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
- Summarize the results with a written report

Here I will consider the points individually and describe how I addressed each point in my implementation.

Files Submitted & Code Quality

1. Submission includes all required files and can be used to run the simulator in autonomous mode

My project includes the following files:

- model.py containing the script to create and train the model
- model.h5 file where my model is stored after training.
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
- writeup_report.md or writeup_report.pdf summarizing the results

2. Submission includes functional code

Using the Udacity provided simulator and my drive.py file, the car can be driven autonomously around the track 1 by executing

python drive.py model.h5

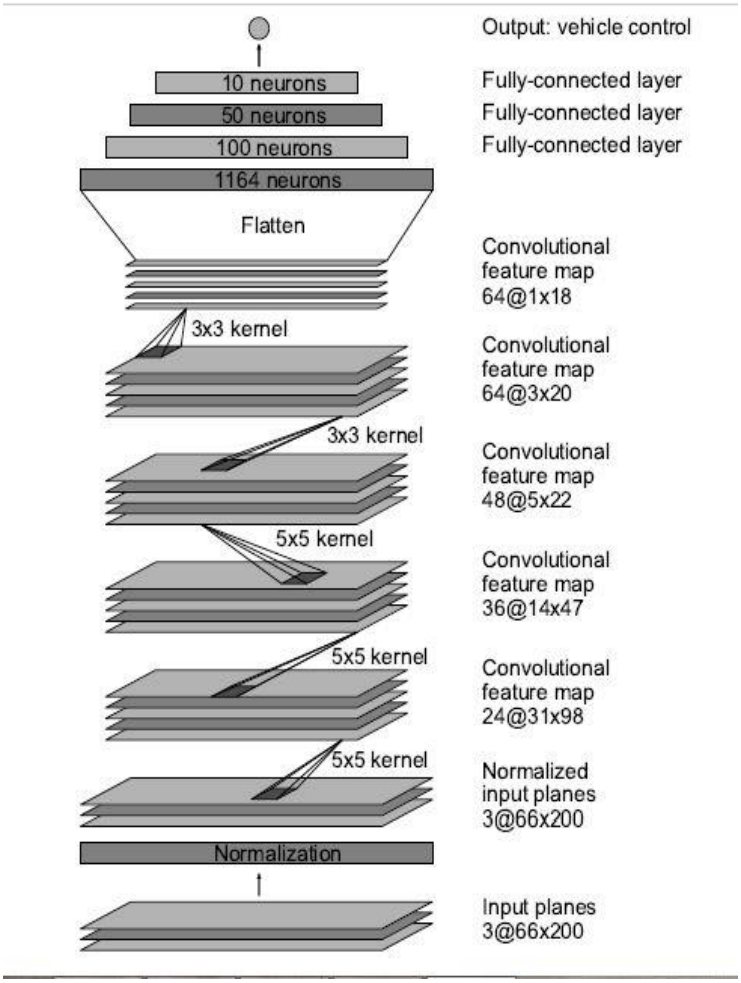
3. Submission code is usable and readable

The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

I have used the model provided in the NVIDIA research paper named as – End to End Learning for Self Driving Cars, given below




```
Layer (type)                 Output Shape              Param #     Connected to
====================================================================================================
lambda_1 (Lambda)            (None, 160, 320, 3)       0           lambda_input_2[0][0]
____________________________________________________________________________________________________
cropping2d_1 (Cropping2D)    (None, 90, 320, 3)        0           lambda_1[0][0]
____________________________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 43, 158, 24)   1824        cropping2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 20, 77, 36)    21636       convolution2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 8, 37, 48)     43248       convolution2d_2[0][0]
____________________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 6, 35, 64)     27712       convolution2d_3[0][0]
____________________________________________________________________________________________________
convolution2d_5 (Convolution2D)  (None, 4, 33, 64)     36928       convolution2d_4[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)          (None, 8448)              0           convolution2d_5[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)              (None, 100)               844900      flatten_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)              (None, 50)                5050        dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)              (None, 10)                510         dense_2[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)              (None, 1)                 11          dense_3[0][0]
====================================================================================================
Total params: 981,819
Trainable params: 981,819
Non-trainable params: 0
```

The model consists of 5 convolutional layers where the first three are having 5x5 as kernel size and the remaining 2 has 3x3 kernel size. This can be found in the lines 116 to 120 in the model.py file. Each convolutional layer is activated by ReLU function to provide the adequate non linearity in the model.

The data is normalized in the model using a Keras lambda layer (code line 115). After this the output of CNN layers are flattened and then 4 dense layers are added with the last one having the output 1 because this model predicts the steering angles which is a regression task.

2. Attempts to reduce overfitting in the model

The model was trained and validated on a large data set provided by the Udacity team only to ensure that the model was not overfitting. The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

3. Model parameter tuning

My model used an Adam optimizer, so the learning rate was not tuned manually (model.py line 149).

4. Appropriate training data

Training data was already provided by the Udacity team and I used their data to train my model. The data contained 8037 images of center, left and right cameras mounted on the car.

Model Architecture and Training Strategy

1. Solution Design Approach

My first step was to use a convolution neural network model similar to the model given in the above mentioned NVIDIA paper as I carefully studied that the team at NVIDIA was able to achieve fantastic results with their model.

I applied data augmentation technique of flipping the images as the track 1 was mostly having left turns which would have made my model more biased towards the left turns. This data augmentation method helped to increase the size of training set from 8037 to 24108.

I finally shuffled the data set and put 20% of the data into a validation set.

I got total 24108 number of data points after data augmentation, out of which 19286 were the training samples and 4822 were the validation samples. Since, I got a large number of samples to train on, so overfitting didn’t occurred in my model and I verified it by training my model without dropout layers and my model worked very well on the simulator in autonomous mode.

The final step was to run the simulator to see how well the car was driving around track one.

At the end of the process, the vehicle was able to drive autonomously around the track without leaving the road.

2. Creation of the Training Set & Training Process

I didn't collected any driving samples and I used the driving samples already given by the Udacity team which had around 8037 images.

Here is an example image of center lane driving:

Left Image

Right Image

To augment the data set, I also flipped images as shown

I performed the training process using the Adam optimizer with the MSE as loss function. I was able to get low loss on both the training and validation set and my model didn’t overfitted as there was large number of data generated using the data augmentation method. My model was trained on 3 epochs.