Model

February 10, 2019

```
In [25]: import os
         import csv
         from scipy import ndimage
         import numpy as np
         import sklearn

         # Importing training data
         PTH_CSV = "../data_bhvr_cln/driving_log.csv"
         PTH_IMG = "../data_bhvr_cln/IMG/"
         #PTH_CSV = "../chk_trg_data_track1_behaviour_cloning2/driving_log.csv"
         #PTH_IMG = "../chk_trg_data_track1_behaviour_cloning2/IMG/"
```

```
In [26]: lines=[]

         with open(PTH_CSV) as csvfile:
             #csv.Sniffer().sniff(f.readline())
             has_header = csv.Sniffer().has_header(csvfile.read(1024))
             csvfile.seek(0)  # Rewind.
             reader = csv.reader(csvfile)
             if has_header:
                 #print header
                 for line in reader:
                     print(line)
                     break
                 next(reader)  # Skip header row.
             for line in reader:
                 lines.append(line)
         print("Total length of data : {}".format(len(lines)))
```

```
['center', 'left', 'right', 'steering', 'throttle', 'brake', 'speed']
Total length of data : 8035
```

```
In [27]: from sklearn.model_selection import train_test_split
         from sklearn.utils import shuffle
         train_lines, validation_lines = train_test_split(lines, test_size=0.2)
```

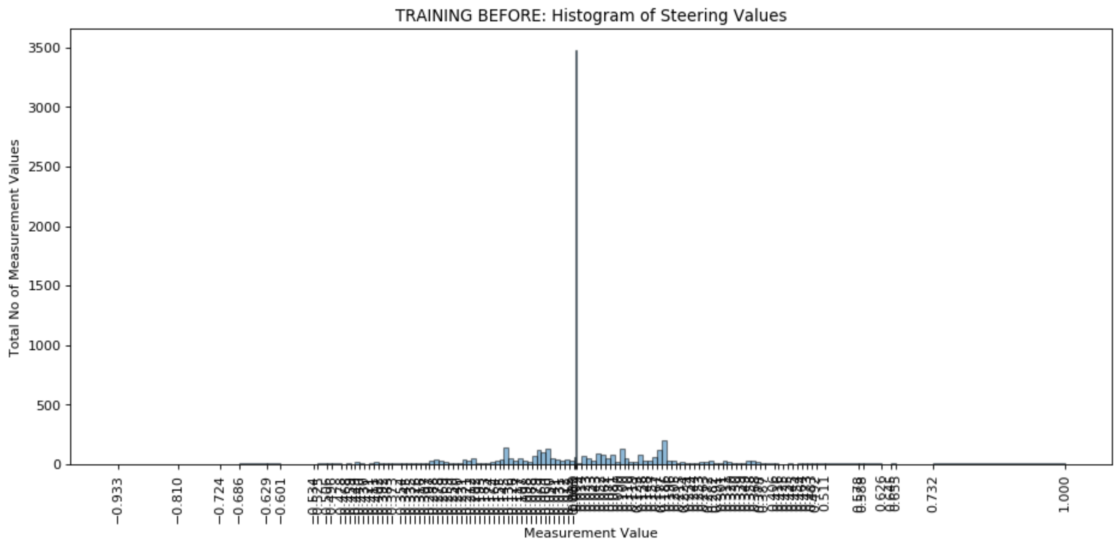
```
In [28]: # Generate Image and measurement arrays for Training Data
         images = []
```

```
measurements = []
def gen_data(lines):
    for line in lines:
        #print(line)
        src_path = line[0]
        #print(src_path)
        f_name = src_path.split('/')[-1]
        #print(f_name)
        #break
        current_path = PTH_IMG +f_name
        image = ndimage.imread(current_path)
        # Appending original image
        images.append(image)
        measurement = float(line[3])
        measurements.append(measurement)

gen_data(train_lines)
X_train = np.array(images)
y_train = np.array(measurements)
# Checking
print("No of Training images : {}".format(len(X_train)))
print("No of Training measurements : {}".format(len(y_train)))
```

```
/anaconda3/envs/IntroToTensorFlow/lib/python3.6/site-packages/ipykernel_launcher.py:13: Depreca
`imread` is deprecated in SciPy 1.0.0.
Use ``matplotlib.pyplot.imread`` instead.
  del sys.path[0]


No of Training images : 6428
No of Training measurements : 6428
```

0.0.1 Visualizing Training Data

```
In [29]: import matplotlib.pyplot as plt
         steering_vals = np.unique(y_train)
         #print(label_nos)
         binwidth= 0.01
         plt.figure(num=None, figsize=(14, 6), dpi=80, facecolor='w', edgecolor='k');
         plt.hist(y_train,steering_vals,alpha=0.5, histtype='bar', ec='black',density=False)
         plt.title('TRAINING BEFORE: Histogram of Steering Values')
         plt.xlabel('Measurement Value')
         plt.ylabel('Total No of Measurement Values')
         plt.xticks(steering_vals,rotation='vertical')
         plt.show()
```



0.0.2 To correct the imbalance we shall reduce the data point with zero steering angle and increase the

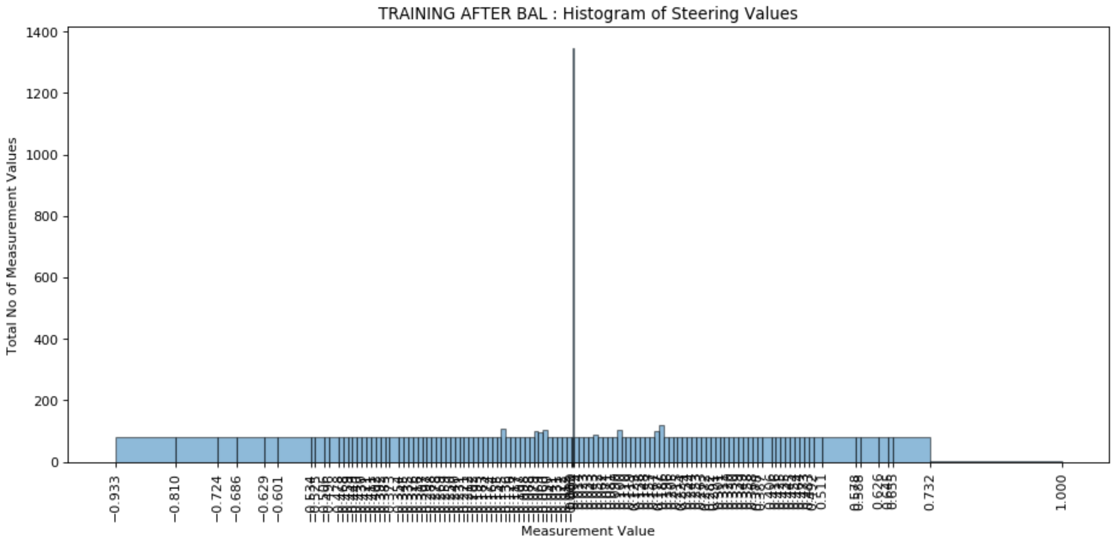
0.0.3 the data points with less values.

```
In [32]: # http://jeffwen.com/2017/07/14/behavioral_cloning

         def  balance_data(data,min_reqd,max_reqd):
             data_output = data.copy()
             no_added = 0
             ## create histogram to know what needs to be added
             steering_angles = np.asarray(data_output[:,3], dtype='float')
             #print(len(np.unique(steering_angles)))
             num_hist, index_hist = np.histogram(steering_angles, np.unique(steering_angles))
             #print(len(num_hist))
             #print(len(index_hist))
             to_be_added = np.empty([1,7])
             to_be_deleted = np.empty([1,1])

             for i in range(1, len(num_hist)):

                 if num_hist[i-1]<min_reqd:

                     ## find the index where values fall within the range
                     match_index = np.where((steering_angles>=index_hist[i-1]) & (steering_ang

                     ## randomly choose up to the minimum needed
                     need_to_add = data_output[np.random.choice(match_index,min_reqd-num_hist[
                    to_be_added = np.vstack((to_be_added, need_to_add))


                elif num_hist[i-1]>max_reqd:

                    ## find the index where values fall within the range
                    match_index = np.where((steering_angles>=index_hist[i-1]) & (steering_angl

                    ## randomly choose up to the minimum needed
                    to_be_deleted = np.append(to_be_deleted, np.random.choice(match_index,num_

         #         if ((index_hist[i-1] <= -0.3) and (index_hist[i-1] >= -0.8)):
         #                 #print(index_hist[i-1])
         #                 match_index = np.where((steering_angles>=-0.8) & (steering_angles<=
         #                 for each in match_index:
         #                     need_to_add = data_output[each,:]
         #                     to_be_added = np.vstack((to_be_added, need_to_add))
         #                     no_added+=1

         #         if ((index_hist[i-1] >= 0.3) and (index_hist[i-1] <= 0.8)):
         #                 match_index = np.where((steering_angles>=3.0) & (steering_angles<=1
         #                 for each in match_index:
         #                     need_to_add = data_output[each,:]
         #                     to_be_added = np.vstack((to_be_added, need_to_add))
         #                     no_added+=1

             ## delete the randomly selected observations that are overrepresented and append
             data_output = np.delete(data_output, to_be_deleted, 0)
             data_output = np.vstack((data_output, to_be_added[1:,:]))


             return data_output

In [33]: # Balance Training data
         train_lines_balanced = balance_data(np.array(train_lines),80,100)
         # Checking
         print("Total length of balanced Training data : {}".format(len(train_lines_balanced))

Total length of balanced Training data : 10892


/anaconda3/envs/IntroToTensorFlow/lib/python3.6/site-packages/ipykernel_launcher.py:52: Depreca


In [34]: # Generate measurement arrays for Balanced Data
         #images = []
         measurements = []
         def gen_data(lines):
             for line in lines:
```

```
#    #print(line)
#    src_path = line[0]
#    #print(src_path)
#    f_name = src_path.split('/')[-1]
#    #print(f_name)
#    #break
#    current_path = PTH_IMG +f_name
#    image = ndimage.imread(current_path)
#    # Appending original image
#    images.append(image)
    measurement = float(line[3])
    measurements.append(measurement)
gen_data(train_lines_balanced)
#X_train = np.array(images)
y_train_balanced = np.array(measurements)
# Checking
#print("No of Training images : {}".format(len(X_train)))
print("No of Training measurements : {}".format(len(y_train_balanced)))
```

```
No of Training measurements : 10892
```

```
In [35]: import matplotlib.pyplot as plt
         steering_vals_bal = np.unique(y_train_balanced)
         #print(label_nos)

         plt.figure(num=None, figsize=(14, 6), dpi=80, facecolor='w', edgecolor='k');
         plt.hist(y_train_balanced,steering_vals_bal,alpha=0.5, histtype='bar', ec='black',dens
         plt.title('TRAINING AFTER BAL : Histogram of Steering Values')
         plt.xlabel('Measurement Value')
         plt.ylabel('Total No of Measurement Values')
         plt.xticks(steering_vals_bal,rotation='vertical')
         plt.show()
```



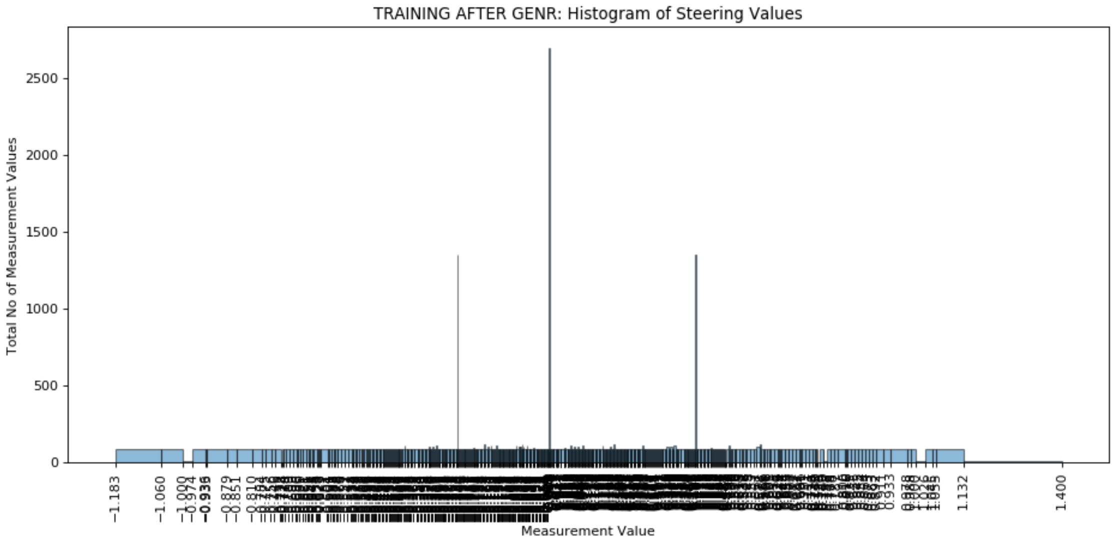
```
In [36]: def  generate_data(lines):
             images = []
             measurements = []

             for line in lines:
                 #print(line)
                 src_path = line[0]
                 #print(src_path)
                 f_name = src_path.split('/')[-1]
                 #print(f_name)
                 #break
                 current_path = PTH_IMG +f_name
                 image = ndimage.imread(current_path)
                 # Appending original image
                 images.append(image)
                 measurement = float(line[3])
                 measurements.append(measurement)
                 #appending flipped image
                 images.append(np.fliplr(image))
                 measurements.append(-measurement)
                 #appending left camera image and steering angle with offset
                 src_path = line[1]
                 f_name = src_path.split('/')[-1]
                 current_path = PTH_IMG +f_name
                 image = ndimage.imread(current_path)
                 images.append(image)
                 measurements.append(measurement+0.4)
                 # appending right camera image and steering angle with offset
                 src_path = line[2]
                 f_name = src_path.split('/')[-1]
                 current_path = PTH_IMG +f_name
                 image = ndimage.imread(current_path)
                 images.append(image)
                 measurements.append(measurement-0.25)

             # trim image to only see section with road
             X_data = np.array(images)
             y_data = np.array(measurements)
             return X_data,y_data

         X_train,y_train = generate_data(train_lines_balanced)
         X_valid,y_valid = generate_data(validation_lines)
         print("No of Training measurements after genr: {}".format(len(y_train)))
```

```
/anaconda3/envs/IntroToTensorFlow/lib/python3.6/site-packages/ipykernel_launcher.py:13: Depreca
`imread` is deprecated in SciPy 1.0.0.
```

```
Use ``matplotlib.pyplot.imread`` instead.
  del sys.path[0]
/anaconda3/envs/IntroToTensorFlow/lib/python3.6/site-packages/ipykernel_launcher.py:25: Depreca
`imread` is deprecated in SciPy 1.0.0.
Use ``matplotlib.pyplot.imread`` instead.
/anaconda3/envs/IntroToTensorFlow/lib/python3.6/site-packages/ipykernel_launcher.py:32: Depreca
`imread` is deprecated in SciPy 1.0.0.
Use ``matplotlib.pyplot.imread`` instead.
```

```
No of Training measurements after genr: 43568
```

```
In [37]: import matplotlib.pyplot as plt
         steering_vals = np.unique(y_train)
         #print(label_nos)
         binwidth= 0.01
         plt.figure(num=None, figsize=(14, 6), dpi=80, facecolor='w', edgecolor='k');
         plt.hist(y_train,steering_vals,alpha=0.5, histtype='bar', ec='black',density=False)
         plt.title('TRAINING AFTER GENR: Histogram of Steering Values')
         plt.xlabel('Measurement Value')
         plt.ylabel('Total No of Measurement Values')
         plt.xticks(steering_vals,rotation='vertical')
         plt.show()
```


TRAINING AFTER GENR: Histogram of Steering Values
Total No of Measurement Values
Measurement Value