Behavioral Cloning Project

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

Rubric Points

Here I will consider the rubric 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
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
- writeup_report.pdf summarizing the results

2. Submission includes functional code

Using the Udacity provided simulator and drive.py file, the car can be driven autonomously around the track 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, and it contains comments to explain how the code works.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

My model consists of a convolution neural network with 5x5 filter sizes a (model.py lines 85-91)

The model includes RELU layers to introduce nonlinearity (code line 85, 91), and the data is normalized in the model using a Keras lambda layer (code line 78).

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 98, 102, 106).

The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 112). The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track1.

3. Model parameter tuning
The model used an adam optimizer, so the learning rate was not tuned manually (model.py line 111).

4. Appropriate training data
Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving, recovering from the left and right sides of the road ...
For details about how I created the training data, see the next section.

Model Architecture and Training Strategy

1. Solution Design Approach
The overall strategy for deriving a model architecture was to feed camera images and steering angles to CNN.
My first step was to use a convolution neural network model similar to the lenet I thought this model might be appropriate because this was the most simplest network for the job.
In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set. I found that my first model had a low mean squared error on the training set but a high mean squared error on the validation set. This implied that the model was overfitting.
To combat the overfitting, I modified the model so that it uses dropout layer in model
Then I also flipped the images and and multiplied by -ve angles so CNN sees both kind of data.
The final step was to run the simulator to see how well the car was driving around track one. There were a few spots where the vehicle fell off the track, to improve the driving behavior in these cases, I recorded car moving from side of the road to the centre of road.
At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

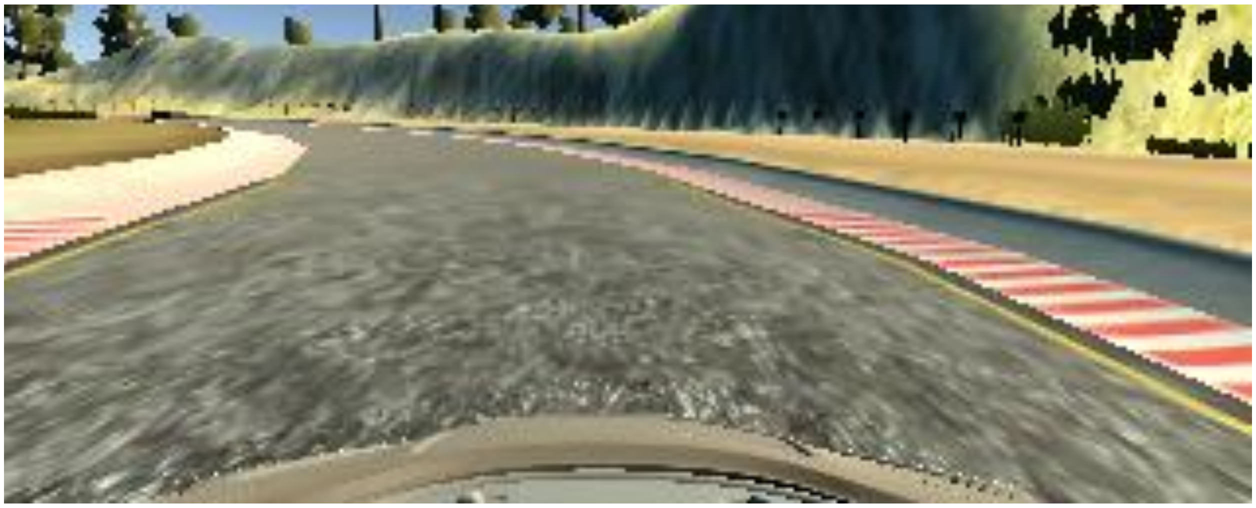
2. Final Model Architecture
The final model architecture (model.py lines 74-109) consisted of a convolution neural network with the following layers and layer sizes as follows

```
____________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================
lambda_1 (Lambda)                (None, 160, 320, 3)   0           lambda_input_1[0]
[0]
____________________________________________________________________________________
cropping2d_1 (Cropping2D)        (None, 85, 320, 3)    0           lambda_1[0][0]
____________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 81, 316, 6)    456         cropping2d_1[0][0]
____________________________________________________________________________________
```

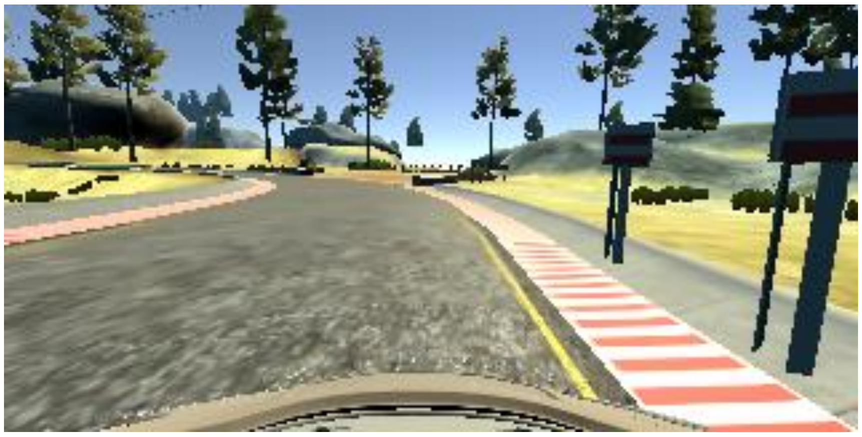
```
maxpooling2d_1 (MaxPooling2D)    (None, 40, 158, 6)    0           convolution2d_1[0]
[0]
____________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 36, 154, 6)    906         maxpooling2d_1[0]
[0]
____________________________________________________________________________________
maxpooling2d_2 (MaxPooling2D)    (None, 18, 77, 6)     0           convolution2d_2[0]
[0]
____________________________________________________________________________________
flatten_1 (Flatten)              (None, 8316)          0           maxpooling2d_2[0]
[0]
____________________________________________________________________________________
dropout_1 (Dropout)              (None, 8316)          0           flatten_1[0][0]
____________________________________________________________________________________
dense_1 (Dense)                  (None, 120)           998040      dropout_1[0][0]
____________________________________________________________________________________
dropout_2 (Dropout)              (None, 120)           0           dense_1[0][0]
____________________________________________________________________________________
dense_2 (Dense)                  (None, 84)            10164       dropout_2[0][0]
____________________________________________________________________________________
dropout_3 (Dropout)              (None, 84)            0           dense_2[0][0]
____________________________________________________________________________________
dense_3 (Dense)                  (None, 1)             85          dropout_3[0][0]
====================================================================================
==============
Total params: 1,009,651
Trainable params: 1,009,651
Non-trainable params: 0
```

Here is a visualization of the architecture (note: visualizing the architecture is optional according to the project rubric)

3. Creation of the Training Set & Training Process
To capture good driving behavior, I first recorded around two laps on track one using center lane driving. Here is an example image of center lane driving:



I then recorded the vehicle recovering from the left side and right sides of the road back to center so that the vehicle would learn to .... These images show what a recovery looks like starting from right side:



After the collection process, I had 3915 number of data points.
I finally randomly shuffled the data set and put 25% of the data into a validation set.
I used this training data for training the model. The validation set helped determine if the model was over or under fitting. The ideal number of epochs was 4. I used an adam optimizer so that manually training the learning rate wasn't necessary.