SELF DRIVING CAR

1.Importing libraries

In [ ]:

```
import os
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from scipy import pi
from itertools import islice
import cv2
import tensorflow as tf
import pickle
```

2.Reading images names from text file

In [ ]:

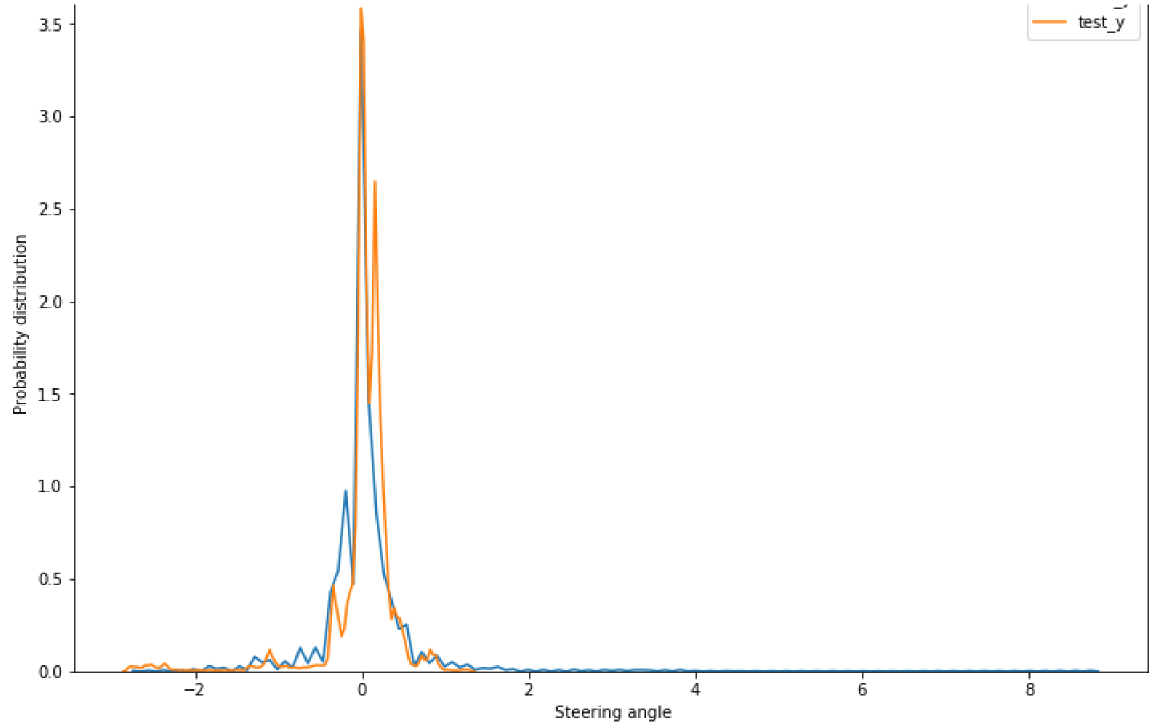
```
#reading the data set from data.txt

dataset_path = '/home/ubuntu/Project/my_data/Autopilot-TensorFlow-master/Autopilot-TensorFlow-mast
er/driving_dataset/'
training_path = os.path.join(dataset_path, 'data.txt')

split = 0.8
X = []
y = []

#islice(text, start, stop, LIMIT) --> ('ABCDEF', 2, None) --> 'C D E F'  --
>https://docs.python.org/2/library/itertools.html
with open(training_path) as f:
    for line in islice(f, None):
        path, angle = line.strip().split()
        full_path = os.path.join(dataset_path, path)
        X.append(full_path)
        y.append(float(angle)*pi/180)

y = np.array(y)
print('Finished processing')

#Split it with 80/20
split_index = int(len(y)*0.8)
y_train = y[:split_index]
y_val = y[split_index:]
```

```
Finished processing
```

2.1 Base Model

In [ ]:

```
#distribution of train and test y
plt.figure(figsize=(12,8))
sns.distplot(y_train, hist=False, label='train_y')
sns.distplot(y_val, hist=False, label='test_y')
plt.xlabel(' Steering angle')
plt.ylabel('Probability distribution')
plt.legend()
plt.show()
```



In [ ]:

```
#Model 0: Base line Model: y_test_pred = mean(y_train_i)
train_mean_y = np.mean(y_train)

print('Test_MSE(MEAN):%f' % np.mean(np.square(y_val-train_mean_y)) )
print('Test_MSE(ZERO):%f' % np.mean(np.square(y_val-0.0)) )
```

```
Test_MSE(MEAN):0.191127
Test_MSE(ZERO):0.190891
```

3. Reading images from the path

In [ ]:

```
dataset_path = '/home/ubuntu/Project/my_data/Autopilot-TensorFlow-master/Autopilot-TensorFlow-mast
er/driving_dataset/'
os.path.exists(os.path.join(dataset_path, 'data.txt'))
```

Out[ ]:

```
True
```

In [ ]:

```
print(len(y_train))
print(len(y_val))
```

```
36324
9082
```

In [ ]:

```
print(len(X[:len(y_train)]))
print(len(X[len(y_train):]))
```

```
36324
9082
```

3.1 Checking on single image

In [ ]:

```
X[0]
```

Out[ ]:

```
'/home/ubuntu/Project/my_data/Autopilot-TensorFlow-master/Autopilot-TensorFlow-
master/driving_dataset/0.jpg'
```

In [ ]:

```
img = cv2.imread(X[0])
print('Original_size of the image:', img.shape)
print("After taking the last 150 rows i.e lower part of the images where road is present, ",img[-1
50:].shape)
img = img[-150:]
resized_img = cv2.resize(img, (200,66))   # 200/66 = 455/150 = 3.0303 => we are keeping aspect ratio
when we are resizing it
print('shape after resizing it:', resized_img.shape)
```

```
Original_size of the image: (256, 455, 3)
After taking the last 150 rows i.e lower part of the images where road is present,  (150, 455, 3)
shape after resizing it: (66, 200, 3)
```

3.2 Doing the same for all images

In [ ]:

```
X_train = []
for i in X[:len(y_train)]:
    img = cv2.imread(i)
    img = img[-150:]
    resized_img = cv2.resize(img, (200,66))
    X_train.append(resized_img)
```

In [ ]:

```
X_train = np.array(X_train)
```

In [ ]:

```
import pickle
with open('X_train', 'wb') as f:
    pickle.dump(X_train, f)
```

In [ ]:

```
print(X_train.shape)
```

```
(36324, 66, 200, 3)
```

In [ ]:

```
X_val = []
for j in X[len(y_train):]:
    img = cv2.imread(i)
    img = img[-150:]
    resized_img = cv2.resize(img, (200,66))
    X_val.append(resized_img)
```

In [ ]:

```
X_val = np.array(X_val)
```

In [ ]:

```
import pickle
with open('X_val', 'wb') as f:
    pickle.dump(X_val, f)
```

In [ ]:

```
print(X_val.shape)
```

```
(9082, 66, 200, 3)
```

3.3 Reading from pickle file

In [ ]:

```
with open('/home/ubuntu/Project/my_data/X_train', 'rb') as f:
    X_train = pickle.load(f)
```

In [ ]:

```
with open('/home/ubuntu/Project/my_data/X_val', 'rb') as f:
    X_val = pickle.load(f)
```

In [ ]:

```
print(X_train.shape)
print(X_val.shape)
print(y_train.shape)
print(y_val.shape)
```

```
(36324, 66, 200, 3)
(9082, 66, 200, 3)
(36324,)
(9082,)
```

4. Preparing data for modelling

In [ ]:

```
img_height = X_train.shape[1]
img_width = X_train.shape[2]
channels = X_train.shape[3]

print(img_height, img_width, channels)
```

```
66 200 3
```

In [ ]:

```
import tensorflow.keras.backend as K

if K.image_data_format() == 'channels_first':
    X_train = X_train.reshape(X_train.shape[0], channels, img_height, img_width).astype('float32')
    X_val = X_val.reshape(X_val.shape[0], channels, img_height, img_width).astype('float32')
    img_size = (channels, img_height, img_width)

else:
    X_train = X_train.reshape(X_train.shape[0], img_height, img_width, channels).astype('float32')
    X_val = X_val.reshape(X_val.shape[0], img_height, img_width, channels).astype('float32')
    img_size = (img_height, img_width, channels)
```

In [ ]:

```
X_train = X_train/255.0
```

```
X_val = X_val/255.0
```

In [ ]:

```
print(X_train.shape)
print(X_val.shape)
print(img_size)
```

```
(36324, 66, 200, 3)
(9082, 66, 200, 3)
(66, 200, 3)
```

5. Building model from Nvidia Architecture

In [ ]:

```
#https://stackoverflow.com/questions/43915482/how-do-you-create-a-custom-activation-function-with-keras
from tensorflow.keras.backend as K

def custom_activation(x):
    return tf.multiply(tf.math.atan(x), 2) #  arctan range is [-pi/2, pi/2]. so we have to get the angle between [-pi, pi] so we are multiplying by 2
```

In [ ]:

```
#to refer the model architecture: https://developer.nvidia.com/blog/deep-learning-self-driving-cars/
#to use atan as a activation function refer: https://www.tensorflow.org/api_docs/python/tf/math/atan

img_input = tf.keras.Input(img_size)
x = tf.keras.layers.Conv2D(filters=24, kernel_size=(5,5), strides=(2,2), padding='valid', name='conv-1')(img_input)
x = tf.keras.layers.Conv2D(filters=36, kernel_size=(5,5), strides=(2,2), padding='valid', name='conv-2')(x)
x = tf.keras.layers.Conv2D(filters=48, kernel_size=(5,5), strides=(2,2), padding='valid', name='conv-3')(x)
x = tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='valid', name='conv-4')(x)
x = tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='valid', name='conv-5')(x)
x = tf.keras.layers.Flatten(name='flatten')(x)
x = tf.keras.layers.Dense(1164, activation='relu', name='dense_layer-1')(x)
x = tf.keras.layers.Dropout(0.2, name='dropout-1')(x)
x = tf.keras.layers.Dense(100, activation='relu', name='dense_layer-2')(x)
x = tf.keras.layers.Dropout(0.2, name='dropout-2')(x)
x = tf.keras.layers.Dense(50, activation='relu', name='dense_layer-3')(x)
x = tf.keras.layers.Dropout(0.2, name='dropout-3')(x)
x = tf.keras.layers.Dense(10, activation='relu', name='dense_layer-4')(x)
x = tf.keras.layers.Dropout(0.2, name='dropout-4')(x)
output = tf.keras.layers.Dense(1, activation=custom_activation, name='output')(x) #note we have used atan to use tan inverse activation

model_1 = tf.keras.models.Model(inputs=img_input, outputs=output, name='model-1')
```

In [ ]:

```
model_1.summary()
```

```
Model: "model-1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 66, 200, 3)]      0
_________________________________________________________________
conv-1 (Conv2D)              (None, 31, 98, 24)        1824
_________________________________________________________________
conv-2 (Conv2D)              (None, 14, 47, 36)        21636
_________________________________________________________________
```

```
conv-3 (Conv2D)              (None, 5, 22, 48)         43248
_________________________________________________________________
conv-4 (Conv2D)              (None, 3, 20, 64)         27712
_________________________________________________________________
conv-5 (Conv2D)              (None, 1, 18, 64)         36928
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense_layer-1 (Dense)        (None, 1164)              1342092
_________________________________________________________________
dropout-1 (Dropout)          (None, 1164)              0
_________________________________________________________________
dense_layer-2 (Dense)        (None, 100)               116500
_________________________________________________________________
dropout-2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_layer-3 (Dense)        (None, 50)                5050
_________________________________________________________________
dropout-3 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_layer-4 (Dense)        (None, 10)                510
_________________________________________________________________
dropout-4 (Dropout)          (None, 10)                0
_________________________________________________________________
output (Dense)               (None, 1)                 11
=================================================================
Total params: 1,595,511
Trainable params: 1,595,511
Non-trainable params: 0
_________________________________________________________________
```

6. Compiling and fit the model

In [ ]:

```
model_1.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
loss='mean_squared_error')
```

In [ ]:

```
batch_size = 100
Epochs = 30
```

In [ ]:

```
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

checkpoint_file_name = '/home/ubuntu/Project/my_data/checkpoint/first_run/weights.{epoch:02d}-{val
_loss:.2f}.hdf5'
model_checkpoint = ModelCheckpoint(checkpoint_file_name, monitor='val_loss', verbose=1, save_best_o
nly=True)

callbacks = [model_checkpoint]
```

In [ ]:

```
history = model_1.fit(X_train, y_train, batch_size=batch_size, epochs=Epochs, validation_data=(X_va
l, y_val), callbacks=callbacks, verbose=2)
```

```
Train on 36324 samples, validate on 9082 samples
Epoch 1/30

Epoch 00001: val_loss improved from inf to 0.23466, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.01-0.23.hdf5
36324/36324 - 5s - loss: 0.2927 - val_loss: 0.2347
Epoch 2/30

Epoch 00002: val_loss did not improve from 0.23466
36324/36324 - 4s - loss: 0.1991 - val_loss: 0.3899
Epoch 3/30
```

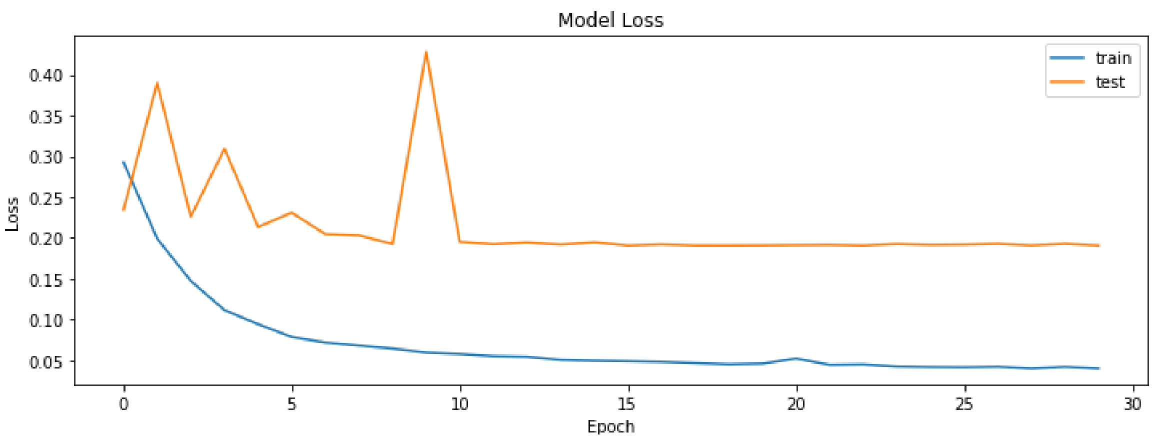
```
Epoch 00003: val_loss improved from 0.23466 to 0.22600, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.03-0.23.hdf5
36324/36324 - 4s - loss: 0.1473 - val_loss: 0.2260
Epoch 4/30

Epoch 00004: val_loss did not improve from 0.22600
36324/36324 - 4s - loss: 0.1115 - val_loss: 0.3093
Epoch 5/30

Epoch 00005: val_loss improved from 0.22600 to 0.21338, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.05-0.21.hdf5
36324/36324 - 4s - loss: 0.0942 - val_loss: 0.2134
Epoch 6/30

Epoch 00006: val_loss did not improve from 0.21338
36324/36324 - 4s - loss: 0.0786 - val_loss: 0.2309
Epoch 7/30

Epoch 00007: val_loss improved from 0.21338 to 0.20462, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.07-0.20.hdf5
36324/36324 - 4s - loss: 0.0717 - val_loss: 0.2046
Epoch 8/30

Epoch 00008: val_loss improved from 0.20462 to 0.20311, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.08-0.20.hdf5
36324/36324 - 4s - loss: 0.0680 - val_loss: 0.2031
Epoch 9/30

Epoch 00009: val_loss improved from 0.20311 to 0.19258, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.09-0.19.hdf5
36324/36324 - 4s - loss: 0.0644 - val_loss: 0.1926
Epoch 10/30

Epoch 00010: val_loss did not improve from 0.19258
36324/36324 - 4s - loss: 0.0594 - val_loss: 0.4283
Epoch 11/30

Epoch 00011: val_loss did not improve from 0.19258
36324/36324 - 4s - loss: 0.0577 - val_loss: 0.1952
Epoch 12/30

Epoch 00012: val_loss improved from 0.19258 to 0.19236, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.12-0.19.hdf5
36324/36324 - 4s - loss: 0.0551 - val_loss: 0.1924
Epoch 13/30

Epoch 00013: val_loss did not improve from 0.19236
36324/36324 - 4s - loss: 0.0543 - val_loss: 0.1944
Epoch 14/30

Epoch 00014: val_loss improved from 0.19236 to 0.19190, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.14-0.19.hdf5
36324/36324 - 4s - loss: 0.0505 - val_loss: 0.1919
Epoch 15/30

Epoch 00015: val_loss did not improve from 0.19190
36324/36324 - 4s - loss: 0.0495 - val_loss: 0.1946
Epoch 16/30

Epoch 00016: val_loss improved from 0.19190 to 0.19071, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.16-0.19.hdf5
36324/36324 - 4s - loss: 0.0489 - val_loss: 0.1907
Epoch 17/30

Epoch 00017: val_loss did not improve from 0.19071
36324/36324 - 4s - loss: 0.0481 - val_loss: 0.1920
Epoch 18/30

Epoch 00018: val_loss did not improve from 0.19071
36324/36324 - 4s - loss: 0.0467 - val_loss: 0.1907
Epoch 19/30

Epoch 00019: val_loss improved from 0.19071 to 0.19070, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.19-0.19.hdf5
36324/36324 - 4s - loss: 0.0451 - val_loss: 0.1907
Epoch 20/30
```

```
Epoch 00020: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0460 - val_loss: 0.1909
Epoch 21/30

Epoch 00021: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0522 - val_loss: 0.1912
Epoch 22/30

Epoch 00022: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0445 - val_loss: 0.1914
Epoch 23/30

Epoch 00023: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0450 - val_loss: 0.1907
Epoch 24/30

Epoch 00024: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0424 - val_loss: 0.1925
Epoch 25/30

Epoch 00025: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0418 - val_loss: 0.1914
Epoch 26/30

Epoch 00026: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0415 - val_loss: 0.1918
Epoch 27/30

Epoch 00027: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0421 - val_loss: 0.1928
Epoch 28/30

Epoch 00028: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0400 - val_loss: 0.1908
Epoch 29/30

Epoch 00029: val_loss did not improve from 0.19070
36324/36324 - 4s - loss: 0.0420 - val_loss: 0.1928
Epoch 30/30

Epoch 00030: val_loss improved from 0.19070 to 0.19070, saving model to
/home/ubuntu/Project/my_data/checkpoint/first_run/weights.30-0.19.hdf5
36324/36324 - 4s - loss: 0.0400 - val_loss: 0.1907
```

6.1 Model loss plot - using dropout

In [ ]:

```
# history plot for accuracy
plt.figure(figsize=(12,4))
plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.title("Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(["train", "test"])
plt.show()
```



Summary:

- As we can see from the base model we got Test_MSE(MEAN):0.191127 , Test_MSE(ZERO):0.190891 and here we got 0.1907 as mean_squared_error with dropout. This makes sense as that most of the time steering angle is 0 and thats why distribution is around with mean as zero.
- Even after so many epochs we can't reduce mean_squared_error further

7. Using the output prediction

In [ ]:

```
#https://stackoverflow.com/questions/55779286/valueerror-unknown-activation-function-my-custom-act
ivation-function
best_model_1 =
tf.keras.models.load_model('/home/ubuntu/Project/my_data/checkpoint/first_run/weights.30-
0.19.hdf5',
                                          custom_objects={'custom_activation':custom_activation})
all_pred = []

for i in X_val[1000:2000]:
    i = cv2.resize(i, (200,66))
    i = i/255.0
    i = np.expand_dims(i, axis=0)
    prediction = best_model_1.predict(i)
    all_pred.extend(prediction[0])
```

In [ ]:

```
(set(all_pred))
```

Out[ ]:

```
{-0.29648045}
```

In [ ]:

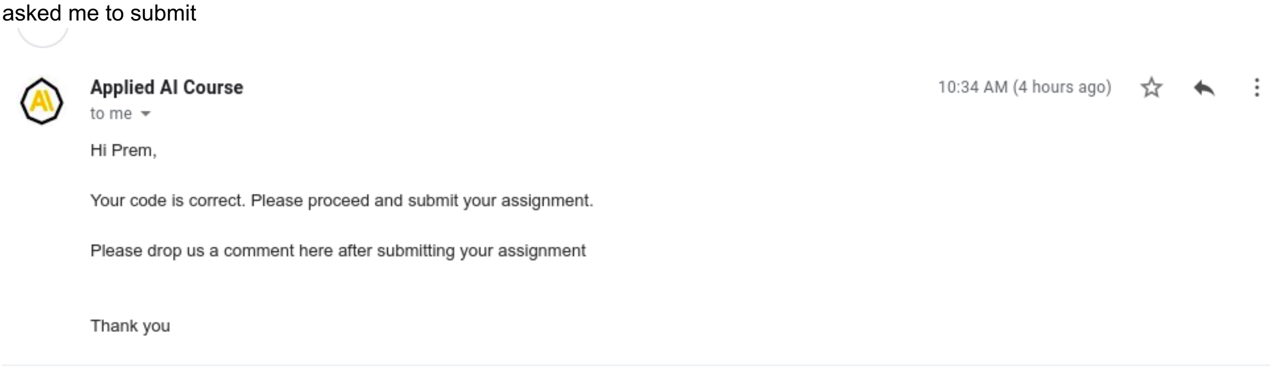
```
len(set(y_val[1000:2000]))
```

Out[ ]:

```
357
```

DOUBT:

- My model predict only one value as a degree and i don't know why. i mailed the team and they said my code is correct and asked me to submit


Applied AI Course
to me
10:34 AM (4 hours ago)

Hi Prem,

Your code is correct. Please proceed and submit your assignment.

Please drop us a comment here after submitting your assignment

Thank you

7.1 Making the video using the images we have so that we can rotate the steering using the video

In [ ]:

```
print(os.path.exists(X[0]))
print(type(X[0]))
```

```
True
<class 'str'>
```

In [ ]:

```
#https://theailearner.com/2018/10/15/creating-video-from-images-using-opencv-python/

#we have X which is a list contains the path of each image
image_array = []
for i in X:
    img = cv2.imread(i)
    image_array.append(img)

out = cv2.VideoWriter('/home/ubuntu/Project/my_data/Self_Driving_Car.avi', cv2.VideoWriter_fourcc(*
'DIVX'), 15, (img.shape[1], img.shape[0]))
for j in range(len(image_array)):
    out.write(image_array[j])
out.release()
```

7.2 Rotate the steering as per the prediction

In [ ]:

```
#to rotate the steering image refer https://opencv-python-
tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric
nsformations.html
#steering_image
model_1 = tf.keras.models.load_model('')
steering_img = cv2.imread('/home/ubuntu/Project/my_data/Autopilot-TensorFlow-master/Autopilot-Tens
orFlow-master/steering_wheel_image.jpg')
rows, cols = steering_img.shape[0], steering_img.shape[1]

cap = cv2.VideoCapture('/home/ubuntu/Project/my_data/Self_Driving_Car.avi')

smoothed_angle = 0
while True:
    ret, frame = cap.read()
    resized_frame = cv2.resize(frame, (200,66))
    resized_frame = resized_frame/255.0
    resized_frame = np.expand_dims(axis=0)
    degrees = (model_1.predict(resized_frame) * 180)/scipy.pi  #to convert the radians to degrees
    cv2.imshow('frame', frame)
    #make smooth angle transitions by turning the steering wheel based on the difference of the cu
rrent angle
    #and the predicted angle
    smoothed_angle += 0/2 * pow(abs(degrees - smoothed_angle), 2.0/3.0) * (degrees-smoothed_angle)
/ abs(degrees-smoothed_angle)
    M = cv2.getRotationMatrix2D((col2/2, rows/2), -degrees, 1)
    dst = cv2.warpAffine(steering_img, M, (cols,rows))
    cv2.imshow('steering_wheel', dst)

    if cv2.waitKey(1)==27:
        break

cv2.destroyAllWindows()
cap.release()
```

8. Assignment:

- split ---> 70%training and 30% testing
- adam optimizer ---> learning_rate = 0.001
- dropout ---> 0.5
- activation --> 'linear' instead of atan

In [ ]:

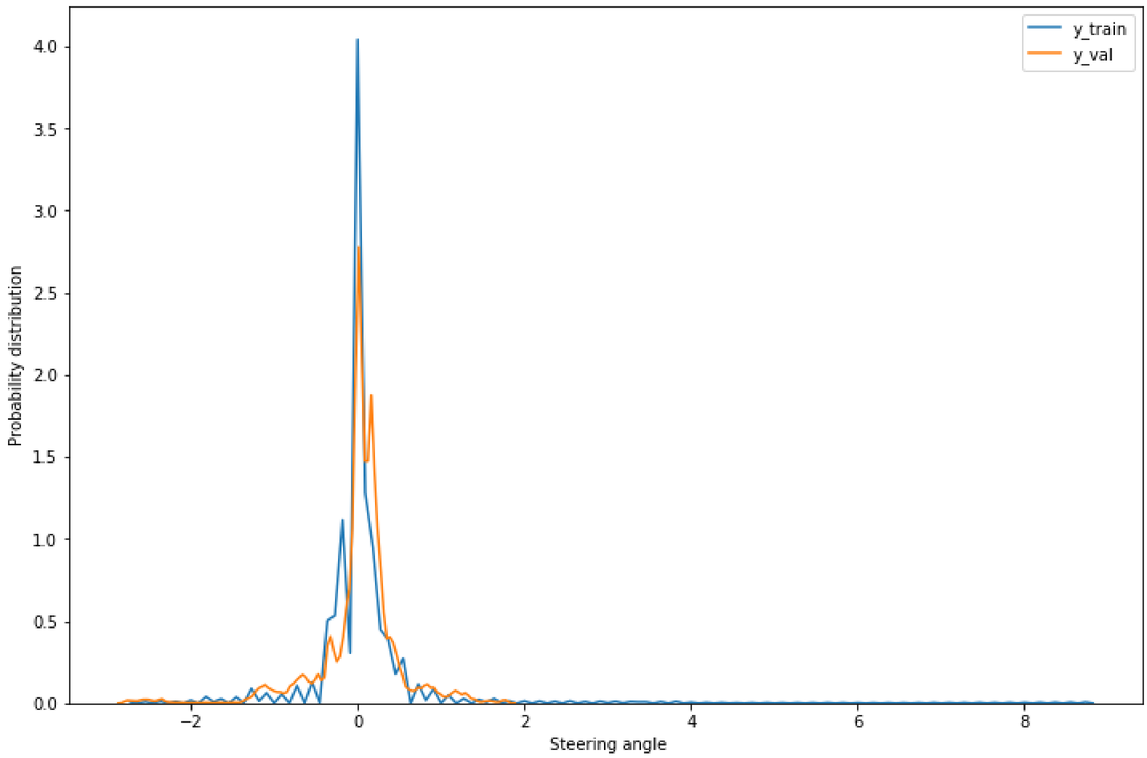
```
#reading the data set from data.txt

dataset_path = '/home/ubuntu/Project/my_data/Autopilot-TensorFlow-master/Autopilot-TensorFlow-mast
er/driving_dataset/'
training_path = os.path.join(dataset_path, 'data.txt')

split = 0.7
X = []
y = []

#islice(text, start, stop, LIMIT) --> ('ABCDEF', 2, None) --> 'C D E F'  --
>https://docs.python.org/2/library/itertools.html
with open(training_path) as f:
    for line in islice(f, None):
        path, angle = line.strip().split()
        full_path = os.path.join(dataset_path, path)
        X.append(full_path)
        y.append(float(angle)*pi/180)

y = np.array(y)
print('Finished processing')

#Split it with 70/30
split_index = int(len(y)*0.7)
y_train = y[:split_index]
y_val = y[split_index:]
```

```
Finished processing
```

8.1 Base Model

In [ ]:

```
#distribution of train and test y
plt.figure(figsize=(12,8))
sns.distplot(y_train, hist=False, label='y_train')
sns.distplot(y_val, hist=False, label='y_val')
plt.xlabel(' Steering angle')
plt.ylabel('Probability distribution')
plt.legend()
plt.show()
```



In [ ]:

```
#Model 0: Base line Model: y_test_pred = mean(y_train_i)
train_mean_y = np.mean(y_train)

print('Test_MSE(MEAN):%f' % np.mean(np.square(y_val-train_mean_y)) )
print('Test_MSE(ZERO):%f' % np.mean(np.square(y_val-0.0)) )
```

```
Test_MSE(MEAN):0.241561
Test_MSE(ZERO):0.241107
```

8.2. Reading images from the path

In [ ]:

```
dataset_path = '/home/ubuntu/Project/my_data/Autopilot-TensorFlow-master/Autopilot-TensorFlow-mast
er/driving_dataset/'
os.path.exists(os.path.join(dataset_path, 'data.txt'))
```

Out[ ]:

```
True
```

In [ ]:

```
print(len(X[:len(y_train)]))
print(len(X[len(y_train):]))
```

```
31784
13622
```

8.2.1 Checking on single image

In [ ]:

```
X[0]
```

Out[ ]:

```
'/home/ubuntu/Project/my_data/Autopilot-TensorFlow-master/Autopilot-TensorFlow-
master/driving_dataset/0.jpg'
```

In [ ]:

```
img = cv2.imread(X[0])
print('Original_size of the image:', img.shape)
print("After taking the last 150 rows i.e lower part of the images where road is present, ",img[-1
50:].shape)
img = img[-150:]
resized_img = cv2.resize(img, (200,66))   # 200/66 = 455/150 = 3.0303 => we are keeping aspect ratio
when we are resizing it
print('shape after resizing it:', resized_img.shape)
```

```
Original_size of the image: (256, 455, 3)
After taking the last 150 rows i.e lower part of the images where road is present,  (150, 455, 3)
shape after resizing it: (66, 200, 3)
```

8.2..2 Doing the same for all images

In [ ]:

```
X_train = []
for i in X[:len(y_train)]:
    img = cv2.imread(i)
    img = img[-150:]
```

```
    resized_img = cv2.resize(img, (200,66))
    X_train.append(resized_img)
```

In [ ]:

```
X_train = np.array(X_train)
```

In [ ]:

```
import pickle
with open('assign_train', 'wb') as f:
    pickle.dump(X_train, f)
```

In [ ]:

```
print(X_train.shape)
```

```
(31784, 66, 200, 3)
```

In [ ]:

```
X_val = []
for j in X[len(y_train):]:
    img = cv2.imread(i)
    img = img[-150:]
    resized_img = cv2.resize(img, (200,66))
    X_val.append(resized_img)
```

In [ ]:

```
X_val = np.array(X_val)
```

In [ ]:

```
import pickle
with open('assign_val', 'wb') as f:
    pickle.dump(X_val, f)
```

In [ ]:

```
print(X_val.shape)
```

```
(13622, 66, 200, 3)
```

8.3 Reading from pickle file

In [ ]:

```
with open('/home/ubuntu/Project/my_data/assign_train', 'rb') as f:
    X_train = pickle.load(f)
```

In [ ]:

```
with open('/home/ubuntu/Project/my_data/assign_val', 'rb') as f:
    X_val = pickle.load(f)
```

In [ ]:

```
print(X_train.shape)
print(X_val.shape)
print(y_train.shape)
print(y_val.shape)
```

```
(31784, 66, 200, 3)
```

```
(13622, 66, 200, 3)
(36324,)
(9082,)
```

8.4. Preparing data for modelling

In [ ]:

```
img_rows = X_train.shape[1]
img_cols = X_train.shape[2]
channels = X_train.shape[3]
```

In [ ]:

```
import tensorflow.keras.backend as K

if K.image_data_format() == 'channels_first':
    X_train = X_train.reshape(X_train.shape[0], channels, img_rows, img_cols).astype('float32')
    X_val = X_val.reshape(X_val.shape[0], channels, img_rows, img_cols).astype('float32')
    img_size = (channels, img_rows, img_cols)

else:
    X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols, channels).astype('float32')
    X_val = X_val.reshape(X_val.shape[0], img_rows, img_cols, channels).astype('float32')
    img_size = (img_rows, img_cols, channels)
```

In [ ]:

```
X_train = X_train/255.0
X_val = X_val/255.0
```

In [ ]:

```
print(X_train.shape)
print(y_train.shape)
print(X_val.shape)
print(y_val.shape)
print(img_size)
```

```
(31784, 66, 200, 3)
(31784,)
(13622, 66, 200, 3)
(13622,)
(66, 200, 3)
```

8.5. Building model from Nvidia Architecture

In [ ]:

```
import tensorflow as tf
```

In [ ]:

```
#https://developer.nvidia.com/blog/deep-learning-self-driving-cars/

img_input = tf.keras.Input(img_size)
x = tf.keras.layers.Conv2D(filters=24, kernel_size=(5,5), strides=(2,2), padding='valid', name='con
v-1')(img_input)
x = tf.keras.layers.Conv2D(filters=36, kernel_size=(5,5), strides=(2,2), padding='valid', name='con
v-2')(x)
x = tf.keras.layers.Conv2D(filters=48, kernel_size=(5,5), strides=(2,2), padding='valid', name='con
v-3')(x)
x = tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='valid', name='con
v-4')(x)
x = tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='valid', name='con
v-5')(x)
x = tf.keras.layers.Flatten(name='flatten')(x)
x = tf.keras.layers.Dense(1164, activation='relu', name='dense_layer-1')(x)
```

```
x = tf.keras.layers.Dense(1164, activation='relu', name='dense_layer-1')(x)
x = tf.keras.layers.Dropout(0.5, name='dropout-1')(x)
x = tf.keras.layers.Dense(100, activation='relu', name='dense_layer-2')(x)
x = tf.keras.layers.Dropout(0.5, name='dropout-2')(x)
x = tf.keras.layers.Dense(50, activation='relu', name='dense_layer-3')(x)
x = tf.keras.layers.Dropout(0.5, name='dropout-3')(x)
x = tf.keras.layers.Dense(10, activation='relu', name='dense_layer-4')(x)
x = tf.keras.layers.Dropout(0.5, name='dropout-4')(x)
output = tf.keras.layers.Dense(1, activation=tf.keras.activations.linear, name='output')(x) #note w
e have used atan to use tan inverse activation

model_2 = tf.keras.models.Model(inputs=img_input, outputs=output, name='model-2')
```

In [ ]:

```
model_2.summary()
```

```
Model: "model-2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 66, 200, 3)]      0
_________________________________________________________________
conv-1 (Conv2D)              (None, 31, 98, 24)        1824
_________________________________________________________________
conv-2 (Conv2D)              (None, 14, 47, 36)        21636
_________________________________________________________________
conv-3 (Conv2D)              (None, 5, 22, 48)         43248
_________________________________________________________________
conv-4 (Conv2D)              (None, 3, 20, 64)         27712
_________________________________________________________________
conv-5 (Conv2D)              (None, 1, 18, 64)         36928
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense_layer-1 (Dense)        (None, 1164)              1342092
_________________________________________________________________
dropout-1 (Dropout)          (None, 1164)              0
_________________________________________________________________
dense_layer-2 (Dense)        (None, 100)               116500
_________________________________________________________________
dropout-2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_layer-3 (Dense)        (None, 50)                5050
_________________________________________________________________
dropout-3 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_layer-4 (Dense)        (None, 10)                510
_________________________________________________________________
dropout-4 (Dropout)          (None, 10)                0
_________________________________________________________________
output (Dense)               (None, 1)                 11
=================================================================
Total params: 1,595,511
Trainable params: 1,595,511
Non-trainable params: 0
_________________________________________________________________
```

8.6 Compiling and fit the model

In [ ]:

```
model_2.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
loss='mean_squared_error')
```

In [ ]:

```
batch_size = 100
Epochs = 30
```

In [ ]:

```python
from datetime import datetime
from tensorflow.keras.callbacks import  ModelCheckpoint

checkpoint_file_name = '/home/ubuntu/Project/my_data/checkpoint/assign/July_01-weights.{epoch:02d}
-{val_loss:.2f}.hdf5'
model_checkpoint = ModelCheckpoint(checkpoint_file_name, monitor='val_loss', verbose=1, save_best_o
nly=True)

callbacks = [model_checkpoint]
```

In [ ]:

```python
history_2 = model_2.fit(X_train, y_train, batch_size=batch_size, epochs=Epochs, validation_data=(X_
val, y_val), callbacks=callbacks, verbose=2)
```

```
Train on 31784 samples, validate on 13622 samples
Epoch 1/30

Epoch 00001: val_loss improved from inf to 0.24105, saving model to
/home/ubuntu/Project/my_data/checkpoint/assign/July_01-weights.01-0.24.hdf5
31784/31784 - 9s - loss: 0.3171 - val_loss: 0.2410
Epoch 2/30

Epoch 00002: val_loss improved from 0.24105 to 0.24098, saving model to
/home/ubuntu/Project/my_data/checkpoint/assign/July_01-weights.02-0.24.hdf5
31784/31784 - 4s - loss: 0.3017 - val_loss: 0.2410
Epoch 3/30

Epoch 00003: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.2919 - val_loss: 0.2411
Epoch 4/30

Epoch 00004: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.2822 - val_loss: 0.2411
Epoch 5/30

Epoch 00005: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.2548 - val_loss: 0.2450
Epoch 6/30

Epoch 00006: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.2374 - val_loss: 0.2661
Epoch 7/30

Epoch 00007: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.2189 - val_loss: 0.2567
Epoch 8/30

Epoch 00008: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.2144 - val_loss: 0.2802
Epoch 9/30

Epoch 00009: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1926 - val_loss: 0.2757
Epoch 10/30

Epoch 00010: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1918 - val_loss: 0.2620
Epoch 11/30

Epoch 00011: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1812 - val_loss: 0.3093
Epoch 12/30

Epoch 00012: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1750 - val_loss: 0.2836
Epoch 13/30

Epoch 00013: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1668 - val_loss: 0.2870
Epoch 14/30

Epoch 00014: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1685 - val_loss: 0.3186
Epoch 15/30
```

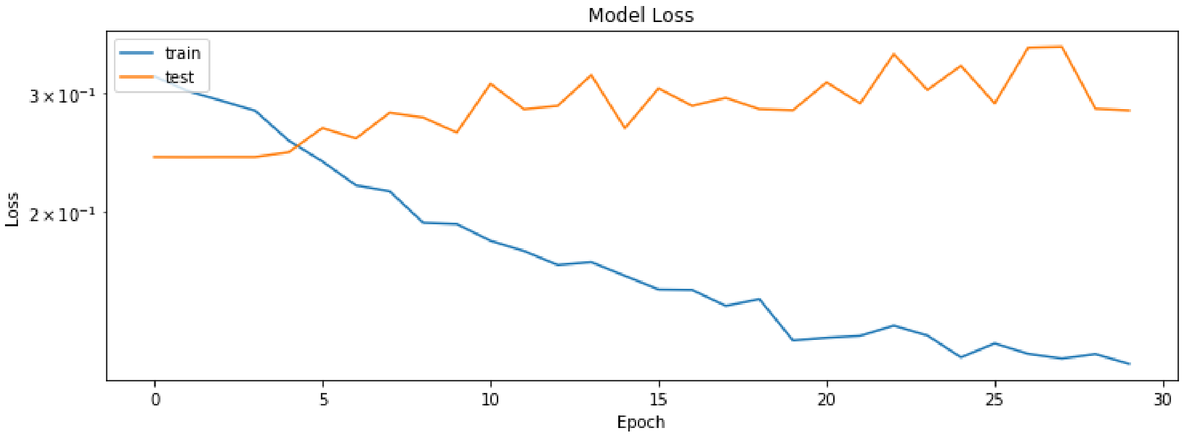
```
Epoch 00015: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1608 - val_loss: 0.2658
Epoch 16/30

Epoch 00016: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1535 - val_loss: 0.3045
Epoch 17/30

Epoch 00017: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1533 - val_loss: 0.2869
Epoch 18/30

Epoch 00018: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1452 - val_loss: 0.2948
Epoch 19/30

Epoch 00019: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1486 - val_loss: 0.2838
Epoch 20/30

Epoch 00020: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1291 - val_loss: 0.2826
Epoch 21/30

Epoch 00021: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1302 - val_loss: 0.3108
Epoch 22/30

Epoch 00022: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1311 - val_loss: 0.2893
Epoch 23/30

Epoch 00023: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1357 - val_loss: 0.3421
Epoch 24/30

Epoch 00024: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1312 - val_loss: 0.3028
Epoch 25/30

Epoch 00025: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1218 - val_loss: 0.3288
Epoch 26/30

Epoch 00026: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1278 - val_loss: 0.2893
Epoch 27/30

Epoch 00027: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1233 - val_loss: 0.3499
Epoch 28/30

Epoch 00028: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1213 - val_loss: 0.3509
Epoch 29/30

Epoch 00029: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1232 - val_loss: 0.2841
Epoch 30/30

Epoch 00030: val_loss did not improve from 0.24098
31784/31784 - 4s - loss: 0.1192 - val_loss: 0.2824
```

Summary:

- As we can see from the base model we got test_mse with mean as a prediction we got 0.241561 and Test_MSE(ZERO):0.241107 and here we got 0.24098 as mean_squared_error with dropout. This makes our model works better than our base model.
- Even after so many epochs we can't reduce mean_squared_error further

```
In [ ]:
```

```
# history plot for accuracy
plt.figure(figsize=(12,4))
plt.plot(history_2.history["loss"])
plt.plot(history_2.history["val_loss"])
plt.title("Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.yscale('log')
plt.legend(["train", "test"], loc="upper left")
plt.show()
```



8.7 Predict using best model

In [ ]:

```
best_model_2 = tf.keras.models.load_model('/home/ubuntu/Project/my_data/checkpoint/assign/July_01-weights.02-0.24.hdf5')

all_pred = []
for i in X_val[1000:2000]:
    i = cv2.resize(i, (200,66))
    i = i/255.0
    i = np.expand_dims(i, axis=0)
    prediction = best_model_2.predict(i)
    all_pred.extend(prediction[0])
```

In [ ]:

```
set(all_pred)
```

Out[ ]:

```
{0.0062604654}
```

In [ ]:

```
len(set(y_val[1000:2000]))
```

Out[ ]:

```
518
```

DOUBT:

- Here also we get predicted the same value and i dont know why

8.8 Using the output prediction

In [ ]:

```
#to rotate the steering image refer https://opencv-python-
tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometri
nsformations.html
#steering_image
model_2 = tf.keras.models.load_model('')
steering_img = cv2.imread('')
rows, cols = steering_image.shape

cap = cv2.VideoCapture('')

smooth_angle = 0
while True:
    ret, frame = cap.read()
    resized_frame = cv2.resize(frame, (200,66))
    resized_frame = resized_frame/255.0
    resized_frame = np.expand_dims(axis=0)
    degrees = (best_model_2.predict(resized_frame) * 180)/scipy.pi  #to convert the radians to degr
ees
    cv2.imshow('frame', frame)
    #make smooth angle transitions by turning the steering wheel based on the difference of the cu
rrent angle
    #and the predicted angle
    smooth_angle += 0/2 * pow(abs(degees - smoothed_angle), 2.0/3.0) * (degrees-smoothed_angle) /
abs(degrees-smoothed_angle)
    M = cv2.getRotationMatrix2D(center=(col2/2, rows/2), angle=-degrees, 1)
    dst = cv2.warpAffine(steering_img, M, (cols,rows))
    cv2.imshow('steering_wheel', dst)

    if cv2.waitKey()==27:
        break

cv2.destroyAllWindows()
cap.release()
```