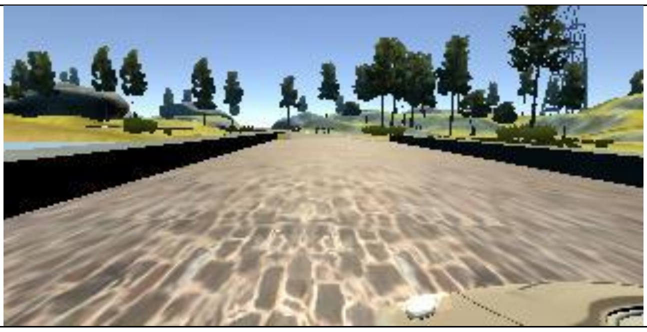
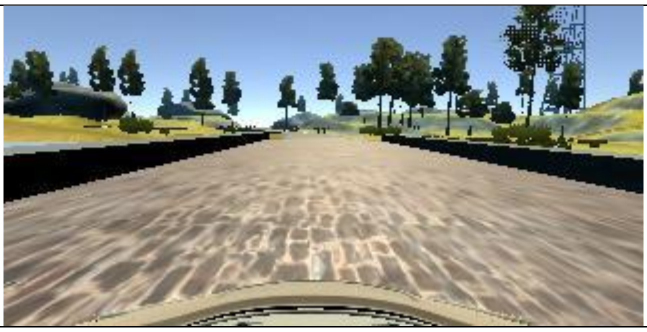
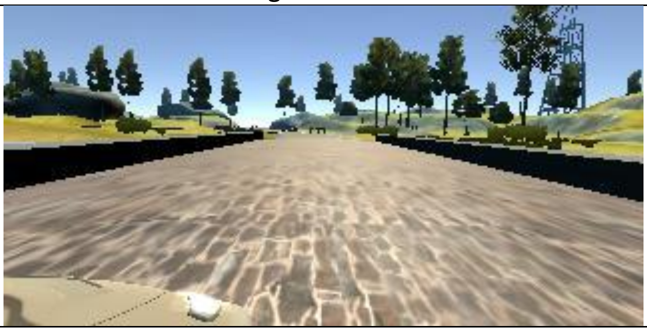
Behavioral Cloning Project

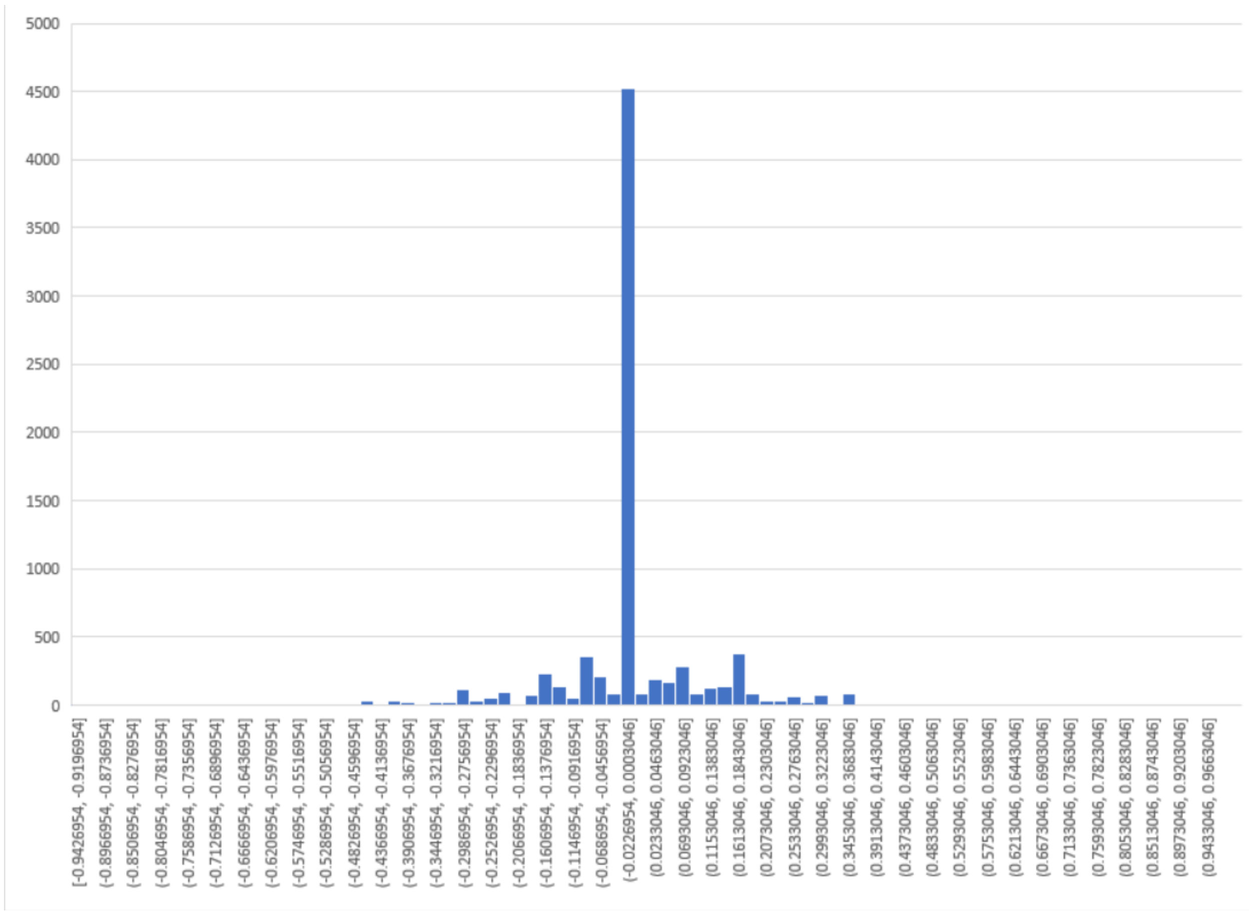
Dataset

For my implementation of the model, I only used the dataset provided by Udacity.

The Udacity dataset consists of 8036 unique entries that consists of 3 images per entry (left cam, center cam, right cam) and 1 number for the steering angle. The dataset is generated from the first track.

Left camera

Center camera

Right camera


The dataset composition is heavily biased toward going straight. In fact, more than half of all entries have a steering angle of 0. This is an unbalanced dataset, which require augmentation to ensure that model will not be skewed toward predicting 0 steering angle.



Data Preprocessing

Prior to preprocessing, the dataset is split into training set and validation set

Each image in the test set has a dimension of 160 x 320 x 3

As part of the Keras model, the image is first normalized using a Lambda function

$$x = \frac{x}{255} - 0.5$$

Then using Cropping2D, the top 70 (sky) rows and bottom 26 (hood of car) rows of the image is removed. The areas that are cropped out are not important to helping the model learn. All that's needed is the line markings features on either side of the road

Finally, another Lambda function is used to resize the image to 64 x 64 x 3

Data Augmentation

The following techniques are used to augment the training data within the generator function

Balancing

Randomly removing 95% of the 0 steering angle entries in the training set. This is necessary to ensure that the 0 steering angle data is not over represented and cause the model to be biased toward predicting going straight

Camera Angle

Randomly selecting images from the left, center, and right camera and adjusting the steering angle compensation based on the selected camera. This is to help the model learn how to re-center the car if it veers too close to the edge of the lane markings. From experimentation, I have found 0.25 to be a reasonable steering angle compensation adjustment.

This also helps with balancing the dataset because the steering angle will be non-0 when the left or right camera view images are chosen, even if the corresponding steering angle label is 0.

The probability of each camera angle being selected is $\frac{1}{3}$

Flipping Image

Randomly selecting images to flip horizontally. This is to ensure that the model is exposed to a more even mix of left turn and right turn. Because the loop is driven clockwise, most of the turns are to the left. Without flipping the images, the model only works well for veering left, and will cause the car to go off road when veering toward the right (after the first sandbanks after the bridge).

The probability of each camera angle being selected is $\frac{1}{2}$

Model Architecture

I started with the NVIDIA model. Unfortunately, that model ended up being way too large and cumbersome to train. It was also performing worse compared to a simple Lenet model. I came to the conclusion that perhaps the model is too complicated and overkill for what is needed on this project. What I did was to drastically truncate the NVIDIA model by removing layers and changing layer sizes and convolution parameters.

My final architecture consists of 3 convolutional layers and 3 fully connected layers.

I selectively added dropout layers to prevent overfitting the data.

Adam optimizer was used with default settings.

I trained the model for 5 epochs, batch size is 32

Layer	Description
Input	64 x 64 x 3 RGB image
L1 Convolution	1x1 stride, 5x5 filter, VALID padding, 60 x 60 x 8 output
L1 Activation	RELU
L1 Pooling	2x2 stride, 30 x 30 x 8 output
L1 Dropout	Keep prob = 0.5
L2 Convolution	1x1 stride, 5x5 filter, VALID padding, 26 x 26 x 8 output
L2 Activation	RELU
L2 Pooling	2x2 stride, 13 x 13 x 8 output
L2 Dropout	Keep prob = 0.5
L3 Convolution	1x1 stride, 5x5 filter, VALID padding, 5 x 5 x 16 output
L3 Activation	RELU
L3 Dropout	Keep prob = 0.5
Flatten	400 output
L4	100 output
L5	50 output
L6	10 output
Prediction	angle

After 5 epochs, the training loss is 0.0266, and the validation loss is 0.0160

```
200/200 [==============================] - 18s - loss: 0.0442 - val_loss: 0.0157

Epoch 2/5

200/200 [==============================] - 16s - loss: 0.0306 - val_loss: 0.0148

Epoch 3/5

200/200 [==============================] - 16s - loss: 0.0284 - val_loss: 0.0134

Epoch 4/5

200/200 [==============================] - 17s - loss: 0.0273 - val_loss: 0.0150

Epoch 5/5

200/200 [==============================] - 17s - loss: 0.0266 - val_loss: 0.0160
```