of the magnitude of the velocities for all of the agents in the data as shown in Figure 3, we find that most of the data are contained within the x and y ranges of [-20, 20]. Therefore, if we only look at those ranges, two major details are revealed. Firstly, we can see that most of the data lies close to the point (0, 0) which means that the objects either are moving very slowly or are not moving at all. The other important detail is that the rest of the data is distributed in straight lines outwards from (0, 0) which indicates that the objects that are moving are typically moving straight along the x-axis, straight along the y-axis, or approximately at a 45 degree angle.

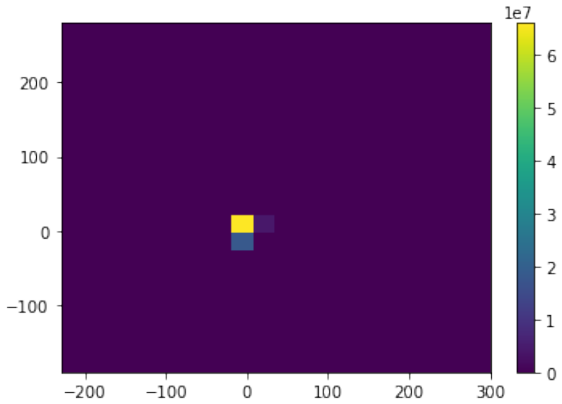


Figure 3: Distribution of magnitude of velocities

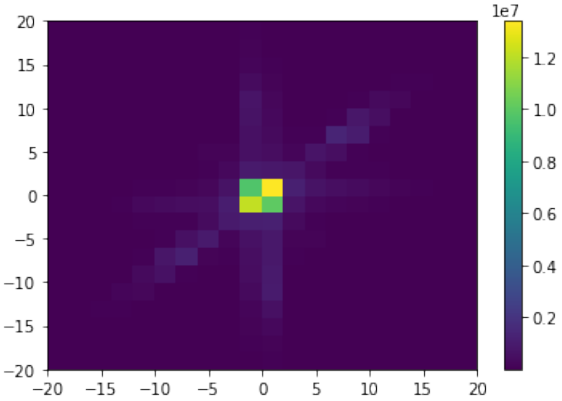


Figure 4: Inner distribution of magnitude of velocities

## 2 Deep learning model and experiment design

### 2.1 Current model design

For training my models, I both trained and tested mainly using my GPU rather than CPU because it greatly accelerated the processes. The specific GPU I am using is the NVIDIA GeForce GTX 1060.

As for the optimizer, I chose to use stochastic gradient descent. In order to tune my learning rate, I ended up doing this manually. In doing so, I would run the model on a subset of the data and furthermore adjust the learning rate based upon the results from that subset, typically by either multiplying or dividing by approximately two. As a small side note, I intend on refining this process by automating it in order to come up with a better learning rate for future models. As for other parameters, I left most of them as their defaults, except for the momentum. I used a momentum value of 0.9 for all the models I tried simply because I found it to be a recommended setting amongst the

research I had done for this task.

In order to make multistep predictions for each target, I used multiple convolutional and linear layers to reshape the input data to the desired output shape. Doing so allowed me to use the initial 19 time steps and artificially create the predicted 30 time steps.

For my model, I trained it for five epochs with a batch size of 32. Each epoch takes approximately 30 minutes to complete. As for the batch size, there is not much significance behind using 32, it is realistically only a number that I chose to have a somewhat large batch size. On the other hand, I chose five epochs for two reasons. The main reason for this is that I found that after around five to seven epochs, the model would begin over fitting on the test set. After this I chose to use only five epochs because I found that the model performance didn't improve significantly enough to justify the extra run time of one or two more epochs.

### 2.2 Previous iterations of models used

In terms of the progress of my models, I began with a simple artificial neural network with multiple convolutional and linear layers. As I continued designing newer models, the overall architecture of the model did go through significant change. The changes that were made were mostly changing certain hidden state sizes such that the new models would work with different and output sizes. Specifically, the first model I had tried using the entirety of the fields 'p_in' and 'v_in' as inputs to predicts the entirety of 'p_out' as an output. In doing such, I found my predictions to be significantly off within a validation set I created from my training set. With this in mind, I changed my model to use 'p_in' and 'v_in' as inputs to predict only the sequence for the target vehicle as an output since that is the only aspect that needs to be predicted for the task. After more testing, I found that the best working model used the specific sequence from 'p_in' corresponding to the target vehicle as an input to predict the corresponding 30 outputs positions.

As a side note, I want to add that with the information I have gathered from building these models, I believe that I have a better understanding on how to build a better model for the future with a large portion of the data that I omitted in my current working model.

## 3 Experiment results and future work

### 3.1 Results of current model

In evaluating my model, Figure 5 shows the function of the loss over the five epochs it was trained for. Although, since the model was only trained for such few epochs, the graph does not truly capture the relationship between the loss and the model training. Therefore, by looking at the loss versus number of mini batches completed, as shown in Figure 6.

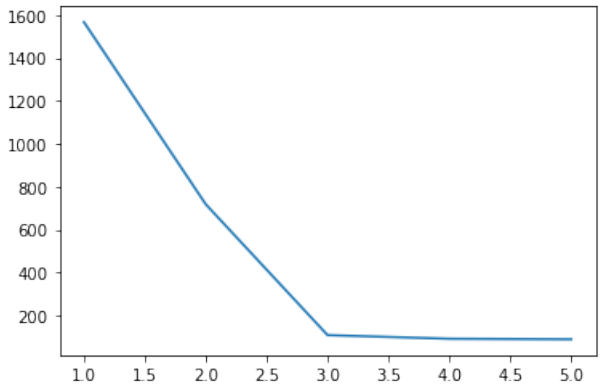


Figure 5: RSME by epoch

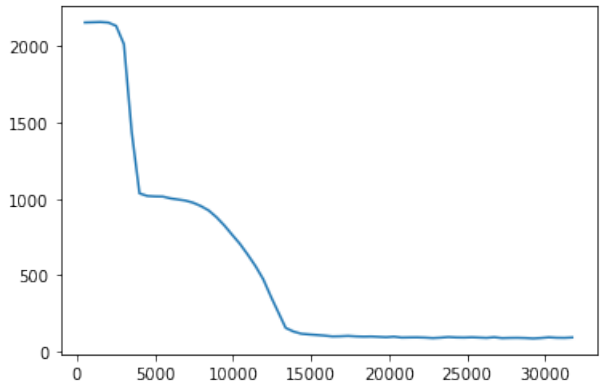


Figure 6: RSME by minibatch

In Figure 7, the actual path as well as my predicted path for 16 vehicles in the train set are shown. In each plot, 'p_in' is represented by the blue points, 'p_out' is represented by the red points, and my predictions are represented by the orange points. As it is clearly seen within these visualizations, one of the biggest downfalls of my model is that the predictions are not very accurate in terms of the movement of the target vehicle. It is very clear that my model doesn't quite predict a path for the vehicle, rather it just locates the general area where the target vehicle is.

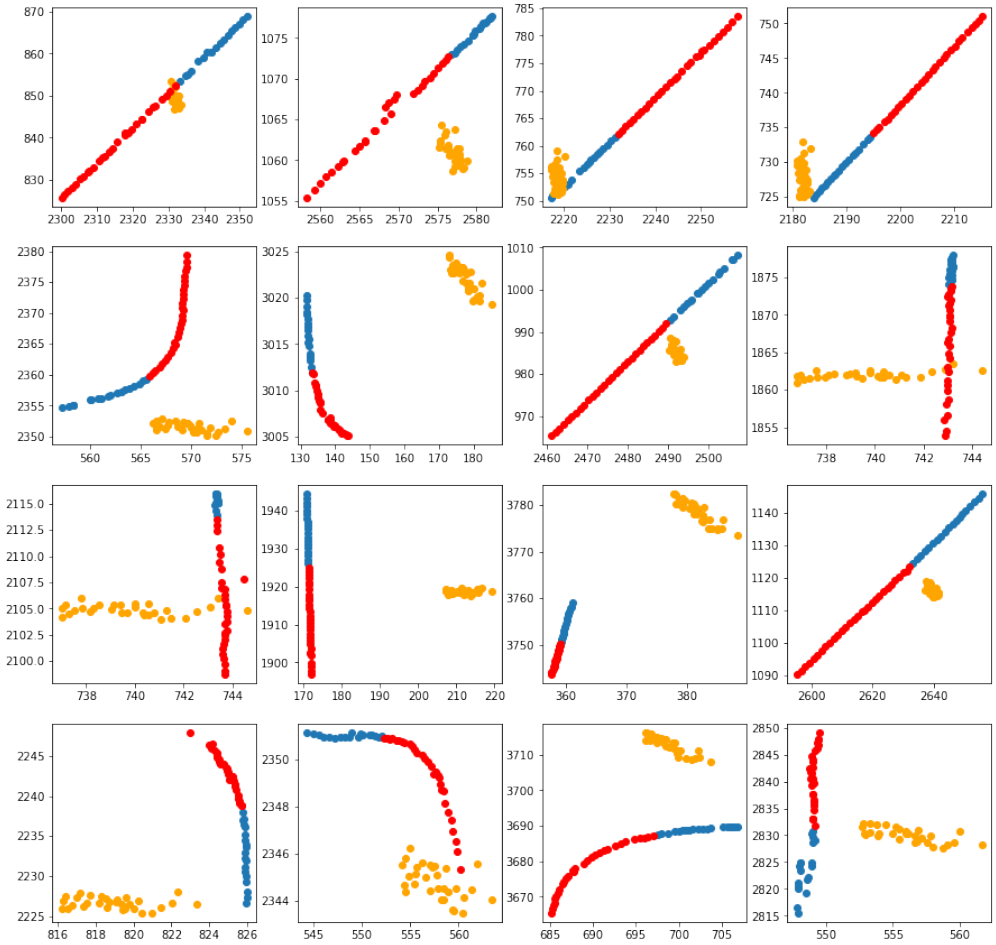

Figure 7: Actual and predicted paths for random samples

As for my overall performance on the Kaggle leader boards, I am currently ranked in 30th place with a test RSME of 16.68.

As for the future, the most helpful thing for me to do will be to work out a model that predicts an actual path rather than predicting the area that is realistically already known from the input data. I believe that a big part of the problem in my predictions stems from my input, being only the target vehicle path. While it is true that I found more success in removing input data from my model, I think that a large part in that is due to the fact that I simply became more competent with what I was doing as I created new models. As the milestone assignment states, "you should always start with simple models," and clearly I did not do a good job in following that. However, what is done is done, and now I believe I have the knowledge to go back and reintegrate parts that I omitted in order to create an overall better model.