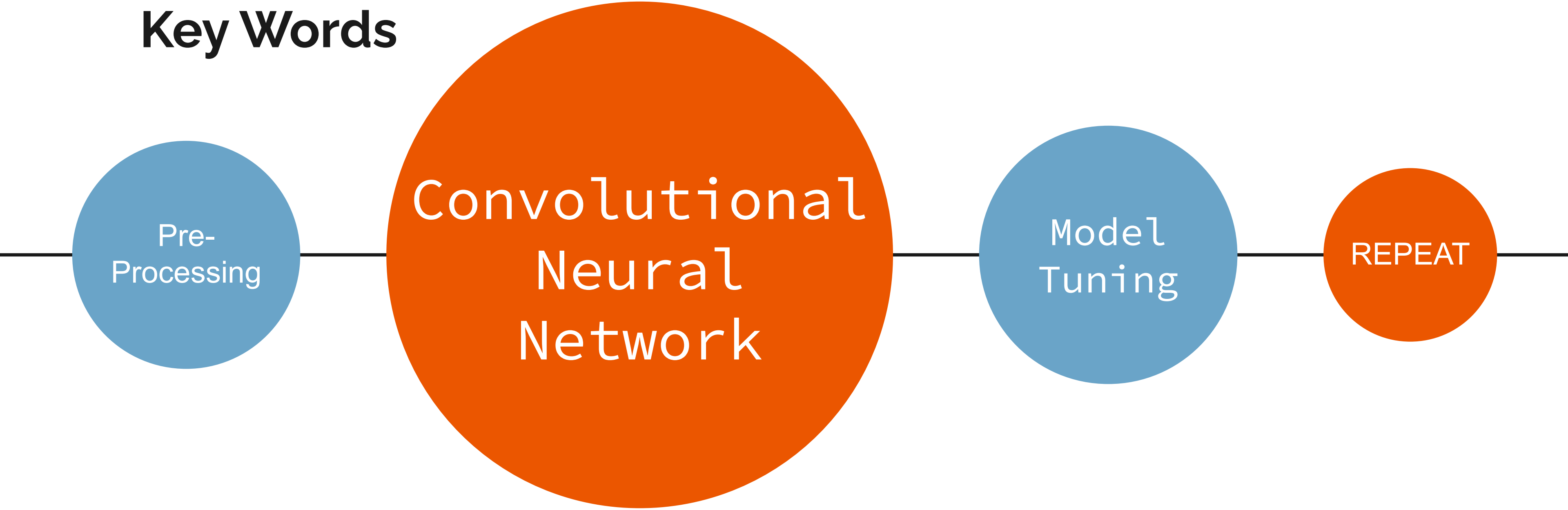
Autonomous Vehicle Predictions with a CNN

Robert Dunn

Summary

- Team Robert Dunn
- Model used is a convolutional neural network that maps the target vehicles input positions and velocities to an output sequence through a series of different layers
- Resulted with a test loss of 2.34

Key Words



Pre-Processing

Convolutional Neural Network

Model Tuning

REPEAT

Introduction

Team Introduction

- Robert Dunn
  - 3rd year Data Science Major
- That’s it, it’s just me

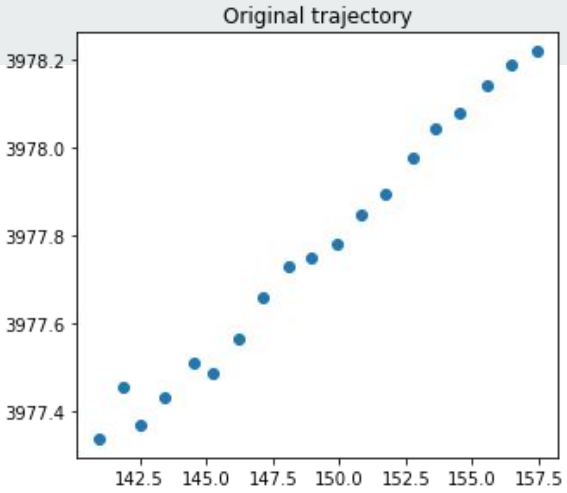
I don’t have a good picture of myself for this part so here is a picture of one of my dogs (I’ll consider him to be my team mascot).

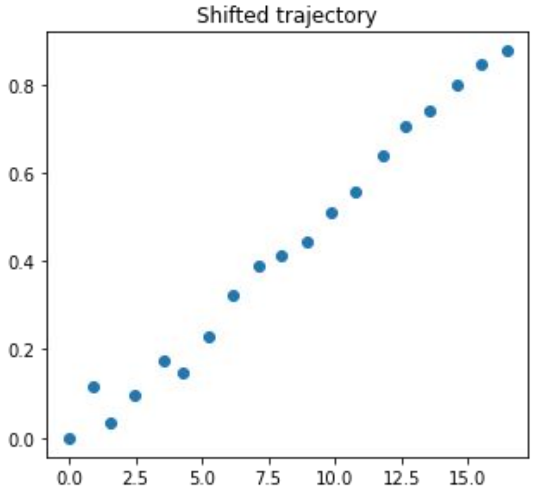
Methodology

Data Processing

- Used only the input positions and velocities from the target vehicle for simplicity

- Shifted every trajectory such that ‘p_in’ starts at (0, 0)
  - Distance between each point in the trajectory stays the same

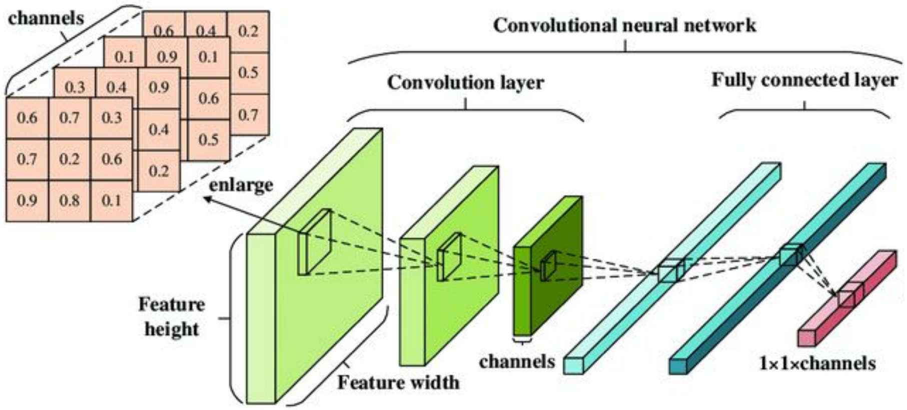

Original trajectory


Shifted trajectory

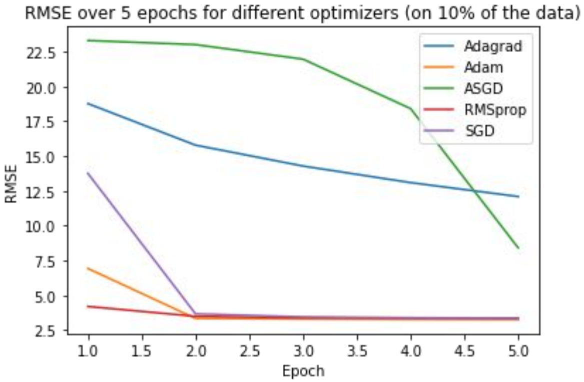
Deep Learning Model

- Convolutional Neural Network
  - Takes tensor input of size (32, 19, 2, 2) and through a series of transformations outputs a tensor of size (32, 30, 2, 1) where the first dimension (32) is the batch size

- RMSprop optimizer
  - Initially used SGD
  - changed to RMSprop for faster convergence



*Not actually what my model looks like, just an example of a CNN



← Only tested with 5 epochs to show which optimizer converges the fastest

Engineering Tricks

- During the day, I would test various different changes to my model on small portions of the data. Then I trained the best performing model on the entire dataset overnight.

- Although I didn’t actually get to doing so, it would be beneficial to have some sort of automated process for testing minor model changes,

Experiments

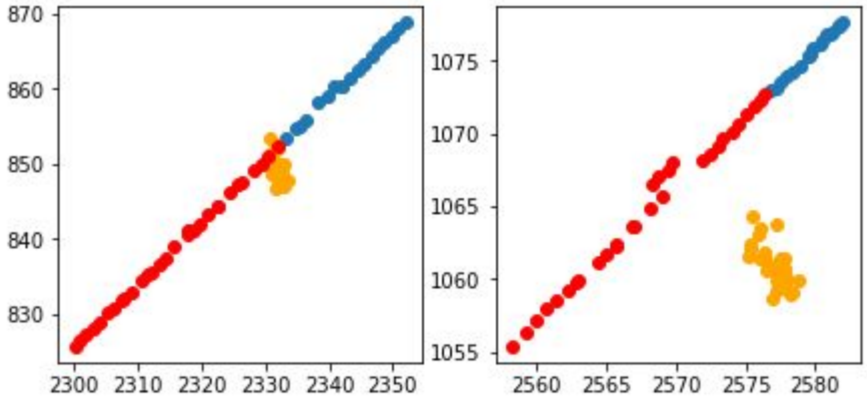
Experiment 1 - Milestone Submission

- Convolutional Neural Network
  - Seven total layers
  - no data preprocessing
  - SGD optimizer

- Predictions mainly just estimated the approximate location of the output trajectory

- RMSE of ~80 on the train set

- RMSE of 16.7 on the public test set



Example predictions on the train set

‘p_in’: blue
‘p_out’: red
predicted: orange

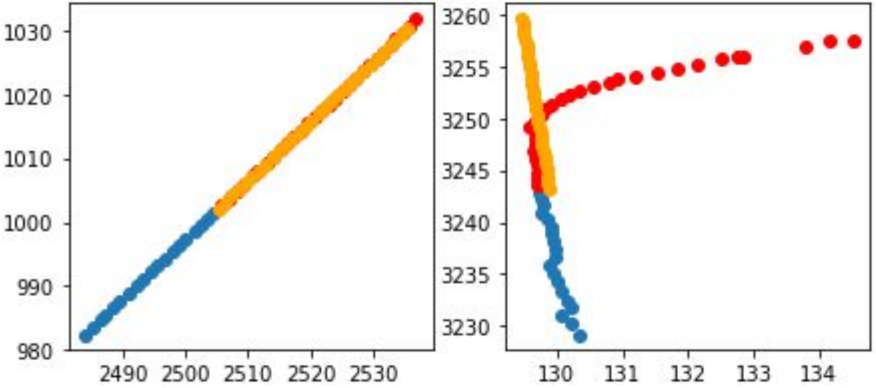
Experiment 2 - Final Submission

- Convolutional Neural Network
  - Twelve total layers
  - Shifted input trajectories to start at (0, 0) before training
  - RMSprop optimizer
  - Learning rate scheduler

- Predictions worked very good on vehicles moving in a straight line however struggled with turning vehicles/ vehicles with noisy inputs

- RMSE of ~3.2 on the train set

- RMSE of 2.34 on the public test set



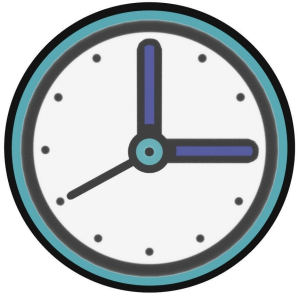
Example predictions on the train set

'p_in': blue
'p_out': red
predicted: orange

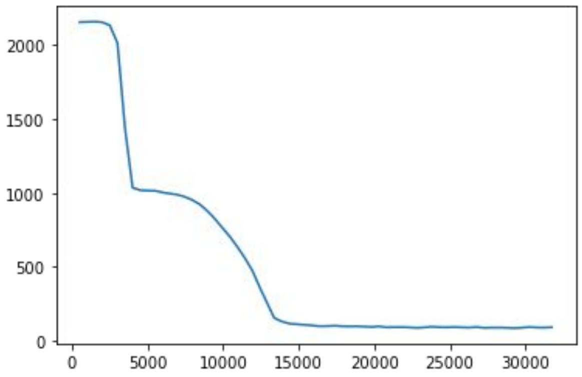
Discussion

What have you learned

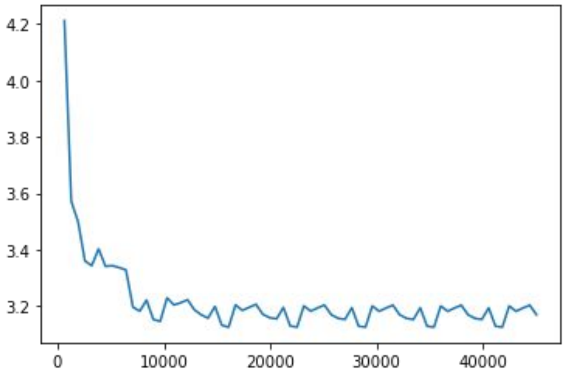
- Importance of pre-processing data
- Overall better understanding of neural networks/ pytorch in general
- START EARLY



RMSE (by minibatch) before pre-processing



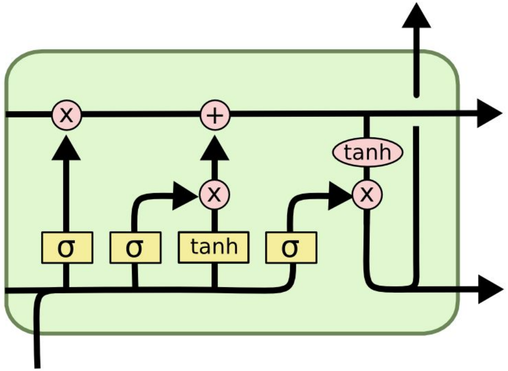
RMSE (by minibatch) after pre-processing



Future Work

- Try different models (other than CNNs)
  - i.e. LSTM or other sequence based models

- Incorporate more data into model
  - i.e. other vehicle positions and/or lane positions
  - This is most likely the biggest problem with my model as it is difficult to know if there are turns/ lane changes soon after the input if the input is completely straight. However, the model may be able to better interpret these kinds of movements based on the data that is not specifically the target vehicle


X
+
tanh
X
X
σ
σ
tanh
σ