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 The model I used. drive.py Udacity provided code, minor adjustments made
   - model.h5 The trained weights for the model
   - model.json the trained convolution neural network
   - writeup_report.html My write up, used the Udacity supplied template for formatting.

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.json
   ```

1. **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 Overview

1. **An appropriate model architecture has been employed**

   My model is based off of the NVidia model (https://arxiv.org/pdf/1604.07316v1.pdf) and contains

   - initial layer is a normalization layer
   - 4 Convolutional layers
   - RELU activations and Max pooling at each layer
   - A layer to flatten
   - 4 fully connected layers.

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

   Each convolutional layer contains dropout to prevent overfitting. Validation data was culled from the training data (20%), and the model was tested by driving 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 a combination of center lane driving, recovering from the left and right sides of the road.

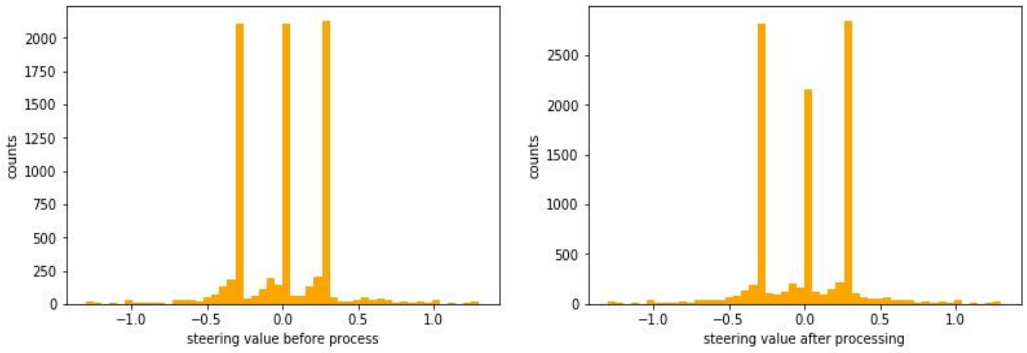
   For details about how I created the training data, see the next section.

Model Architecture and Training Strategy

1. **Solution Design Approach**

I originally tried a smaller model based off what we created for the traffic signs projects. This model proved inaccurate. I then investigated the NVidia model, as it seemed to have some success with other students. I was originally looking at the accuracy of the model, however my mentor suggested I concentrate more on the MSE or MAE, depending on the architecture used. I switched to MSE and could see the loss drop considerably as it trained.

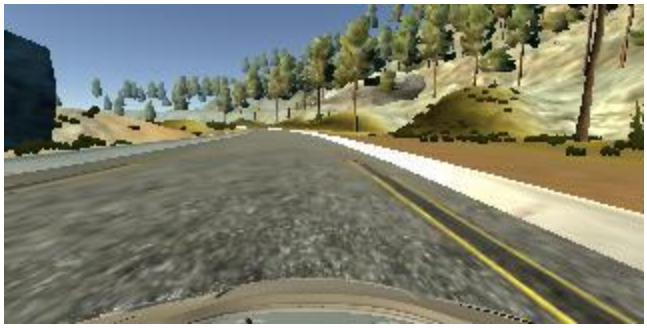
I performed a split (20%) of the training data to provide validation data. I found the model would have a low MSE but would not be able to navigate the course. Now, I began to manipulate the images. I performed random image processing (ProcessImage function). I learned how to plot a histogram in Python as could visualize the steering angles. I found random flipping didn't always work out to well to balance the data as I would lose one image. I then implemented code to keep the original image and then provide a flipped copy. This helped to balance out the data much better. I then performed a random gamma adjust to darken the images.



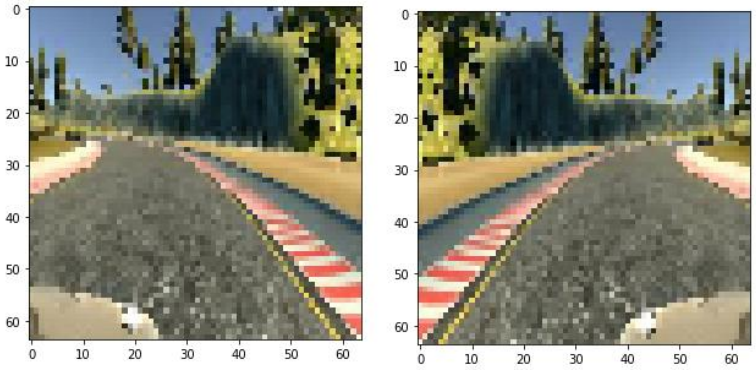
The original data did not have any recovery images, so I gathered these by recovering from both the left and right hand sides of the track. Because the track was so heavily left turn skewed, I decided to drive the track going the opposite direction.

Originally I was not resizing my images, but found resizing to 64x64 still allowed the model to train, and train faster. This is still a bit confusing as the images are not proportionally correct, but they did work.

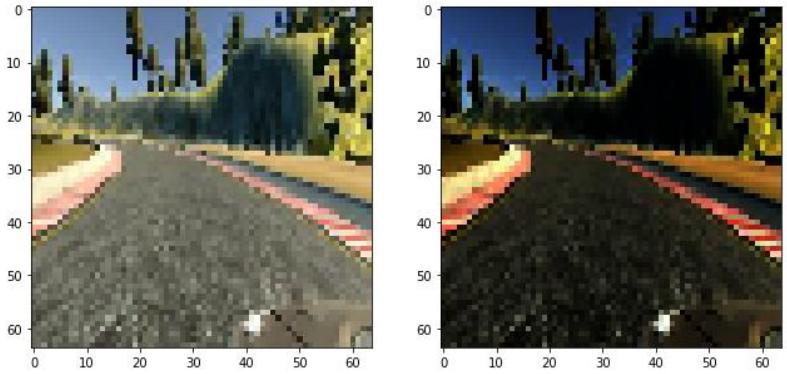
Left Side Recovery



Right side recovery



Original and flipped image



Original and gamma adjusted image

At the end of the process, the vehicle could drive autonomously around the track without leaving the road.

2. Final Model Architecture

The final model architecture consisted of a convolution neural network with the following layers and layer sizes: (please excuse the formatting, still getting used to Jupyter Notebooks)

Model Architecture

Layer (type)	Output Shape	Param #	Connected to
lambda_1 (Lambda)	(None, 64, 64, 3)	0	lambda_input_1[0][0]
convolution2d_1 (Convolution2D)	(None, 60, 60, 24)	1824	lambda_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 30, 30, 24)	0	convolution2d_1[0][0]
dropout_1 (Dropout)	(None, 30, 30, 24)	0	maxpooling2d_1[0][0]
activation_1 (Activation)	(None, 30, 30, 24)	0	dropout_1[0][0]
convolution2d_2 (Convolution2D)	(None, 26, 26, 36)	21636	activation_1[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 13, 13, 36)	0	convolution2d_2[0][0]
dropout_2 (Dropout)	(None, 13, 13, 36)	0	maxpooling2d_2[0][0]
activation_2 (Activation)	(None, 13, 13, 36)	0	dropout_2[0][0]
convolution2d_3 (Convolution2D)	(None, 9, 9, 48)	43248	activation_2[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 4, 4, 48)	0	convolution2d_3[0][0]
dropout_3 (Dropout)	(None, 4, 4, 48)	0	maxpooling2d_3[0][0]
activation_3 (Activation)	(None, 4, 4, 48)	0	dropout_3[0][0]
convolution2d_4 (Convolution2D)	(None, 2, 2, 64)	27712	activation_3[0][0]
maxpooling2d_4 (MaxPooling2D)	(None, 1, 1, 64)	0	convolution2d_4[0][0]
dropout_4 (Dropout)	(None, 1, 1, 64)	0	maxpooling2d_4[0][0]
activation_4 (Activation)	(None, 1, 1, 64)	0	dropout_4[0][0]
flatten_1 (Flatten)	(None, 64)	0	activation_4[0][0]
dense_1 (Dense)	(None, 100)	6500	flatten_1[0][0]
activation_5 (Activation)	(None, 100)	0	dense_1[0][0]
dense_2 (Dense)	(None, 50)	5050	activation_5[0][0]
activation_6 (Activation)	(None, 50)	0	dense_2[0][0]
dense_3 (Dense)	(None, 10)	510	activation_6[0][0]
activation_7 (Activation)	(None, 10)	0	dense_3[0][0]
dense_4 (Dense)	(None, 1)	11	activation_7[0][0]

Total params: 106491