```
Vb7Em2cW?e=download&authuser=0&nonce=bb755n5u3r77m&user=00955149336419715002&hash=g9ipkfqc
```

```
--2020-01-14 14:54:03--  https://doc-04-1c-docs.googleusercontent.com/docs/securesc/e
Resolving doc-04-1c-docs.googleusercontent.com (doc-04-1c-docs.googleusercontent.com)
Connecting to doc-04-1c-docs.googleusercontent.com (doc-04-1c-docs.googleusercontent.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/rar]
Saving to: ‘Autopilot-TensorFlow-master.rar’

Autopilot-TensorFlo     [<=>                 ]   2.22G  62.7MB/s    in 35s

2020-01-14 14:54:38 (65.7 MB/s) - ‘Autopilot-TensorFlow-master.rar’ saved [2380979121
```

```
!pip3 install patool
import patoolib
patoolib.extract_archive("Autopilot-TensorFlow-master.rar")
```

```
Collecting patool
  Downloading https://files.pythonhosted.org/packages/43/94/52243ddff508780dd2d811096
     |████████████████████████████████| 81kB 6.0MB/s
Installing collected packages: patool
Successfully installed patool-1.12
patool: Extracting Autopilot-TensorFlow-master.rar ...
patool: running /usr/bin/unrar x -- /content/Autopilot-TensorFlow-master.rar
patool:     with cwd='./Unpack_5lagpfeg'
patool: ... Autopilot-TensorFlow-master.rar extracted to `Autopilot-TensorFlow-master
'Autopilot-TensorFlow-master'
```

```
%cd ./Autopilot-TensorFlow-master
%cd ./Autopilot-TensorFlow-master
```

```
[Errno 2] No such file or directory: './Autopilot-TensorFlow-master'
/content/Autopilot-TensorFlow-master/Autopilot-TensorFlow-master
[Errno 2] No such file or directory: './Autopilot-TensorFlow-master'
/content/Autopilot-TensorFlow-master/Autopilot-TensorFlow-master
```

```
from google.colab import files
files.upload()
```

Choose Files data.txt

- **data.txt**(text/plain) - 888817 bytes, last modified: 12/11/2019 - 100% done

```
Saving data.txt to data.txt
{'data.txt': b'0.jpg 0.000000\n1.jpg 0.000000\n2.jpg 0.000000\n3.jpg 0.000000\n4.jpg
```

```
!ls
```

```
cmd_output        License.txt   README.md       'save copy'
data.txt          logs          run_dataset.py   Self_driving_car.ipynb
driving_data.py   model.py      run.py           steering_wheel_image.jpg
driving_dataset   __pycache__   save             train.py
```

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import pi
import cv2
import scipy.misc
import tensorflow as tf
import imageio
```

```
The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %te
```

```
#https://github.com/gauravtheP/Self-Driving-Car/blob/master/SelfDrivingCar/Self-Driving-Ca

x = []
y = []


with open('data.txt') as f:
  for line in f:
    image_name, angle = line.split()
    image_path = os.path.join('driving_dataset', image_name)
    x.append(image_path)
    angle_radians = float(angle) * (pi / 180)  #converting angle into radians
    y.append(angle_radians)
y = np.array(y)
print(str(len(x))+" "+str(len(y)))
```

```
45406 45406
```

```
split_ratio = int(len(x) * 0.8)

train_x = x[:split_ratio]
train_y = y[:split_ratio]

test_x = x[split_ratio:]
test_y = y[split_ratio:]

len(train_x), len(train_y), len(test_x), len(test_y)
```

```
(36324, 36324, 9082, 9082)
```

```
train_batch_pointer = 0
test_batch_pointer = 0


def loadTrainBatch(batch_size):
    global train_batch_pointer
    x_result = []
    y_result = []
    for i in range(batch_size):
```

```
    for i in range(batch_size):
        read_image = imageio.imread(train_x[(train_batch_pointer + i) % len(train_x)]) #he
        #"train_batch_pointer + i" should not cross the number of train images. As soon as
        #equal to number of train images then it will again start reading the train images
        #index onwards.
        read_image_road = read_image[-150:] #here, we are taking only the lower part of th
        #image. As, we are concern only with the curves of the road to predict angles so t
        #part of the image. Hence, here -"150" is equivalent to the last 150 matrix pixels
        read_image_resize = cv2.resize(read_image_road, (200, 66)) #After, resizing, each
        #now since we have kept only the last 150 matrices in the image so the size of our
        #Now 455/150 = 3.0303. Also 200/66 = 3.0303. Hence, here we are keeping the aspect
        read_image_final = read_image_resize/255.0  #here, we are normalizing the images

        x_result.append(read_image_final) #finally appending the image pixel matrix

        y_result.append(train_y[(train_batch_pointer + i) % len(train_y)]) #appending corr

    train_batch_pointer += batch_size

    return x_result, y_result


def loadTestBatch(batch_size):
    global test_batch_pointer
    x_result = []
    y_result = []
    for i in range(batch_size):
        read_image =imageio.imread(test_x[(test_batch_pointer + i) % len(test_x)]) #here %
        #"test_batch_pointer + i" should not cross the number of test images. As soon as t
        #equal to number of test images then it will again start reading the test images f
        #index onwards.
        read_image_road = read_image[-150:] #here, we are taking only the lower part of th
        #image. As, we are concern only with the curves of the road to predict angles so t
        #part of the image. Hence, here -"150" is equivalent to the last 150 matrix pixels
        read_image_resize = cv2.resize(read_image_road, (200, 66)) #After, resizing, each
        #now since we have kept only the last 150 matrices in the image so the size of our
        #Now 455/150 = 3.0303. Also 200/66 = 3.0303. Hence, here we are keeping the aspect
        read_image_final = read_image_resize/255.0  #here, we are normalizing the images

        x_result.append(read_image_final) #finally appending the image pixel matrix

        y_result.append(test_y[(test_batch_pointer + i) % len(test_y)]) #appending corresp

    test_batch_pointer += batch_size

    return x_result, y_result


def weightVariable(shape):
    initial = tf.truncated_normal(shape = shape, stddev = 0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)

def convolution(previous_input, filter_input, strides):
```

```
    return tf.nn.conv2d(previous_input, filter_input, strides = [1, strides, strides, 1],



x_input = tf.placeholder(tf.float32, shape = [None, 66, 200, 3], name = "Plc_1")
y_true = tf.placeholder(tf.float32, name = "Plc_2")

input_image = x_input


#Convolution Layers
#First convolution layer
W_Conv1 = weightVariable([5,5,3,24])
B_Conv1 = bias_variable([24])
Conv1 = tf.nn.relu(convolution(input_image, W_Conv1, 2) + B_Conv1)
#strides = 2
#Output size: 31*98*24

#Second convolution layer
W_Conv2 = weightVariable([5,5,24,36])
B_Conv2 = bias_variable([36])
Conv2 = tf.nn.relu(convolution(Conv1, W_Conv2, 2) + B_Conv2)
#strides = 2
#Output size: 14*47*36

#Third convolution layer
W_Conv3 = weightVariable([5,5,36,48])
B_Conv3 = bias_variable([48])
Conv3 = tf.nn.relu(convolution(Conv2, W_Conv3, 2) + B_Conv3)
#strides = 2
#Output size: 5*22*48

#Fourth convolution layer
W_Conv4 = weightVariable([3,3,48,64])
B_Conv4 = bias_variable([64])
Conv4 = tf.nn.relu(convolution(Conv3, W_Conv4, 1) + B_Conv4)
#strides = 1
#Output size: 3*20*64



#Fifth convolution layer
W_Conv5 = weightVariable([3,3,64,64])
B_Conv5 = bias_variable([64])
Conv5 = tf.nn.relu(convolution(Conv4, W_Conv5, 1) + B_Conv5)
#strides = 1
#Output size: 1*18*64

#Fully-Connected Dense Layers
keep_prob = tf.placeholder(tf.float32)
#First FC-Dense
#Input = 1*18*64 = 1152
W_FC1 = weightVariable([1152, 1164])
B_FC1 = bias_variable([1164])
FC1_Flatten = tf.reshape(Conv5, [-1, 1152]) #here, -1 indicates 1. It means that the shape
Output_FC1 = tf.nn.relu(tf.matmul(FC1_Flatten, W_FC1) + B_FC1) #so, here shape of FC1_Flat
#be 1152*1164. Therefore, there will be a matrix multiplication of matrices: (1*1152) * (1
```

```
Output_FC1_drop = tf.nn.dropout(Output_FC1, keep_prob)

#Second FC-Dense
#Input = 1*1164 = 1164
W_FC2 = weightVariable([1164, 100])
B_FC2 = bias_variable([100])
Output_FC2 = tf.nn.relu(tf.matmul(Output_FC1_drop, W_FC2) + B_FC2) #so, here shape of Outp
#W_FC2 will be 1164*100. Therefore, there will be a matrix multiplication of matrices: (1*
Output_FC2_drop = tf.nn.dropout(Output_FC2, keep_prob)

#Third FC-Dense
#Input = 1*100 = 100
W_FC3 = weightVariable([100, 50])
B_FC3 = bias_variable([50])
Output_FC3 = tf.nn.relu(tf.matmul(Output_FC2_drop, W_FC3) + B_FC3) #so, here shape of Outp
#W_FC3 will be 100*50. Therefore, there will be a matrix multiplication of matrices: (1*10
Output_FC3_drop = tf.nn.dropout(Output_FC3, keep_prob)

#Fourth FC-Dense
#Input = 1*50 = 50
W_FC4 = weightVariable([50, 10])
B_FC4 = bias_variable([10])
Output_FC4 = tf.nn.relu(tf.matmul(Output_FC3_drop, W_FC4) + B_FC4) #so, here shape of Outp
#W_FC4 will be 50*10. Therefore, there will be a matrix multiplication of matrices: (1*50)
Output_FC4_drop = tf.nn.dropout(Output_FC4, keep_prob)

#Final Output to one neuron with linear/identity function
#Input = 1*10 = 10
W_FC5 = weightVariable([10, 1])
B_FC5 = bias_variable([1])
y_predicted = tf.identity(tf.matmul(Output_FC4_drop, W_FC5) + B_FC5) #so, here shape of Ou
#W_FC5 will be 10*1. Therefore, there will be a matrix multiplication of matrices: (1*10)
#regression problem so we have applied identity fuction in the end. We can also apply "ata
#power is available then the model should be tested with both identity and atan functions.
#considered which gives better result.
```

```
WARNING:tensorflow:From <ipython-input-17-f3c5b10d7747>:45: calling dropout (from ten
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob
```

```
SAVEDIR = "../Saver/"
sess = tf.InteractiveSession()

L2NormConst = 0.001
train_vars = tf.trainable_variables() #it will return all the variables. Here, all the wei
#are trainable.

loss = tf.reduce_mean(tf.square(tf.subtract(y_true, y_predicted))) + tf.add_n([tf.nn.l2_lo
#since this is a regression problem so above loss is mean-squared-error loss
train_step = tf.train.AdamOptimizer(learning_rate = 10**-4).minimize(loss)
sess.run(tf.global_variables_initializer())

saver = tf.train.Saver()
```

```
epochs = 30
batch_size = 100
epoch_number, train_loss, test_loss,  = [], [], []

for epoch in range(epochs):
    train_avg_loss = 0
    test_avg_loss = 0
    te_loss_old = 10000  #any big number can be given

    for i in range(int(len(x)/batch_size)):
        train_batch_x, train_batch_y = loadTrainBatch(batch_size)
        train_step.run(feed_dict = {x_input: train_batch_x, y_true: train_batch_y, keep_pr
        tr_loss = loss.eval(feed_dict = {x_input: train_batch_x, y_true: train_batch_y, ke
        train_avg_loss += tr_loss / batch_size

        test_batch_x, test_batch_y = loadTestBatch(batch_size)
        te_loss_new = loss.eval(feed_dict = {x_input: test_batch_x, y_true: test_batch_y,
        test_avg_loss += te_loss_new / batch_size

        if te_loss_new < te_loss_old:
            print("Epoch: {}, Train_Loss: {}, Test_Loss: {} *".format(epoch+1, tr_loss, te
        else:
            print("Epoch: {}, Train_Loss: {}, Test_Loss: {}".format(epoch+1, tr_loss, te_l
        te_loss_old = te_loss_new

        if (i+1) % batch_size == 0:
            if not os.path.exists(SAVEDIR):
                os.makedirs(SAVEDIR)
            save_path = os.path.join(SAVEDIR, "model.ckpt")
            saver.save(sess = sess, save_path = save_path)
            print("Model saved at location {} at epoch {}".format(save_path, epoch + 1))

    epoch_number.append(epoch)
    train_loss.append(train_avg_loss)
    test_loss.append(test_avg_loss)

#creating dataframe and record all the losses and accuracies at each epoch
log_frame = pd.DataFrame(columns = ["Epoch", "Train Loss", "Test Loss"])
log_frame["Epoch"] = epoch_number
log_frame["Train Loss"] = train_loss
log_frame["Test Loss"] = test_loss
log_frame.to_csv(os.path.join(SAVEDIR, "log.csv"), index = False)
```

```
Epoch: 9, Train_Loss: 0.9199235439300537, Test_Loss: 0.9623548984527588
Epoch: 9, Train_Loss: 0.9216077327728271, Test_Loss: 0.9829785823822021
Epoch: 9, Train_Loss: 0.9236642718315125, Test_Loss: 1.0099233388900757
Epoch: 9, Train_Loss: 0.9309858083724976, Test_Loss: 0.9327641129493713 *
Epoch: 9, Train_Loss: 0.9270564913749695, Test_Loss: 0.937228798866272
Epoch: 9, Train_Loss: 0.9474039673805237, Test_Loss: 0.9087814092636108 *
Epoch: 9, Train_Loss: 0.9170969724655151, Test_Loss: 0.9083108305931091 *
Epoch: 9, Train_Loss: 0.9154853820800781, Test_Loss: 0.9097443222999573
Epoch: 9, Train_Loss: 0.9335434436798096, Test_Loss: 0.9108772873878479
Epoch: 9, Train_Loss: 0.9103251099586487, Test_Loss: 0.9091949462890625 *
Epoch: 9, Train_Loss: 0.9096845388412476, Test_Loss: 0.9089282155036926 *
Epoch: 9, Train_Loss: 0.9174407720565796, Test_Loss: 0.9098047614097595
Epoch: 9, Train_Loss: 0.9519089460372925, Test_Loss: 0.9073426127433777 *
Epoch: 9, Train_Loss: 0.9368179440498352, Test_Loss: 0.9092163443565369
Epoch: 9, Train_Loss: 0.9079607725143433, Test_Loss: 0.9182690382003784
Epoch: 9, Train_Loss: 0.9072350263595581, Test_Loss: 0.9088107347488403 *
Epoch: 9, Train_Loss: 0.9583017826080322, Test_Loss: 0.9305936694145203
Epoch: 9, Train_Loss: 0.9860822558403015, Test_Loss: 0.9217752814292908 *
Epoch: 9, Train_Loss: 0.9542723298072815, Test_Loss: 1.2445274591445923
Epoch: 9, Train_Loss: 0.9426730871200562, Test_Loss: 1.2834700345993042
Epoch: 9, Train_Loss: 0.9571837186813354, Test_Loss: 1.0112183094024658 *
Epoch: 9, Train_Loss: 0.9929379820823669, Test_Loss: 0.912960946559906 *
Epoch: 9, Train_Loss: 0.9556196331977844, Test_Loss: 0.9193041920661926
Epoch: 9, Train_Loss: 0.9607957601547241, Test_Loss: 0.9467144012451172
Epoch: 9, Train_Loss: 1.0608136653900146, Test_Loss: 1.0823421478271484
Epoch: 9, Train_Loss: 0.9401500225067139, Test_Loss: 1.8302123546600342
Model saved at location ../Saver/model.ckpt at epoch 9
Epoch: 9, Train_Loss: 0.9244081377983093, Test_Loss: 1.6685453653335571 *
Epoch: 9, Train_Loss: 0.9074921607971191, Test_Loss: 0.9506394863128662 *
Epoch: 9, Train_Loss: 0.9071184992790222, Test_Loss: 0.949327290058136 *
Epoch: 9, Train_Loss: 0.905782163143158, Test_Loss: 0.9070402979850769 *
Epoch: 9, Train_Loss: 0.9068833589553833, Test_Loss: 0.9178903698921204
Epoch: 9, Train_Loss: 0.9065029621124268, Test_Loss: 0.9141891002655029 *
Epoch: 9, Train_Loss: 4.734259128570557, Test_Loss: 0.9180861115455627
Epoch: 9, Train_Loss: 2.1017584800720215, Test_Loss: 0.9754448533058167
Epoch: 9, Train_Loss: 0.9004679322242737, Test_Loss: 0.9051167368888855 *
Epoch: 9, Train_Loss: 0.9129488468170166, Test_Loss: 0.917950451374054
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-18-65c5d0500e5d> in <module>()
     24     for i in range(int(len(x)/batch_size)):
     25         train_batch_x, train_batch_y = loadTrainBatch(batch_size)
---> 26         train_step.run(feed_dict = {x_input: train_batch_x, y_true: train_bat
     27         tr_loss = loss.eval(feed_dict = {x_input: train_batch_x, y_true: trai
     28         train_avg_loss += tr_loss / batch_size

                                7 frames
/usr/local/lib/python3.6/dist-packages/tensorflow_core/python/client/session.py in _c
   1441     return tf_session.TF_SessionRun_wrapper(self._session, options, feed_dict
   1442                                             fetch_list, target_list,
-> 1443                                             run_metadata)
   1444
   1445   def _call_tf_sessionprun(self, handle, feed_dict, fetch_list):

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW