Behavior Cloning Project

The 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

My project includes the following files:

- model.py containing the script to create and train the model
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
- run1.MP4 video of the car driving in autonomous mode based on the trained model
- writeup_report.pdf summarizing the results

1. Submission includes functional code

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

```
python drive.py model.h5
```

In the autonomous mode, the car was keeping on the track all the time.

2. 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, and it contains comments to explain how the code works.

3. Model Architecture and Training Strategy

The model architecture is based on the Nvidia CNN architecture (shown in figure below)

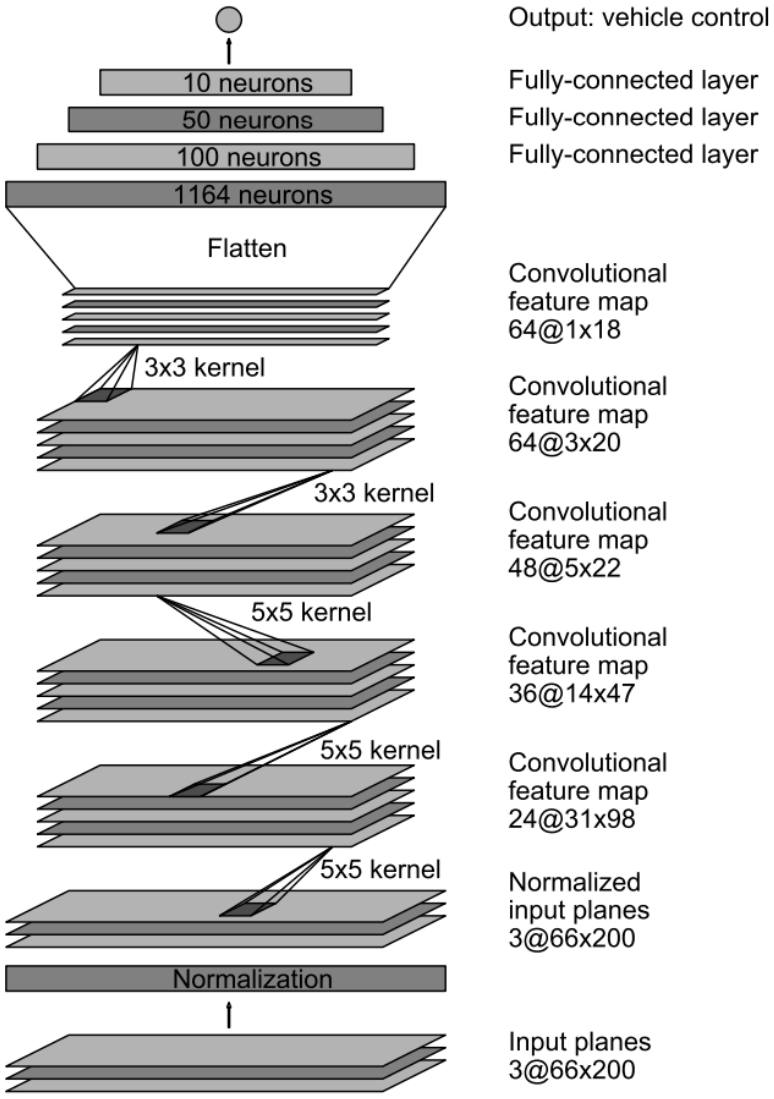

Figure 1 Nvidia CNN architecture

My model consists of 1 Lambda layer to normalize the data, 1 Cropping layer to remove the useless data,5 convolution layers, 1 flatten layer and 4 direct connect layers. The table below is a summary of the architecture of my model.

Layer	Output Shape
Lambda	(160,320,3)
Cropping2D	(65,320,3)
Convolution2D	(65,320,24)
Convolution2D	(65,320,36)
Convolution2D	(65,320,48)
Convolution2D	(65,320,64)
Convolution2D	(65,320,64)

Flatten	1331200
Dense	100
Dense	50
Dense	10
Dense	1

3.1 Attempts to reduce overfitting in the model

The split of test data and validation by a ratio of 4:1 was used to reduce the over fitting. The model was trained and validated on different data sets to ensure that the model was not overfitting.

3.2 Model parameter tuning

The parameter I used to tune the model is the steering correction. Initially, I used 0.2 for the steering correction and trained the model with the data I collected. However, in the autonomous mode, the car went off track. That was because the initial steering correction was not big enough to pull the car back when it was off center. So I tried to increase the steering correction to 0.35 and the car was successfully driving along the center of the track.

3.3 Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving for 3 laps in the counter-clockwise direction and 2 laps in clockwise direction, 1 lap focusing on recovering from the left and right sides of the road and 1 lap focusing on driving smoothly at the corners. The data flow of my model is: 1. The collected data was randomly split into train sample and validation sample, 80% and 20% respectively. When the train data and validation data was provided to the model through a generator. In the generator, the images were flipped to reduce the impact of driving direction. The figures below show the original image and the flipped image.

Figure 2 Original image

Figure 3 Flipped image

The figure below shows the mean square error of the model is reduced from first epoch. As is shown, after 3 epochs, the MSE of the training set did not change much.