Term 1 project three, Behavior cloning.

Training:

The simulator contains two tracks. The goal is to train the car to be able to drive in the two track that was provided by Udacity. These two tracks are very different in nature. The background of the images and the track color are different. In addition, the curves of the two track are completely different. In order to be able to train the car and drive property, a lot of training data is needed and these data has to come from both tracks. To reduce the amount of data, we only collect a small subset of drive data on a road that does not have new information. For example, on a straight road, even we may have to drive the straight road for 5 minutes, there is no good reason to collect data for all five minutes of straight. In this case, only small subset of data will be collected.

The training data collection involve general driver such as try to teach the car stay as much in the middle of the lane as possible. Move the car from the edge of the lane back to center of the lane. And turning on the curve as well as moving the car from the edge of the curve back to the center of the curve. I collect the training data for two laps for each track, one lap clockwise, and the other lap counterclockwise.

Data distribution:

After collecting the data, it is divided into three-piece, training set, validation set, and finally, test set. 70% of the data are reserve for training. 21% of the data are reserve of validation and the rest are for testing.

Network Architecture.

The simulator collect data from three cameras and we only use the middle camera for this purpose since the goal is to clone the driving behavior on two different tracks. There are no other cars on the tracks and we do not have to worry about slowing down the car or stopping the car at all if the camera detects an object.

We first crop the top and bottom parts of the image since there are no useful information for the network to train, in our network, the top 55 rows and the bottom 20 rows of pixels are remove.

Next, we normalize the image so that the network can be trained faster. Normalization of the images in simply divide by 255 and then minus 0.5 since we want to pixel to be between -0.5 to 0.5

After the above preprocessing, we have the following:

Convolution layers

Relu

Maxpool

Convolution layers

Relu

Maxpool

Convolution layers

Relu

Maxpool

Convolution layers

Relu

Maxpool

Flatten layer

Dropout of 0.5

Fully connected with linear activation

Fully connected with Relu activation

Dropout of 0.5

Fully connected.

Finally, the network use mean square error and Adam optimizer to reduce the error. Epoch of one is used since higher value does not seem to improve the validation and test data loss.

Loading image data:

Images are load with python generator and a batch size of 32 is used. Shuffle routine is also in the generator function to randomize the data and made the trained network generalize better.