# Behavioral Cloning

## Project Writeup

**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.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, 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 contains the following architecture.  I got to this point first by researching existing neural networks, even some of those mentioned in the classes (googlenet, etc.).  Most specifically I found the NVIDIA video/article to be the most helpful and best starting point.  I found this to be a good compromise between a good network and computation time.  I found that very quickly computation time can get out of control and totally eliminate any practicality from this toolset.  As far as the parameters (dense value, convolution sizes, etc.) that was mostly a trial and error process but I did borrow some concepts from the class and existing networks.

```
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
cropping2d_1 (Cropping2D)        (None, 90, 320, 3)    0           cropping2d_input_1[0][0]
____________________________________________________________________________________________________
lambda_1 (Lambda)                (None, 90, 320, 3)    0           cropping2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 88, 318, 20)   560         lambda_1[0][0]
____________________________________________________________________________________________________
activation_1 (Activation)        (None, 88, 318, 20)   0           convolution2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 86, 316, 10)   1810        activation_1[0][0]
____________________________________________________________________________________________________
```

```
activation_2 (Activation)        (None, 86, 316, 10)   0          convolution2d_2[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 84, 314, 5)    455        activation_2[0][0]
____________________________________________________________________________________________________
activation_3 (Activation)        (None, 84, 314, 5)    0          convolution2d_3[0][0]
____________________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 82, 312, 2)    92         activation_3[0][0]
____________________________________________________________________________________________________
maxpooling2d_1 (MaxPooling2D)    (None, 41, 156, 2)    0          convolution2d_4[0][0]
____________________________________________________________________________________________________
convolution2d_5 (Convolution2D)  (None, 39, 154, 10)   190        maxpooling2d_1[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 39, 154, 10)   0          convolution2d_5[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 60060)         0          dropout_1[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 16)            960976     flatten_1[0][0]
____________________________________________________________________________________________________
activation_4 (Activation)        (None, 16)            0          dense_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 16)            272        activation_4[0][0]
____________________________________________________________________________________________________
activation_5 (Activation)        (None, 16)            0          dense_2[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 16)            272        activation_5[0][0]
____________________________________________________________________________________________________
activation_6 (Activation)        (None, 16)            0          dense_3[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 16)            272        activation_6[0][0]
____________________________________________________________________________________________________
activation_7 (Activation)        (None, 16)            0          dense_4[0][0]
____________________________________________________________________________________________________
dense_5 (Dense)                  (None, 16)            272        activation_7[0][0]
____________________________________________________________________________________________________
activation_8 (Activation)        (None, 16)            0          dense_5[0][0]
____________________________________________________________________________________________________
dense_6 (Dense)                  (None, 1)             17         activation_8[0][0]
====================================================================================================
```

## 2. Attempts to reduce overfitting in the model

As you can see there are two layers designed to mitigate overfitting.  The maxpooling and dropout layers.  I also included more data than udacity provided.  I recorded two types of data, recovery and smooth curve tracking.  Additionally it should be noted a shuffle function from sklearn is used before the model to avoid any issues with overfitting and the data being in order.

## 3. Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually.  I found this to be a good thing because it was one less hyper parameter to be concerned about.

## 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 and smooth curve tracking.  I chose the later two because they were issues I found with my earlier versions.  In short whatever I had issues with I collected more data on.

## Model Architecture and Training Strategy

### 1. Solution Design Approach

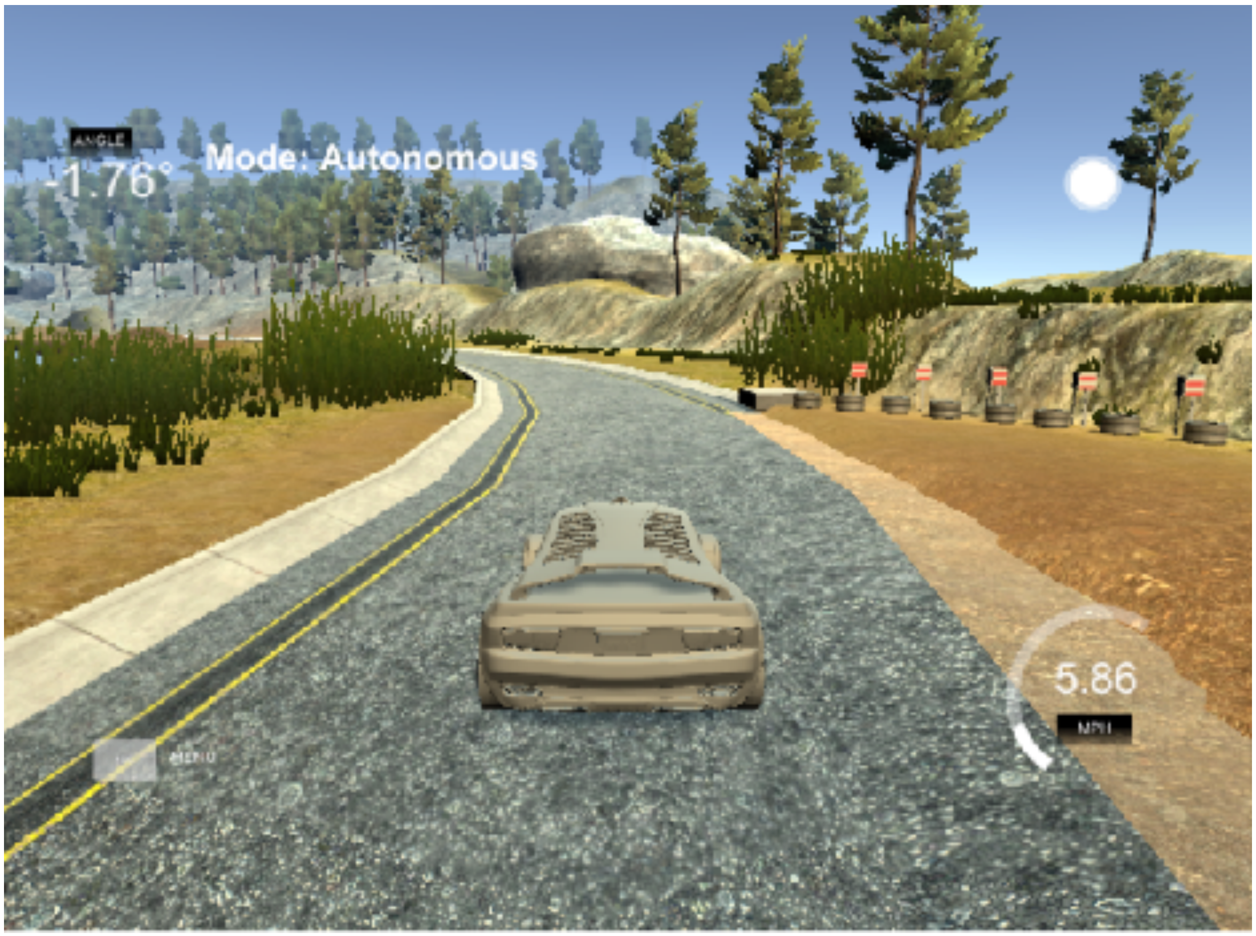
The overall strategy for deriving a model architecture was to first use several convolutions and RELU activations. Following that I used activations and the Dense() layer to increase the workability of the model.  I found that without these the model was very lost in testing.  What I found most interesting about this project was there is very little connection between validation accuracy and how well the car drives.  It is more about the data you collect and learn on.  If all your data is on driving in a straight line you could have 80% validation accuracy, but the car will head into the trees at the first curve.  I found this challenging because it increased development lead time.  Mainly because it meant to “validate” my model I had to see if the car would make it around the track, rather than examining a set of metrics.

### 2. Final Model Architecture

See the above for exact model architecture but the idea is several convolution layers followed by dense layers.

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

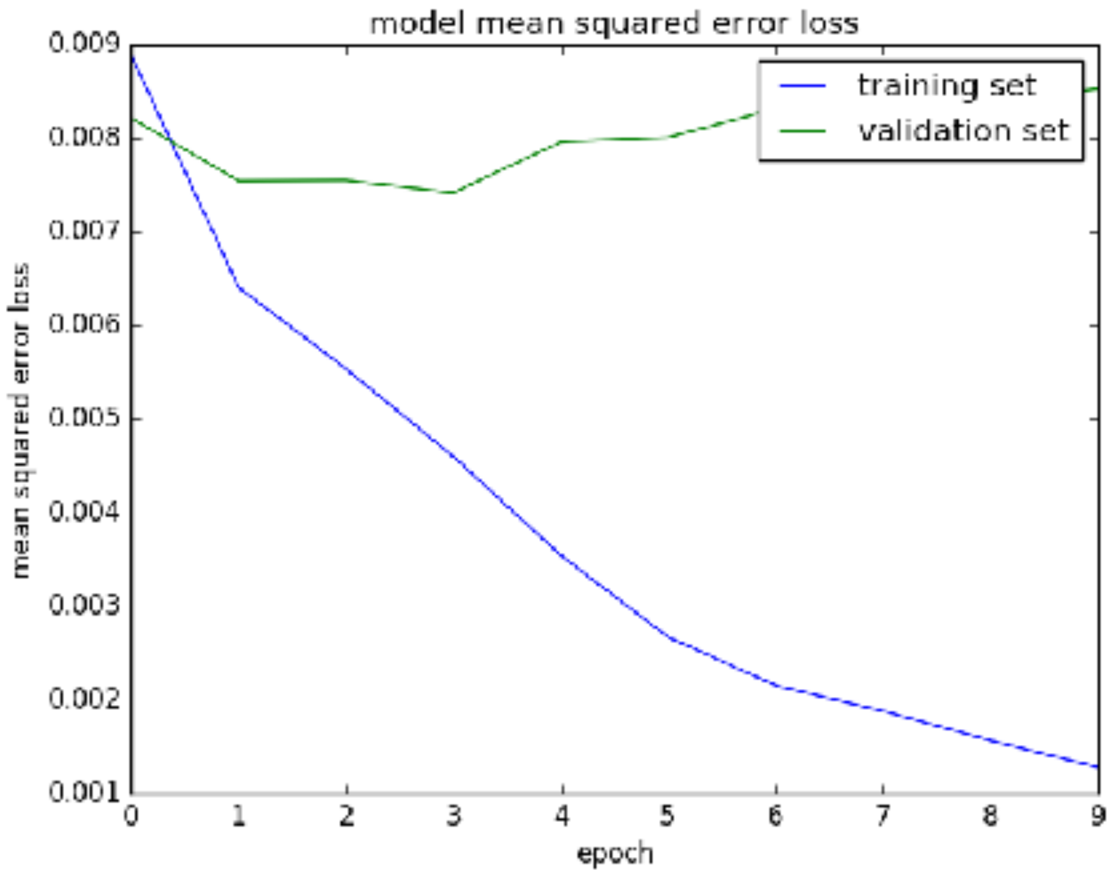
To capture the driving behavior we are interested in I used the provided udacity data and added data when I had problem areas. Specifically tracking curves smoothly and recovering when getting close to the edge of the road. Here is an example image of center lane driving:



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 recover from mistakes. I also found it had issues following curves smoothly.  I recored more data of that as well.

To augment the data set I needed data of the car turning right.  instead of recording more data I just flipped 25% of my data points on a random basis.  That is to say I flipped the image and the associated steering angle.  I found the right turn to be the biggest challenge without this method.  So much so that I even added about 400 data points on just this.

After all my data collection I had 16833 data points. I then preprocessed this data by shuffling with sklearn.  The rest of the preprocessing was done with keras.  I used the crop tool to change the shape of the image being looked at and the lambda



tool to perform a basic normalization.  Additionally I also used a validation set to get an idea of the what sort of results the model was getting during training.

Above is a graph of MSE loss. I would definitely say this is an example of an overfit model.  That being said though it functions in practice.