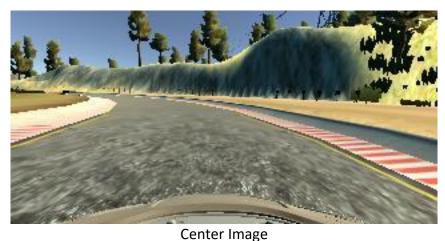
# **Behavioral Cloning Project**

By: Tyler Zamjahn

# **Approach**

To develop training data in the simulator, I recorded runs traversing the track in both directions using the mouse to steer. The dataset consisted of 5765 center, left and right images with corresponding steering angles. In order to increase the number of images in the dataset and to train the model to more aggressively correct itself when the car moved closer to the lane boundaries, the left and right images were used with an addition of 0.15 to the angle measure in the opposite direction of the camera position. An example from each camera position with its original and corrected angle inputs is shown below.



Original Angle: 0, Corrected Angle: 0



Left Image Original Angle: 0, Corrected Angle: 0.15



Right Image
Original Angle: 0, Corrected Angle: -0.15

By using the left and right images, the total number of images in the dataset grew to17295 images. These images were split in to training and validation sets of 13836 images and 3459 images respectively, using an 80/20 split.

### **Model Architecture**

The model used was based on the NVIDIA model shown in the project lesson. This model was chosen based on its larger architecture and prove results in the self-driving space.

Input Shape	Layer	Description	Output Shape
160x320x3	Normalization Around 0	Mean centered around 0	160x320x3
160x320x3	Image Cropping	Cropped 70 pixels from top and 25 from bottom of each image	65x320x3
65x320x3	Convolutional Layer	24 5x5 filters with a step size of 2	31x158x24
31x158x24	Convolutional Layer	36 5x5 filters with a step size of 2	14x77x36
14x77x36	Convolutional Layer	48 5x5 filters with a step size of 2	5x37x48
5x37x48	Convolutional Layer	64 3x3 filters with a step size of 1	3x35x64
3x35x64	Convolutional Layer	64 3x3 filters with a step size of 1	1X33x64
1x33x64	Flatten		2112
2112	Densely Connected Layer		100
100	Densely Connected Layer		50
50	Densely Connected Layer		10
10	Densely Connected Layer		1

Mean square error was used for the loss function and the Adam optimizer was used to train the network. The model was only trained for three epochs and the data was augmented to prevent overfitting. The model was trained using an AWS GPU instance so a generator was not needed for the model.

### **Running the Model**

Once the model was trained, it was then evaluated on the autonomous simulator. The model performed well with a desired speed set to 25 for the controller in the drive.py file, however, running the model while recording introduced some lag in the frame rate which cause the car to crash. Setting the desired speed to a lower rate of 15 allowed the car to safely maneuver around the track at a faster rate than the initial speed of 9.

#### Improvements to the Model

After successfully implementing the NVIDIA model, the third convolutional layer was changed in to a max pooling layer with a step size of two in order to reduce the size of the model to prevent overfitting. The cropping and normalization layers were also swapped to reduce the preprocessing computations since the normalization was not relative to the entire image. The models architecture then became:

Input Shape	Layer	Description	Output Shape
160x320x3	Image Cropping	Cropped 70 pixels from top	65x320x3
		and 25 from bottom of	
		each image	
65x320x3	Normalization Around 0	Mean centered around 0	65x320x3
65x320x3	Convolutional Layer	24 5x5 filters with a step	31x158x24
		size of 2	
31x158x24	Convolutional Layer	36 5x5 filters with a step	14x77x36
		size of 2	
14x77x36	Max Pooling Layer	Step size of 2	7x38x36
7x38x36	Convolutional Layer	64 3x3 filters with a step	5x36x64
		size of 1	
5x36x64	Convolutional Layer	64 3x3 filters with a step	3X34x64
		size of 1	
3x34x64	Flatten		6528
6528	Densely Connected Layer		100
100	Densely Connected Layer		50
50	Densely Connected Layer		10
10	Densely Connected Layer		1

Also, this second model was trained using the EarlyStopping callback function to prevent overfitting.

# **Further Improvements**

The model could be further improved by incorporating more data from both tracks which would likely require the implementation of a generator function. Additionally, the drive.py controller could be fine-tuned to implement smoother driving and varying speeds to account for straightaways and turns.