Behavior Cloning

*Udacity Term - 1*

*Period October to January*

## System & Software Specification

OS - Windows 7

Hardware: Intel  i7 core CPU

Programming language: Python 3.x

 Python Libraries used:

* OpenCV:  library name "cv2" . Used for image processing
* Numpy: Array related functionality
* Matplotlib: used for plotting images
* Math: Used for finding square roots
* Glob: To load images from folder
* moviepy.editor: For loading and creating a video
* pickle: To save and load training feature
* deque: library to create and use double ended que
* scipy: Library for labelling
* skimage: A library for getting the Hog features
* keras.models : Sequential, Model
* keras.layers : Flatten, Dense, Lambda, Convolution2D, Cropping2D
* keras.layers.pooling : MaxPooling2D

## Description

Objective: Goal of the project is to train a model using deep learning and use the model to drive a car in a simulator.

## Training Process

### 3.1 Training data generation

The simulator provided by Udacity is used to generate the training data. Track 1 & 2 is used for this process. The images generated by the simulator is saved along with the driving\_log.csv which has the list of images generated. Additionally on track 1, the car is also driven backward so that the training can be more generalized.

### 3.2 Model Architecture

The model is based on the NVidia group model. The link for the same is provided below

https://devblogs.nvidia.com/deep-learning-self-driving-cars/

Loading Images: The images are loaded from the folder where the simulator saved the images.

Some sample images are shown below

The input images are cropped

model.add(Cropping2D(cropping=((50,20), (0,0))))

sample cropped image is as shown below

In order to generate more data set, the data collected from the simulator is augmented, by flipping the image

images.append(cv2.flip(image,1))

angles.append(measurement\*-1.0)

The data is split into training and validation sets

from sklearn.model\_selection import train\_test\_split

samples = list(zip(imagePaths, measurements))

train\_samples, validation\_samples = train\_test\_split(samples, test\_size=0.2)

The model is build based on the nvidia group architecture

model.add(Convolution2D(24,5,5, subsample=(2,2), activation='relu'))

model.add(Convolution2D(36,5,5, subsample=(2,2), activation='relu'))

model.add(Convolution2D(48,5,5, subsample=(2,2), activation='relu'))

model.add(Convolution2D(64,3,3, activation='relu'))

model.add(Convolution2D(64,3,3, activation='relu'))

model.add(Flatten())

model.add(Dense(100))

model.add(Dense(50))

model.add(Dense(10))

model.add(Dense(1))

Adam optimizer is used

model.compile(loss='mse', optimizer='adam')

The number of epochs is set to 3

history\_object = model.fit\_generator(training\_generator, samples\_per\_epoch= \

len(train\_samples), validation\_data=validation\_generator, \

nb\_val\_samples=len(validation\_samples), nb\_epoch=3, verbose=1)

The generated model is saved as below

model.save('model.h5')

The saved model is testing with the simulator in autonomous mode for track 1

python drive.py model.h5