Behavioral Cloning Project

Yi Zhu

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

###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.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 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

The main structure is based on the neural network constructed by Nvidia's end to end learning paper.

My model consists of four convolution neural network with 5x5 or 3x3 filter sizes and depths between 24 and 64

The model includes RELU layers to introduce nonlinearity (code line 41), and the data is normalized in the model using a Keras lambda layer (code line 39).

Also there is a layer in model cropping the image to let network focus on what is happening right ahead of the car.

####2. Attempts to reduce overfitting in the model

Model includes a dropout layer (line 47) to avoid overfitting.
Moreover, the model is trained and validated using different data (splitted) to ensure there is less overfitting. However, since most of data is draw from first track. Lack of diversity of data source might result in overfitting when applied to other tracks.

####3. Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually (model.py line 54).

####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. No fancy tricks to augment more data, just run the training in simulator enough times including different scenarios(like near left or right edge of road) to make training accurate. In my opinion, having enough real data with enough diversity is the most important factor to improve generalization accuracy.

Here attach two pictures of my training data.