Term 1 project three, Behavior cloning.

Data collection and 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 tracks are completely different. In order to be able to train the car and drive property, a lot of training data is needed and these training data has to come from both tracks to generalize better.

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 the first track, one lap clockwise, and the other lap counterclockwise. In addition, I also collect data for second track on one lap. To generalize better under all possible driving condition, I also try to teach the network how to drive from the edge of the road back to the center of the road. This is very important since I did not change the content of the original image captured during the training such as changing the brightness of the image. The neural network see that is actually being captured from the cameras on the car. Consider the following images

Both pictures show that the car is in the curve road. However, the difference is one is at the outer part of the curve and the second one is at the inner part of the curve. I purposely collect data for curve in this scenario so that when the network is being trained, it learn how to drive from the edge of the curve (both outer and inner) back to the center of the road. The same is true for straight road but straight road impact is not as big as curve road for the reason that when the image is flip horizontally, it is symmetrical (more detail later in data generator with data augmentation section).

In addition, the road border may change on the track. Consider the following image, even it is on the outer edge of the curve, it is obviously different curve with different color. Therefore, when collecting data, I will have to collect data for all kind of road with different border.

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 of the data are for testing the model.

Data generator with data augmentation:

The simulator collect data from three cameras and we randomly use any of the three images then then we randomly flip the image. The steering angle collection adjustment for left and right images is set to 0.4 degree. So, we add 0.4 for the left camera image and substract 0.4 for the right camera image. Then we randomly flip this image.

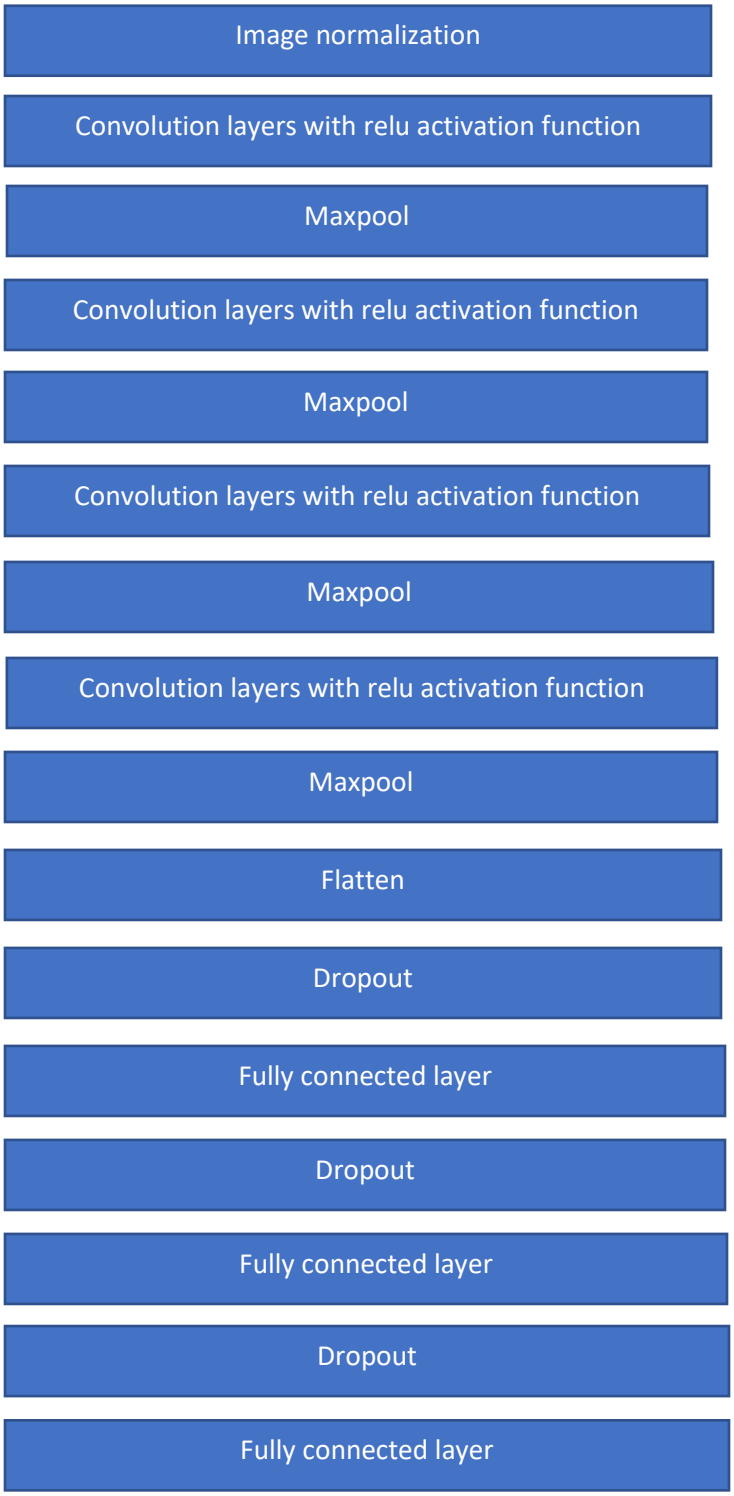
When calling the fit_generator function in keras, the steps_per_epoch is set to the length of the training sample times three. The reason for it is we want the network to train original data from all three cameras as well as the argumentation data (flipped image).

Network Architecture.

We first normalize the image data to help training faster, then we 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 network:



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.

Final Results:

I got about 0.083 of training loss and 0.064 validation loss. The car relativity stays in the middle most of the time and able to run many laps without going beyond the road boundary.

Additional Note for second submission:

My model.py use cv2 to read camera image, which is BGR format. I change the drive.py to use BGR as well and there are a lot of improvement. Even my training loss is 8.33% and validation loss is 6.4%, it performs very well in both first track and second track. I have set epoch to only one. I believe setting the epoch to something like 5 will make the actual car run much smoother on the road.