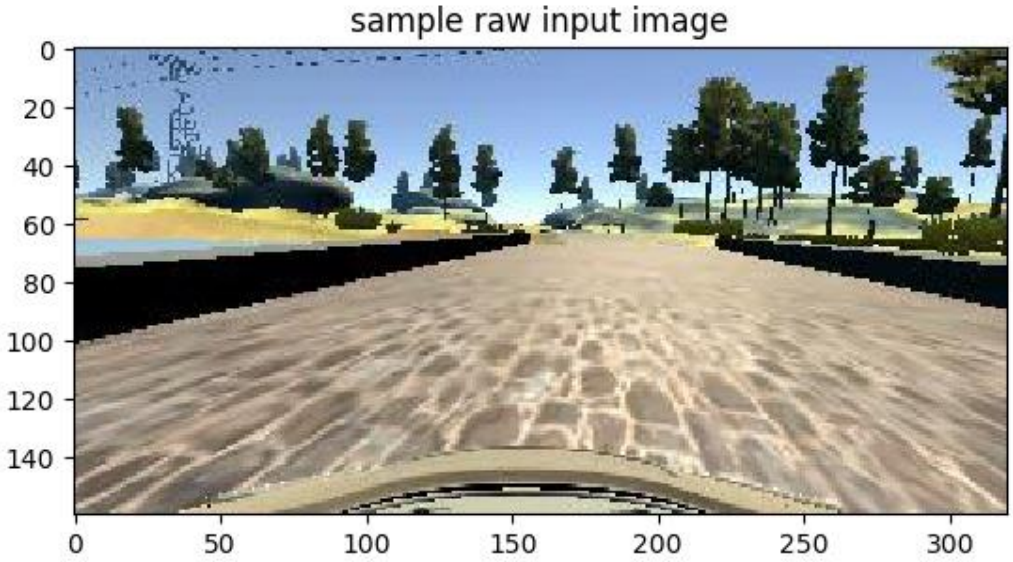
UDACITY Self Driving Car Engineering Program

Project #3:Behavioral Cloning

In this Project it is asked to drive the car and record the steering angle, and from those input – output pairs train a model that drives on itself.

The Pipeline

1- The input data gathering
I used both the supplied Udacity's data and my own data. I record recovery data, ie when the car was inside the lane but driving towards the edges, I centered the car back and recorded while doing so. It significanlty improved the model behaviour.

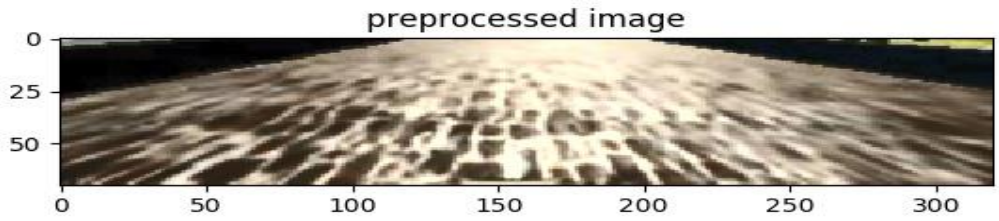


2- Preprocessing

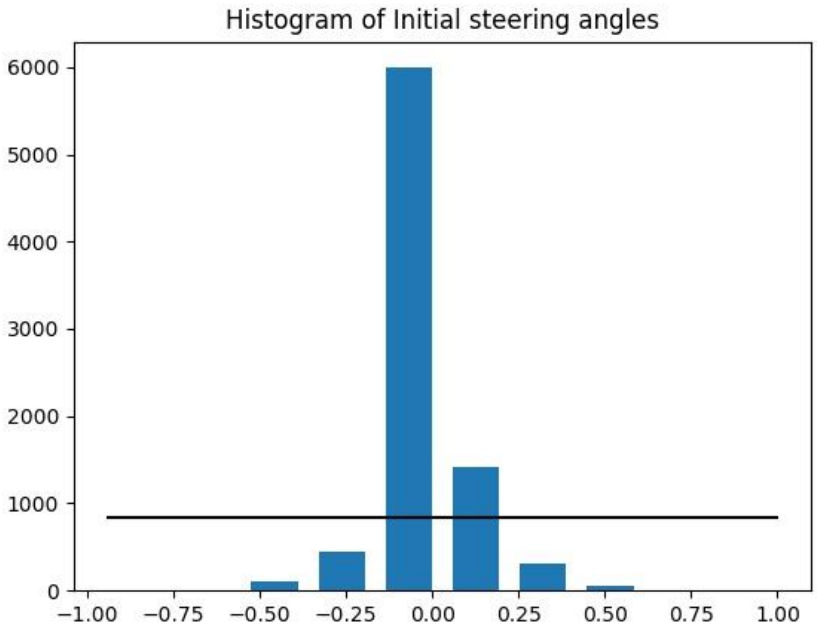
The top part of the input image, as seen above, has unnecessary information, thus firstly I cropped the image. It also reduced the processing power required for both training and driving the car.

Secondly, I normalized the V channel in YUV colorspace to standardize the input.

Also, I use YUV colorspace fort he rest of the model instead of RGB data.



3- Augmentation

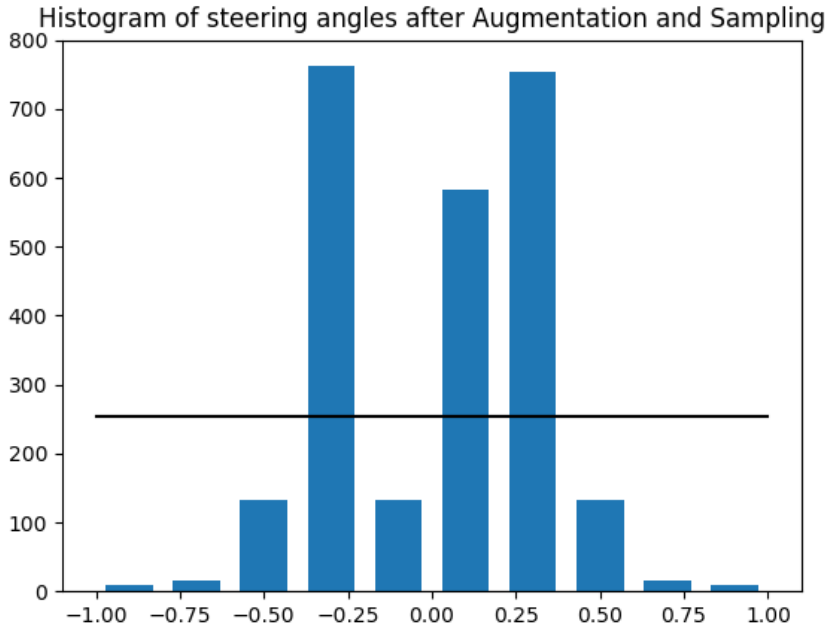


As seen from the initial histogram, the steering angle distribution is not even, thus augmentation was necessary.

I flipped the image and reverse the steering angle who are not near zero. It not only helped increase the

lturning data number but also eliminate the bias towards left or right turning, as each side has exactly same number of inputs.

Besides that, it did not needed to further augment the data, the model smoothly followed the roads and bridges.



4- Model

I used Nvidia's model as a baseline, however it did not give satisfying results. Thereafter I modified it a bit and here is the final model:

-normalization layer

-5*5*24 convulution layer
-5*5*36 convolution layer
-5*5*48 convolution layer

-3*3*64 convolution layer
-3*3*64 convolution layer

(Above is same as Nvidia's model)

-150 dense layer
-70 dense layer
-10 dense layer
-1 final layer (output)

Since I did not use right or left images, I want the model to be bigger to adapth to the given center image, and it turned out well.

At the end of the learning, the losses obtained are: **loss: 0.0425 - val_loss: 0.05**

Since there is not much gap in between training and validation loss and they both are low, it can be summarized that the model neither underfitted nor overfitted