**Behavioural Cloning**

**Behavioral Cloning Project Overview**

This project aims to learn from the human driving behavior using CNN, and drive the car in a simulator provided by Udacity. In the recording mode, the simulator records the training examples (images sampled from video at a frequency of 10 Hz), and the respective steering angle. The throttle and vehicle speed are not taken into consideration in this project. This training data is then fed to the network to build the final model.

**Files Submitted**

I have updated the following files:

* models.py containing the entire CNN architecture and processing pipeline
* model.h5 the trained convolution neural network data\_pipe.py, containing code for generators which stream the training images from disk and apply data augmentation
* drive.py,
* README.pdf summarizing the project
* video.mp4 the video recording of model running on Track1 in simulator

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

**Implementation**

Training Data

A good driving behaviour data is essential for a good mode. I recorded the training data on Track 1. Some of the driving behaviours recorded are

* Two full laps on a nominal speed driving swiftly on the turns.
* One lap in the reverse direction.
* Some recovery driving samples also, where I took the car to one edge of the road, and steered it back to centre position.

Then I combined this data with the dataset which was provided by Udacity.

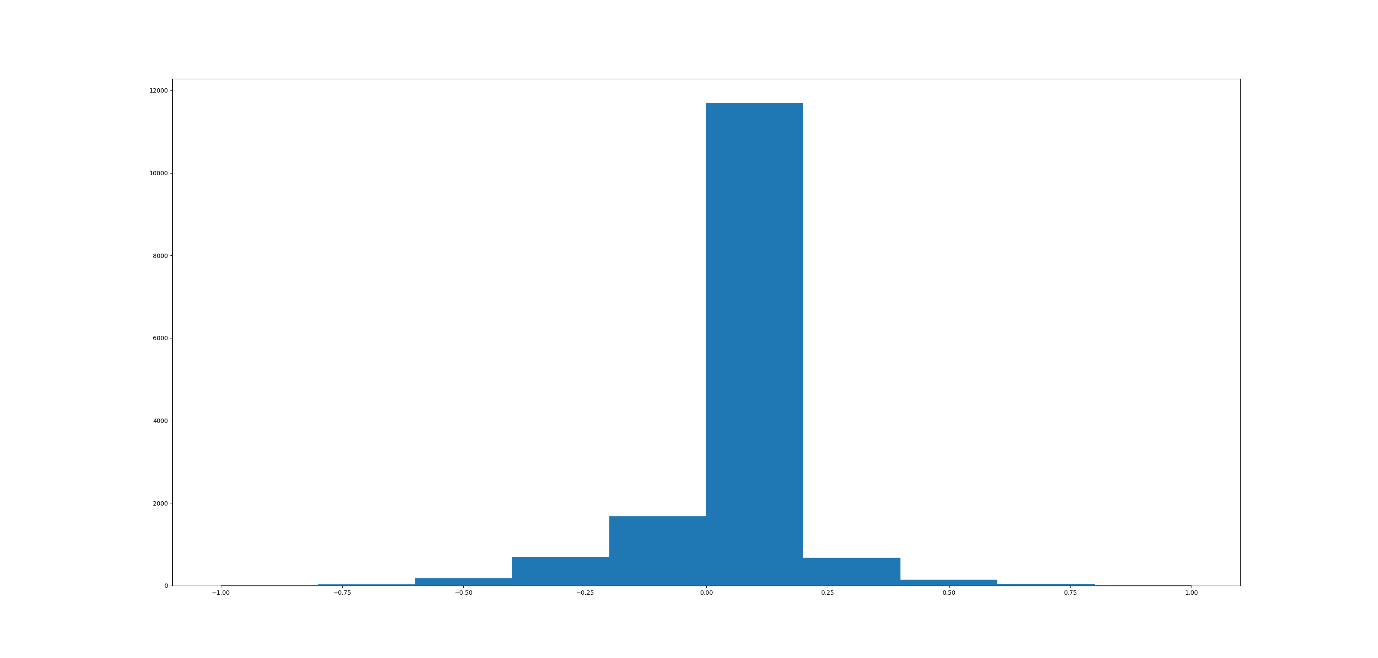
Data Visualization

The training data is then plotted to see the spread of steering angles. There is a huge imbalance in the data and most of data is with steering angle of 0. Thereby some pre-processing techniques are applied to have a good data for neural network.

Total training examples: 15172

Total input images: 45616 (15172 \* 3)

Frequency of training examples with Zero Steering angle: ~12000



Data Pre-Processing

I have applied the following pre-processing techniques

1. Data augmentation is done by flipping the input images horizontally and negating the angles.
2. Removal of some of the training examples with zero steering angles.
3. Data is normalized with the use of lambda layers.

A correction factor of 0.15 is applied to account for images coming from side cameras.

1. The input image has some of the area on top and bottom which is not required for this model. Hence image is cropped by removing top ~35% and bottom ~10% area. And then the image is resized to be able to map to NVIDIA architecture.

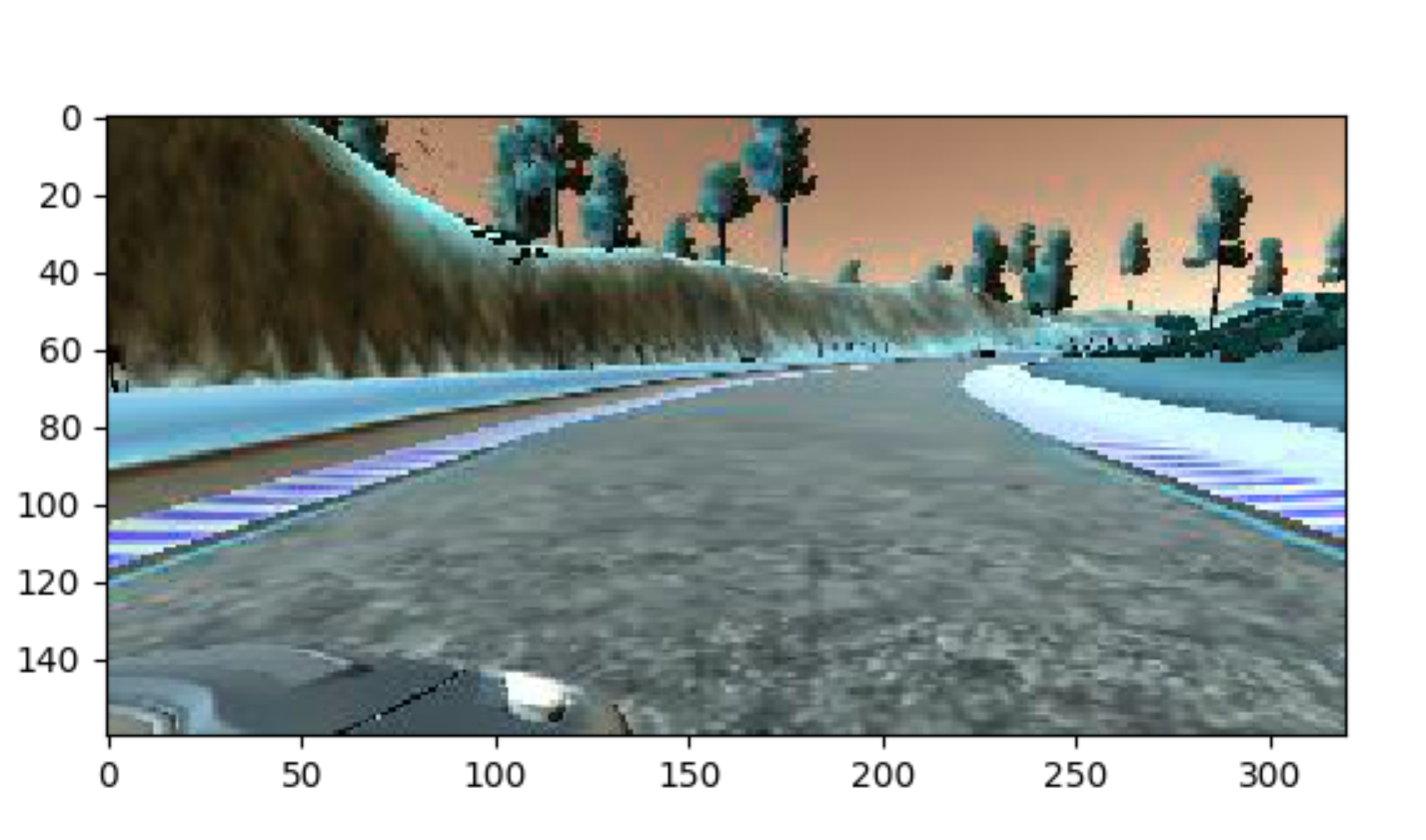


Figure 1 : Original image (160x320x3)

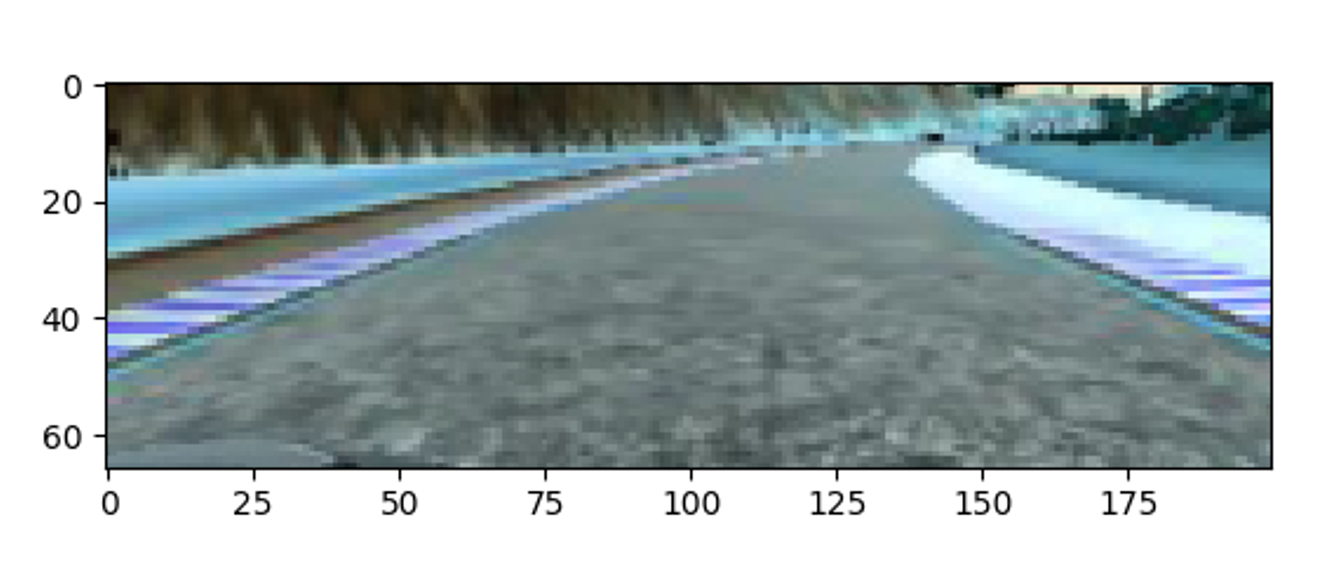


Figure 2 : Cropped and Resized image (66x200x3)

**Network Architecture**

Training:

The convolutional neural network architecture is inspired by NVIDIA's End to End Learning for Self-Driving Cars paper. To add non-linearity to model ELU activations are used. And to deal with overfitting L2 regularization is added to each convolutional and fully connected layer.

For training batch\_size of 64 is selected. Since training data is large and limited memory, fit\_generator API from Keras library is used. An Adam optimizer is used for optimization. This requires little or no tuning as the learning rate is adaptive.

The model is trained for 10 epochs leading to training loss of 0.0731 and validation loss of 0.0682.

Testing:

The trained model is tested on Track 1 and the recording of video is included. The performance is quite smooth. I did not use training samples from Track 2, and testing on Track 2 showed average results.

**Conclusion and Further:**

It was interesting to see the improvement in the final model by experimenting with different types of training data and CNN layers.

Initially it was challenging to get the vehicle to stay on the road and it would go out at times. However after inserting in the recovery data, the performance had improved drastically.

1. Further improvement to the model can be made by making use of advanced image processing API’s to augment the data (jitters, rotations, shifts)
2. Since the performance was model was not good on Track 2 , the model can further be generalized by making use of other techniques which help to reduce overfitting.
3. It will be interesting to learn later how other factors like speed and throttle are incorporated in a machine learning model.