Behavioral Cloning Project

The submitted project for behavioral cloning includes model.py, drive.py, model.h5, video.mp4 and write up pdf files.

The model.py includes the code written for processing features (images captured during manual driving of the vehicle) and code to train a Convolutional Neural Network (CNN).

The training results are stored in model.h5 file. This file is being used to drive the vehicle in the autonomous mode.

I also saved a mp4 file from captured images of vehicle while being driven in the autonomous mode. I also included a video file from my screen while the vehicle was in autonomous mode.

The combination of the simulator and drive.py file provided by Udacity do execute my model and successfully drives the vehicle around the track. To do so I used the following command:

Python drive.py model.h5

I used a CNN to train my model. In my model I used Keras library with GPU version of Tensorflow at the backend. In this Sequential model first I used Keras Lambda layer (line 72) to normalize feature set around zero with a small standard deviation. The I used crop layer (line 74) to eliminate the unwanted data in the feature set to reduce the calculation cost. Then my model is followed by three 2-dimensional convolutional layers (lines 75 through 77) with kernel size of 5X5 and stride of 2. Each of these convolutional layers are followed by a rectified linear unit layer to add nonlinearity to the model. Then I used two convolutional layers (lines 78 and 79) again this time with kernel size of 3x3 followed by Relu activation again. After using the Keras Dropout layer and Flattening the model, with using three Dense layers, an output of size 1 created to complete the model (lines 80 through 84).

This model is a regression model and since there is no classification happening here there was **no** need to use Softmax activation layers.

To compile and run this model I used Adam optimizer (line 86) to eliminate the need of optimizing training rate, and since this model is a regression model I used a Mean Squared Error method.

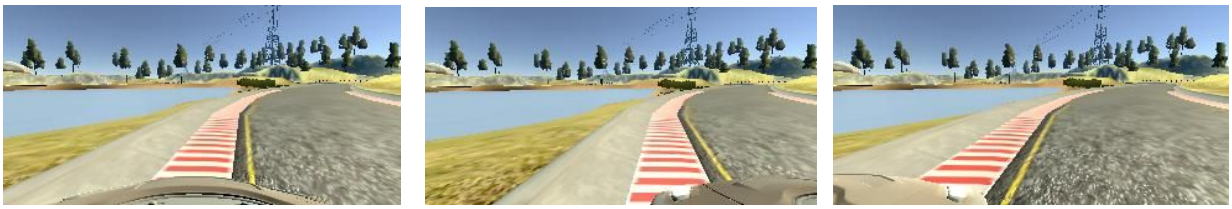
Following the recommendation of using generators to reduce the calculation time, I defined and used a generator function for processing inputs, merging images from all three cameras and assigning corrected steering angle for left and right images based on the center image, augmenting images by flipping them and creating feature and label sets (lines 26 through 62).

I shuffled the features and labels on the output of the generator function.

Sample of Counter Clockwise Driving (Center, Left, Right)

Sample of Clockwise Driving (Center, Left, Right)

Sample of Recovery at Right Turn (Center, Left, Right)



Sample of Sand Shoulder Recover (Center, Left, Right)

While I was working on this project, first I created my model without using generators. However when I started to use generators I noticed significant increase in calculation time and more optimized behavior of GPU memory. This is actually very important for me since I am using an onboard Nvidia Quadro M2000M gpu with 4GBs of RAM. This reduced the calculation time and power consumption which was significant.

For training my model first I drove the car only once counter clockwise to work with a small data set since I already included images from all three cameras and added data augmentation to increase my input size. Data augmentation (lines 53 through 57) automatically added significant

amount of data like the car was driven clockwise on the track. Running my model using the training set and validation set with one lap of driving only proved to be fairly accurate with small mean squared errors in both training and validation sets. Note: I created a validation set equal to 20% inputs.

The only problems I encountered was first the vehicle was drifting to right at the first sand shoulder on the track and second the vehicle would get off the track and drive into water at the final right turn. To remedy first I recorded recovery methods at the sand shoulder.

The second problem I encountered proved that the model was biased to the left since the track mostly has left turns, to remedy I drove the car of full lap clockwise to record data to train the model to turn right when necessary.