Connecting to drive

```
from google.colab import drive
drive.mount("/content/drive/",force_remount=False)
```

```
Mounted at /content/drive/
```

Importing libraries

```
import numpy as np
import pandas as pd
import cv2
import os
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from keras.models import load_model
from PIL import Image
from random import choice
from google.colab.patches import cv2_imshow
```

Importing the model

```
model_path = 'drive/My Drive/Colab Notebooks/traffic signs python/tr
model = load_model(model_path)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 60)        1560
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 60)        90060
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 60)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 30)        16230
```

```
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 30)          8130      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 30)          0         
_________________________________________________________________
dropout (Dropout)            (None, 4, 4, 30)          0         
_________________________________________________________________
flatten (Flatten)            (None, 480)               0         
_________________________________________________________________
dense (Dense)                (None, 500)               240500    
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 43)                21543     
=================================================================
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
_________________________________________________________________
```

```
def grayscale(img):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    return img
def equalize(img):
    img =cv2.equalizeHist(img)
    return img
def preprocessing(img):
    img = grayscale(img)
    img = equalize(img)
    img = img/255
    return img

data = pd.read_csv('drive/My Drive/Colab Notebooks/traffic signs pyt
images_path = 'drive/My Drive/Colab Notebooks/traffic signs python/t
#print(data.head(5))
###############
#img_test = cv2.imread('drive/My Drive/Colab Notebooks/traffic signs
img_test = cv2.imread(images_path)
img = cv2.resize(img_test,(32,32))
print('img shape0',img.shape)
img = np.array(img)
img = preprocessing(img)
img = img.reshape(1, 32, 32, 1)
print('img shape',img.shape)
#img = img.reshape(1, 32, 32, 1)
#img = cv2.resize(img_test,(32,32))
print(img.shape)

print('predecting..')
pr=  model.predict(img)
lis = list(pr[0])
print(lis)
print([round(i) for i in lis])
###############
"""
classId = list(data['ClassId'])
labels = list(data['Name'])
classId_classNo = dict(list(zip(classId,labels)))
#list_images = os.listdir(images_path)
images = list()
for index in range(len(list_images)):
    images.append(cv2.imread(images_path + '/' + list_images[index])

images = list(map(np.asarray,images))
images = [cv2.resize(image,(32,32)) for image in images]
images = list(map(preprocessing,images))
plt.imshow(images[5])
plt.show()
images = [image.reshape(1,32,32,1) for image in images]



pr=  model.predict(img)
lis = list(pr[0])
print([round(i) for i in lis])
"""
```

```
img shape0 (32, 32, 3)
img shape (1, 32, 32, 1)
(1, 32, 32, 1)
predecting..
[7.5046955e-08, 0.00022620881, 1.7670078e-06, 6.9421094e-07, 2.981645e-08, 2.6126007e-0
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0
'\nclassId = list(data['ClassId'])\nlabels = list(data['Name'])\nclassId_classNo = dict
(list(zip(classId,labels)))\n#list_images = os.listdir(images_path)\nimages = list()\nf
or index in range(len(list_images)):\n    images.append(cv2.imread(images_path + '/' +
list_images[index]))\n\nimages = list(map(np.asarray,images))\nimages = [cv2.resize(ima
ge,(32,32)) for image in images]\nimages = list(map(preprocessing,images))\nplt.imshow
(images[5])\nplt.show()\nimages = [image.reshape(1,32,32,1) for image in images]\n\n\n
```

Animation

```
def animate(i, data1, data2, line1, line2):
    temp1 = data1.iloc[:int(i+1)]
    temp2 = data2.iloc[:int(i+1)]

    line1.set_data(temp1.index, temp1.value)
    line2.set_data(temp2.index, temp2.value)

    return (line1, line2)


def create_loss_animation(model_type, data1, data2):
    fig = plt.figure()
    plt.title(f'Loss on Train & Test', fontsize=25)
    plt.xlabel('Epoch', fontsize=20)
    plt.ylabel('Loss MSE for Sx-Sy & Sxy', fontsize=20)
    plt.xlim(min(data1.index.min(), data2.index.min()), max(data1.in
    plt.ylim(min(data1.value.min(), data2.value.min()), max(data1.va

    l1, = plt.plot([], [], 'o-', label='Train Loss', color='b', mark
    l2, = plt.plot([], [], 'o-', label='Test Loss', color='r', marke
    plt.legend(loc='center right', fontsize='xx-large')

    Writer = animation.writers['ffmpeg']
    writer = Writer(fps=5, bitrate=1800)

    ani = matplotlib.animation.FuncAnimation(fig, animate, fargs=(da
    ani.save(f'{model_type}.mp4', writer=writer)

# create datasets
x = np.linspace(0,150,50)
y1 = 41*np.exp(-x/20)
y2 = 35*np.exp(-x/50)

my_data_number_1 = pd.DataFrame({'x':x, 'value':y1}).set_index('x')
my_data_number_2 = pd.DataFrame({'x':x, 'value':y2}).set_index('x')

create_loss_animation('test', my_data_number_1, my_data_number_2)
```