Behavioural Cloning Project

This project uses the Udacity Simulator to acquire driving data in the form of images from the left, right, and center of the vehicle and steering angle.

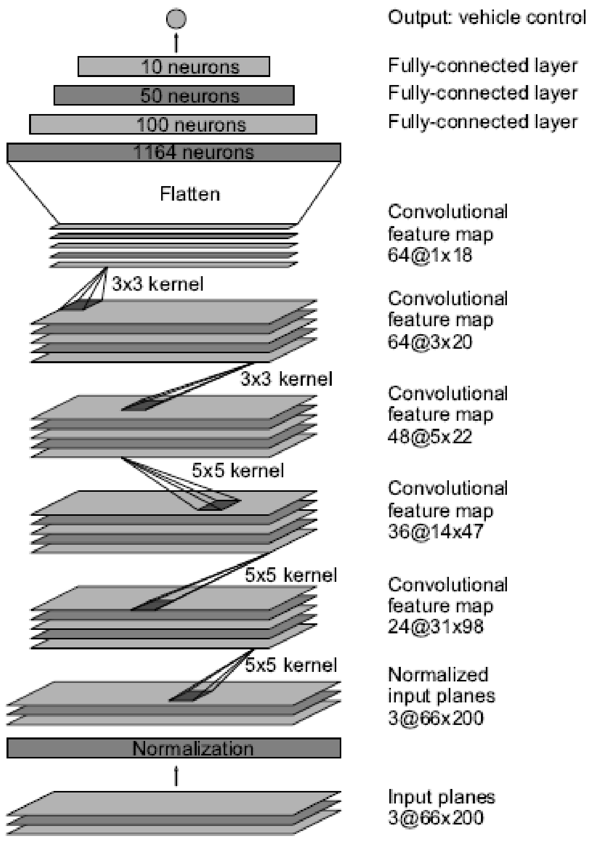
The Submitted Files include model.h5, which is the trained convolutional neural network, and model.py which is the script that creates and trains the convolutional neural network.

It also includes drive.py which is the script to run the car in autonomous mode in the Udacity simulator.

Model Architecture and Training Strategy

The Model Architecture I choose to use is the same as the Nvidia end-to-end Deep learning for vehicles.

It consists of 5 Convolutional layers and 4 Fully-Connected Layers. RELU layers were also used to introduce nonlinearity. I also included a Dropout Layer with a rate of 5% after the first fully connected layer to handle some over fitting.



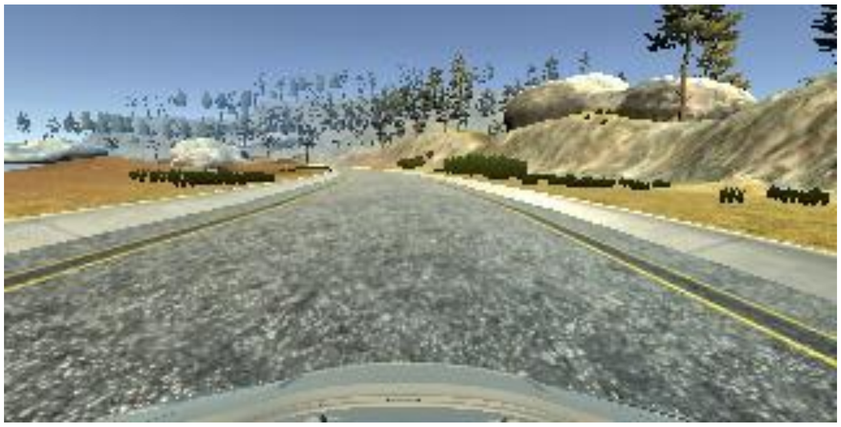
The model uses the Adam Optimizer so the learning rate was not tuned manually.

When collecting Training Data, I tried to ensure that the data represented the ability to stay in the center of the lane, recover from the sides of lanes back to the center, all while generalizing driving data to combat over-fitting to training on the same track.

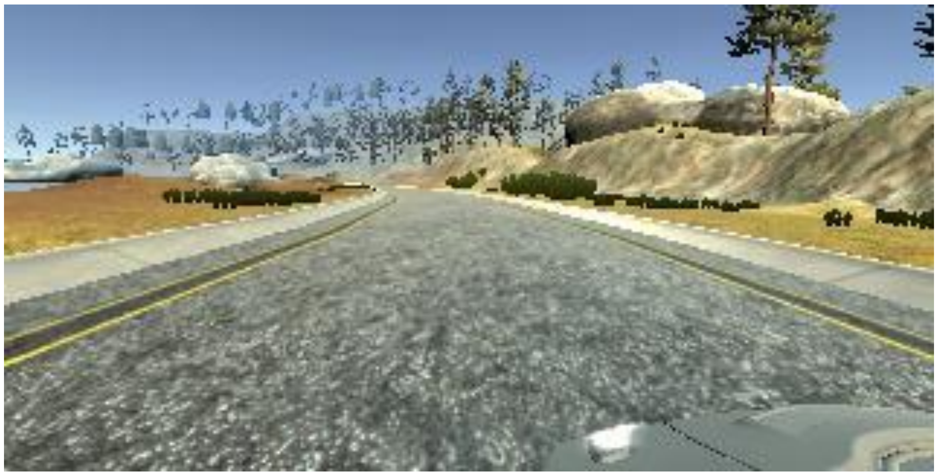
To do this, I made sure to record data when driving in the center of the road carefully for 2 laps, record data of the vehicle recovering from drifting off the center lane back to the center. Lastly to generalize the data, I made sure the record 2 laps driving in the center of the lane, around the track in the opposite direction.

Previous runs would have trouble keeping on track when the lanes disappear and there is only dirt signaling the end of the lane. To correct this issue, I went and collected more data at those specific locations.

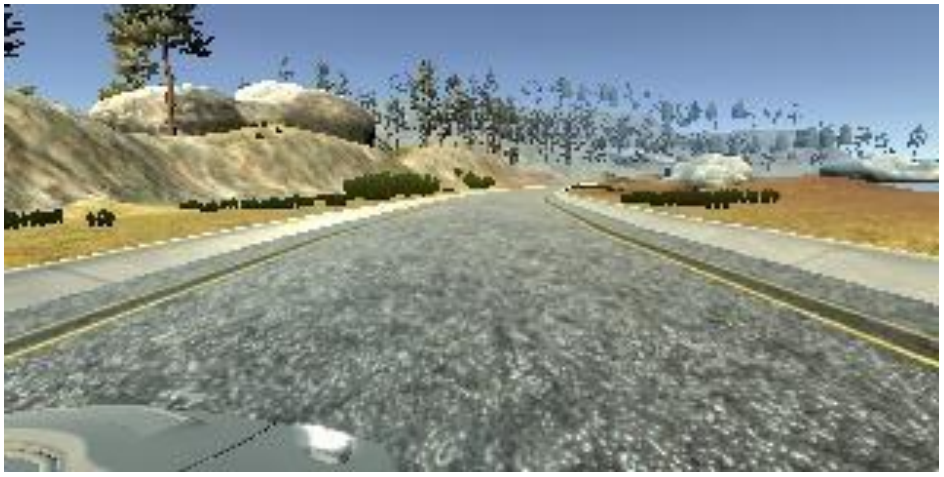
The simulator saves images that are 160x360x3 images. As part of the preprocessing, I cropped the top portion of the image that represents the distance rather than the lane, and the bottom of the image that displays the hood of the car. I also normalized the data to values between 0 and 1, with a mean around 0.5.



I also performed some data augmentation by flipping images horizontally and multiplying the steering angle by negative 1. This helped keep the data balanced when collecting data around a track that mostly involves turns in the same direction, while also acquiring more training data.

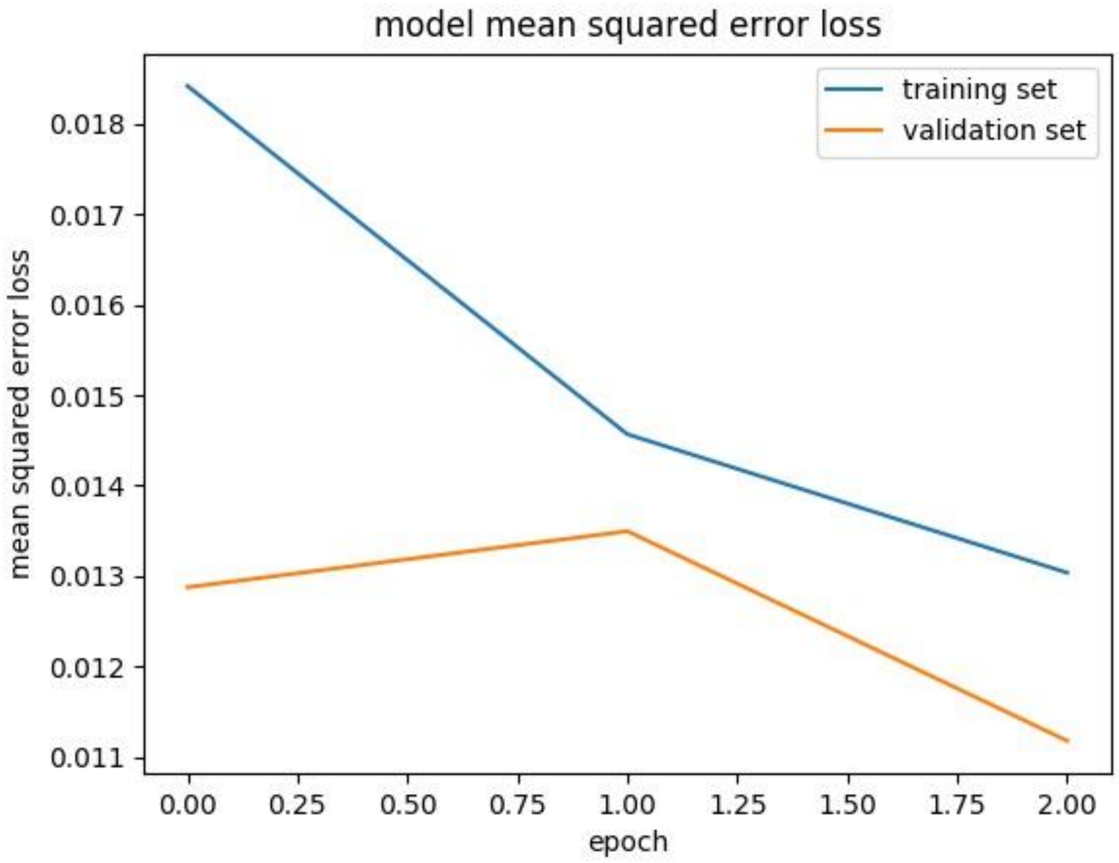


Original Image



Flipped Image

The Data acquired was shuffled and split into 80% training data and 20% validation data. The model was trained with 3 epochs.


model mean squared error loss
training set
validation set
mean squared error loss
epoch