Behavioral cloning

-by K.Vikraman

Link to YouTube video: https://www.youtube.com/watch?v=q7oJLq4uXQo

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road

Data acquisition

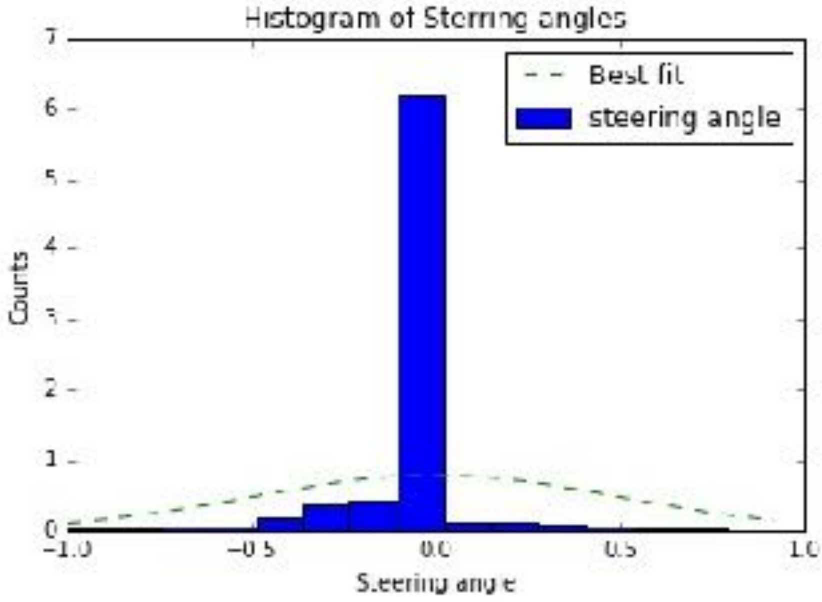
Training data :

- 3 Laps center line driving
- 1 lap recovery
- 1 lap smooth drive in curve

In total I collected 10,257 images including images from all three camera positions

Steering angles:

The following plot is the histogram of steering angle from the dataset. The data is centered around Zero and has a standard deviation of 0.5 radians.



Zero value is predominant indicating straight roads in majority of the track.

Data preprocessing :

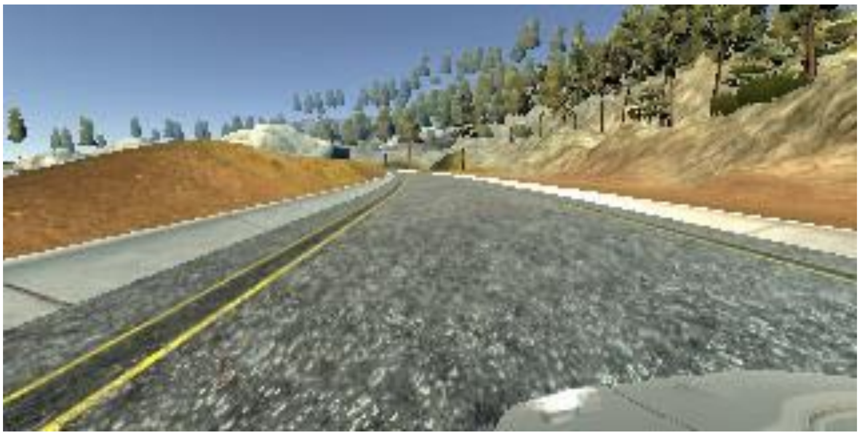
1) Cropping of Unwanted data. The upper section of the image has a lot of unwanted data. So cropped them.
2) Normalisation : To ensure smooth optimisation I normalised the data.

Data Augmentation :

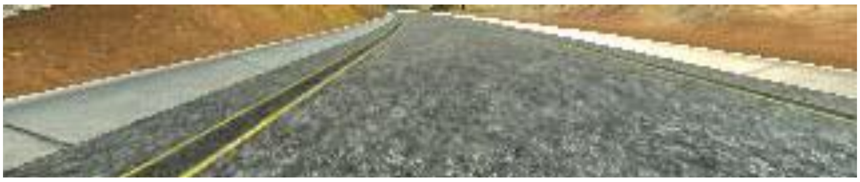
- I used images from left and right cameras as well and adjusted the steering angle by +/- 0.2 accordingly
- I randomly flipped images and inverted the steering angle

Validation Data:
I used 20% of my training data for validation

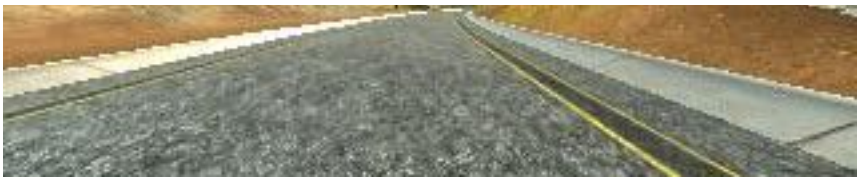
Data preparation



Original Image



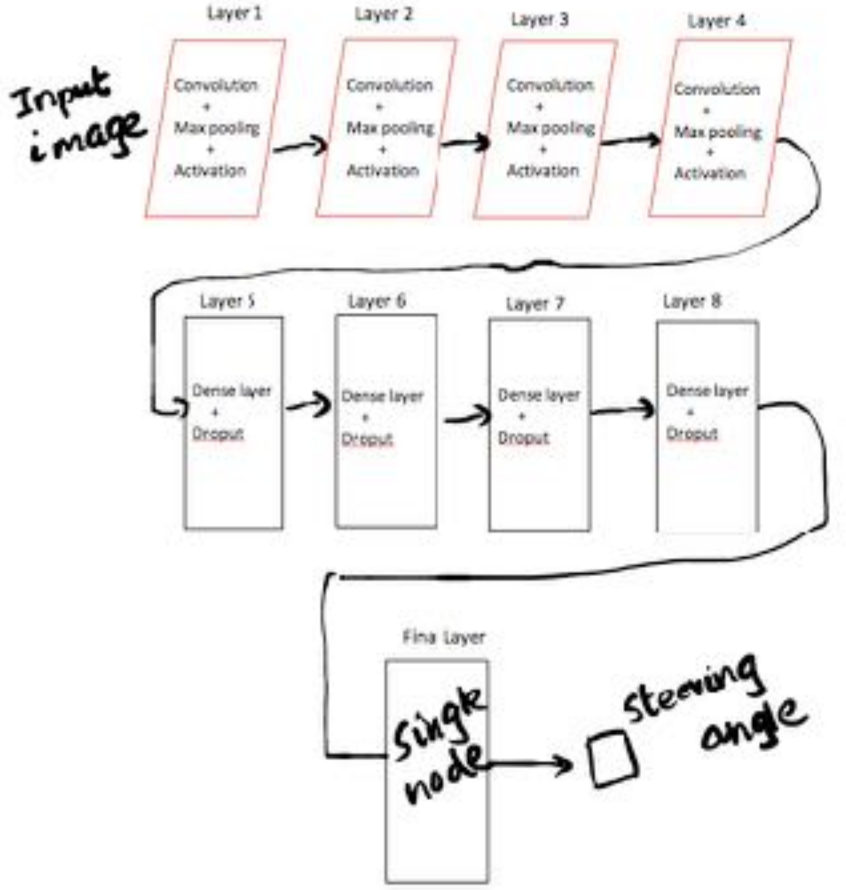
Cropped Image



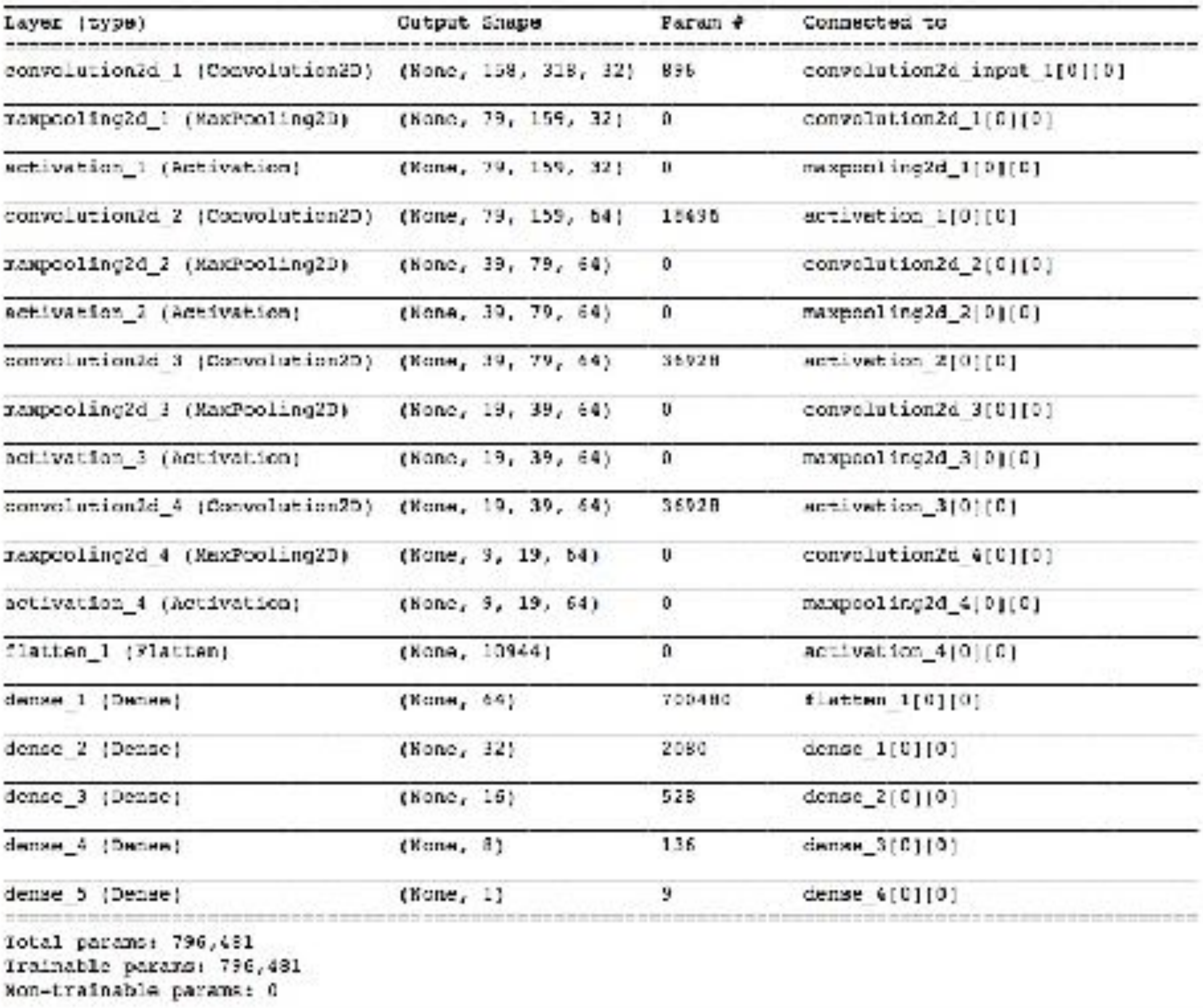
Flipped Image

Model architecture:

For the given model, we have an image as an input and a steering angle(floating value) as an output. In order to accomplish this, I used few layers of Convolutional Neural Network to handle the images and few layers of fully connected layers to perform regression and produce the desired output.
I used a variant of AlexNet to accomplish this task. The exact network architecture is as follows.



The visual of the architecture with exact parameters:



```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
convolution2d_1 (Convolution2D)  (None, 158, 318, 32)  896         convolution2d_input_1[0][0]
____________________________________________________________________________________________________
maxpooling2d_1 (MaxPooling2D)    (None, 79, 159, 32)   0           convolution2d_1[0][0]
____________________________________________________________________________________________________
activation_1 (Activation)        (None, 79, 159, 32)   0           maxpooling2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 79, 159, 64)   18496       activation_1[0][0]
____________________________________________________________________________________________________
maxpooling2d_2 (MaxPooling2D)    (None, 39, 79, 64)    0           convolution2d_2[0][0]
____________________________________________________________________________________________________
activation_2 (Activation)        (None, 39, 79, 64)    0           maxpooling2d_2[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 39, 79, 64)    36928       activation_2[0][0]
____________________________________________________________________________________________________
maxpooling2d_3 (MaxPooling2D)    (None, 19, 39, 64)    0           convolution2d_3[0][0]
____________________________________________________________________________________________________
activation_3 (Activation)        (None, 19, 39, 64)    0           maxpooling2d_3[0][0]
____________________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 19, 39, 64)    36928       activation_3[0][0]
____________________________________________________________________________________________________
maxpooling2d_4 (MaxPooling2D)    (None, 9, 19, 64)     0           convolution2d_4[0][0]
____________________________________________________________________________________________________
activation_4 (Activation)        (None, 9, 19, 64)     0           maxpooling2d_4[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 10944)         0           activation_4[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 64)            700480      flatten_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 32)            2080        dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 16)            528         dense_2[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 8)             136         dense_3[0][0]
____________________________________________________________________________________________________
dense_5 (Dense)                  (None, 1)             9           dense_4[0][0]
====================================================================================================
Total params: 796,481
Trainable params: 796,481
Non-trainable params: 0
```

Since the number of parameters are high, I employed regularisation techniques like dropout and L2 norm to address the issue of overfitting. Initially, I increased the number of parameters to avoid underfitting and improve accuracy.

Model hyperparameters :

After careful fine tuning of the following parameters, I used the following values:

- No. of epochs : 10
- Dropout keep probability: 0.5
- Number of CNN layers: 4
- Number of feature maps per layer(32 in first two layers and 64 in the last two layers of CNN)
)

Discussion :

My model performs very well in the given track. But might not work in a real world scenario because it has been trained completely in a simulated world. Also the training tracks should include situations like speed breaker, where the car should learn to slow down, curves of the shape of ‘U’ that are popular in hill stations.