Behavioral Cloning

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road

1. Model Architecture and Training Strategy

The convolutional neural network used is very similar to the NVIDIA article for controlling the self-driving car, which consist of 5 conv layers and 4 fully connected layers. I used 5 conv and 3 fully connected layers for my network architecture:

Layer	Description
Input	160, 320, 3 RGB image
Lambda	Normalizing the input
Cropping	(70, 25), (1, 1) output 65 × 318 × 3
BatchNormalization	
Convolution 5 × 5	2 × 2 stride, same padding, output 33 × 159 × 24, Activation Relu
BatchNormalization	
Convolution 5 × 5	2 × 2 stride, same padding, output 17 × 80 × 32 Activation Relu
BatchNormalization	
Convolution 5 × 5	2 × 2 stride, same padding, output 9 × 40 × 48 Activation Relu
BatchNormalization	
Convolution 3 × 3	1 × 1 stride, same padding, output 9 × 40 × 48 Activation Relu
BatchNormalization	
Convolution 3 × 3	1 × 1 stride, same padding, output 9 × 40 × 48 Activation Relu
BatchNormalization	
Flatten	
Fully Connected	Input 23040, Output 100
BatchNormalization	
Fully Connected	Input 100, Output 50
BatchNormalization	
Fully Connected	Input 50, Output 10

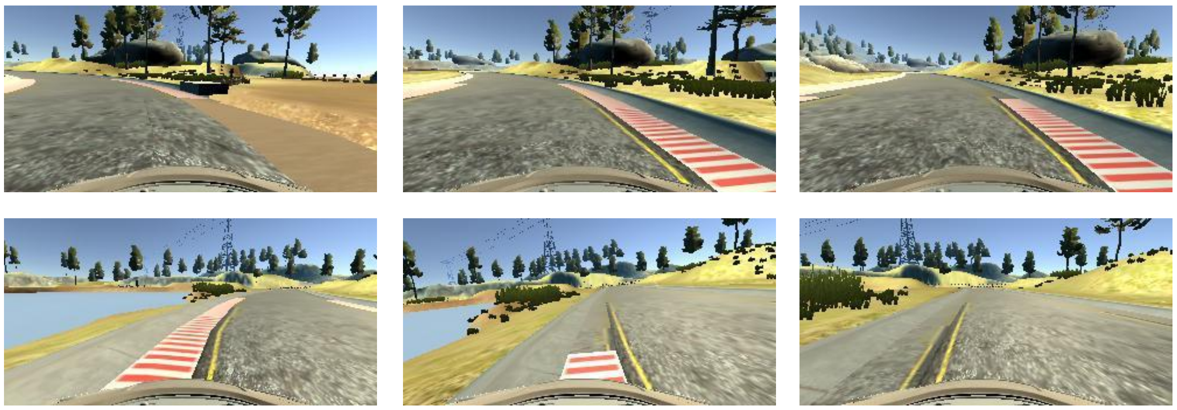
BatchNormalization	
Fully Connected	Input 10, Output 1

I realized adding BatchNormalization between the layers can improve the result a lot. It is not necessary to add the BatchNormalization after the Lambda layer at the beginning because Lambda already normalized the output. The output of fully connected layer are linear to calculate the steering angles. Also the beginning I have cropped the input image to remove the landscape and keep the road and lanes. By this, model can keep concentrated on the road. This model does not suffer from over-fitting then there is no need add any Dropout layers and as I tried, adding Dropout can reduce the validation accuracy and quality of autonomous driving.

Training the model for 6 epochs with batch size of 64 can lead to low enough validation loss of 0.0084. The model used an Adam optimizer, so the learning rate was not tuned manually.

2. Appropriate training data

The optimized model with the mentioned parameters in the previous section can achieve the validation loss of 0.102 after 6 epochs training on the primary provided dataset. I have split this dataset to train and validation set with ratio of 8 to 2. In this training I used python generator and the left and right side camera images also added to training data, with correction angle of 0.05 degree. This training can drive the car in the autonomous mode like a human driver but in some region car passes the lines, but can come back to the center of the road as show in the images below:



To improve the result I have collected additional data form simulator. To collect these data points I have consider few things:

- I connected a play station joystick to my computer and used it to control the car in the road. It can brings more accurate analog right and left steering angle values, while using keyboard to control the car can just result of -1 and 1 values.
- I was keeping the car between the lines while collecting the data, but I was trying to sometimes get close to left or right lines and come back to the center of road to show the model how can come back to the center in case it got close to the lines.
- I also drove the car in the opposite direction for 5 laps to provide the model with data that mostly has right curves instead of left curves.

These new data points have been added to the primary one, which end up having 60783 center, right and left images. Splitting this dataset to training and validation sets and train for 6 epochs can lead to validation loss of 0.0084. For this training again the python generator is used and left and right camera images also added to the training data with correction angle of 0.05 degree. As in this dataset the images with right curves are added, I realized adding flipped images does not improve the result.

3. Final result

The weights of trained model based on extended dataset are saved as model.h5, which has been used to drive the car in the autonomous mode and the video is saved as run5.mp4, shows that the simulator using the saved weights can drive the car like human driver and always keep the car between the lines and close to center of road.