~~~~
#**Behavioral Cloning**

##Writeup Template

###You can use this file as a template for your writeup if you want to
submit it as a markdown file, but feel free to use some other method and
submit a pdf if you prefer.

---

**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


[//]: # (Image References)

[image1]: ./examples/placeholder.png "Model Visualization"
[image2]: ./examples/placeholder.png "Grayscaling"
[image3]: ./examples/placeholder_small.png "Recovery Image"
[image4]: ./examples/placeholder_small.png "Recovery Image"
[image5]: ./examples/placeholder_small.png "Recovery Image"
[image6]: ./examples/placeholder_small.png "Normal Image"
[image7]: ./examples/placeholder_small.png "Flipped Image"

## Rubric Points
###Here I will consider the [rubric
points](https://review.udacity.com/#!/rubrics/432/view) 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.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
```sh
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 Nvidia model convolution neural network. The model includes RELU layers to introduce nonlinearity.

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

The model was trained and validated on different data sets to ensure that the model was not overfitting. The training-validation data was spilt as 80-20%. I kept the epochs to 7. The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

####3. Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually.

####4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used the data provided by Udacity.  I used some correction for steering angles and flipped images to create more versatile data for training.

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 use nVidia model for CNN.

It is a simple yet effective model. 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 tried introducing lambda layer and cropping, but there were errors while creating model. I decided to try simplest approach without extra layers.

I tried with epoch of 3 but when the simulator ran autonomously, it worked well for most of the lap but at one point it went off track when the dusty road came on the right. I then trained with epoch = 7 and this
```

time it worked well. Validation loss kept on decreasing with the epochs, which was a good sign.

The final step was to run the simulator to see how well the car was driving around track one. It successfully completed whole lap and with joy, I let it run little more just to ensure it didn't behave differently further down the line.

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

Below is the output on my AWS instance which created model.h5

```
carnd@ip-172-31-86-73:~$ python model.py
Using TensorFlow backend.
@@@@@@@@@@@@@@@@@@@
*****************
19286
Epoch 1/7
19200/19286 [============================>.] - ETA: 0s - loss: 1.0369/
19456/19286 [==============================] - 88s - loss: 1.0236 -
val_loss: 0.0228
Epoch 2/7
19372/19286 [==============================] - 88s - loss: 0.0281 -
val_loss: 0.0270
Epoch 3/7
19456/19286 [==============================] - 88s - loss: 0.0220 -
val_loss: 0.0210
Epoch 4/7
19372/19286 [==============================] - 88s - loss: 0.0205 -
val_loss: 0.0196
Epoch 5/7
19456/19286 [==============================] - 88s - loss: 0.0175 -
val_loss: 0.0157
Epoch 6/7
19372/19286 [==============================] - 87s - loss: 0.0144 -
val_loss: 0.0147
Epoch 7/7
19456/19286 [==============================] - 88s - loss: 0.0129 -
val_loss: 0.0142
```

####2. Final Model Architecture

The final model architecture) consisted of a convolution neural network with the following layers and layer sizes

1. Convolution2D with kernel (5,5) , activation: relu, strides(2,2), filters:24
2. Convolution2D with kernel (5,5) , activation: relu, strides(2,2), filters:36
3. Convolution2D with kernel (5,5) , activation: relu, strides(2,2), filters:48
4. Convolution2D with kernel (3,3) , activation: relu, strides(1,1), filters:64

5. Convolution2D with kernel (3,3) , activation: relu, strides(1,1), filters:64
6. Flatten
7. Dense with output:100
8. Dense with output:50
9. Dense with output:10
10. Dense with output:1

(note: visualizing the architecture is optional according to the project rubric)

![alt text][image1]

####3. Creation of the Training Set & Training Process

I used the data provided by Udacity, which really helped me.

To augment the data set, I also flipped images and corrected steering angles

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 7 I used an adam optimizer so that manually training the learning rate wasn't necessary.