import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
import tensorflow as tf  
import os  
# from IPython.display import Image  
import easyocr  
  
  
def minist\_traning():  
 *"""  
 Does the traning from the MNIST dataset* ***:return****: None  
 """* mnist = tf.keras.datasets.mnist  
 (x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()  
 print(x\_train.shape)  
 x\_train = tf.keras.utils.normalize(x\_train, axis=1)  
 x\_test = tf.keras.utils.normalize(x\_test, axis=1)  
 model = tf.keras.models.Sequential()  
 model.add(tf.keras.layers.Flatten(input\_shape=(28, 28)))  
 model.add(tf.keras.layers.Dense(units=128, activation=tf.nn.relu))  
 model.add(tf.keras.layers.Dense(units=128, activation=tf.nn.relu))  
 model.add(tf.keras.layers.Dense(units=10, activation=tf.nn.softmax))  
 model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])  
 model.fit(x\_train, y\_train, epochs=3)  
 accuracy, loss = model.evaluate(x\_test, y\_test)  
 print(accuracy)  
 print(loss)  
 model.save('digits.model')  
 return model  
  
  
def predict\_using\_trained():  
 *"""  
 Takes the images that are present in the trained folder and uses the mnist traning to predict the numbers* ***:return****: None  
 """* model = minist\_traning()  
 cur\_wd = os.getcwd()  
 path = cur\_wd + "\\" + "Trained"  
 print(path)  
 imgs = os.listdir(path)  
 for cur\_img in range(1,len(imgs)+1):  
 os.chdir(path)  
 img = cv.imread(f'{cur\_img}.png')[:,:,0]  
 img = np.invert(np.array([img]))  
 prediction = model.predict(img)  
 print(f'The result is probably:{np.argmax(prediction)}')  
 plt.imshow(img[0], cmap=plt.cm.binary)  
 plt.show()  
 os.chdir('..')  
  
  
def easy\_ocr():  
 *"""  
 This module uses the easy ocr library to predict the output* ***:return****: None  
 """* reader = easyocr.Reader(['en','hi'])  
 cur\_wd = os.getcwd()  
 path = cur\_wd + "\\" + "EasyOCR"  
 imgs = os.listdir(path)  
 for cur\_img in range(1,len(imgs)+1):  
 os.chdir(path)  
 pre\_img = cur\_img  
 img = cv.imread(f'{cur\_img}.png')[:, :, 0]  
 img = np.invert(np.array([img]))  
 plt.imshow(img[0], cmap=plt.cm.binary)  
 plt.show()  
 output = reader.readtext(f'{cur\_img}.png')  
 tup1 = output[0]  
 print(f" Predicted Words / characters are : {tup1[1]} ", end=" ")  
 print(f" Accuracy: {tup1[2]\*100} ",end= "\n")  
 print(output)  
 os.chdir('..')  
  
  
if \_\_name\_\_=="\_\_main\_\_":  
 print("\n Executing the pre trained function \n")  
 predict\_using\_trained()  
 print("\n Executing the easy ocr function \n")  
 easy\_ocr()