**NAME: AAKASH MOHAN**

**ID: 1001656408**

**EE5353 NEURAL NETWORKS and DEEP LEARNING**

**ASSIGNMENT 7 :**

**CHARACTER RECOGNITION USING CONVOLUTIONAL NEURAL NETS using KERAS using GOOGLE COLAB**

|  |  |
| --- | --- |
|  | |
|  | # -\*- coding: utf-8 -\*- """Character\_Recognition\_for\_Students.ipynb  Automatically generated by Colaboratory. |
| Original file is located at  https://colab.research.google.com/drive/1KDJ-tXKqHR5YafdcC5ajfnch3MJsZS-u  """  # -\*- coding: utf-8 -\*- """  Created on Thur Nov 7 14:20:37 2019  Reference from https://github.com/anujshah1003/own\_data\_cnn\_implementation\_keras/blob/mast @author: RaneChinmayAppa  """  import numpy as np import os,cv2  import glob  from sklearn.utils import shuffle  from sklearn.model\_selection import train\_test\_split import re  from keras.utils import np\_utils  import matplotlib.pyplot as plt  from keras.utils import to\_categorical from keras.models import Sequential  from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout  def sorted\_aphanumeric(data):  convert = lambda text: int(text) if text.isdigit() else text.lower() alphanum\_key = lambda key: [ convert(c) for c in re.split('([0-9]+)', key) ] return sorted(data, key=alphanum\_key)  def gen\_image(arr):  two\_d = (np.reshape(arr, (28, 28)) \* 255).astype(np.uint8) plt.imshow(two\_d, interpolation='nearest')  return plt def unique(list1):  # insert the list to the set list\_set = set(list1)  # convert the set to the list unique\_list = (list(list\_set)) for x in unique\_list:  print(x) | |

#from sklearn.cross\_validation import train\_test\_split

from google.colab import drive drive.mount('/content/drive')

PATH = os.getcwd() # Define data path

data\_path = '/content/drive/My Drive/Colab Notebooks/Character Images' # inset your path data\_dir\_list = sorted\_aphanumeric(os.listdir(data\_path)) # os.listdir(data\_path)

img\_rows=128 img\_cols=128 num\_channel=1 num\_epoch=20

# Define the number of classes num\_classes = 34

labels\_name={'0':0,'1':1,'2':2,'3':3,'4':4,'5':5,'6':6,'7':7,'8':8,'9':9,'A':10,'B':11,'C'

img\_data\_list=[] labels\_list = []

for dataset in data\_dir\_list:

img\_list = glob.glob(data\_path+'/'+ dataset +'/\*.png')

label = labels\_name[dataset] # label is generated as the library updated above for img in img\_list:

input\_img=cv2.imread(img,1 ) input\_img=cv2.cvtColor(input\_img, cv2.COLOR\_BGR2GRAY) input\_img\_resize=cv2.resize(input\_img,(28,28)) img\_data\_list.append(input\_img\_resize) labels\_list.append(label)

#print(unique(labels\_list)) img\_data = np.array(img\_data\_list)

img\_data = img\_data.astype('float32') labels = np.array(labels\_list)

#print(unique(labels)) print(np.unique(labels,return\_counts=True))

Y = np\_utils.to\_categorical(labels, num\_classes)

#Shuffle the dataset

x,y = shuffle(img\_data,Y, random\_state=2)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=2) # #Normalization of the data

X train = X train / 255

X\_test = X\_test / 255

Nv = X\_train.shape[0] Nv\_test = X\_test.shape[0]

#reshape data to fit model

X\_train = X\_train.reshape(int(Nv),28,28,1) X\_test = X\_test.reshape(int(Nv\_test),28,28,1)

model = Sequential()

################## add model layers described in the assignment ######################

model.add(Conv2D(32, kernel\_size=5, activation='relu', input\_shape=(28,28,1)))

# 64 are the number of filters, kernel size is the size of the filters example 3\*3 #model.add(Conv2D(32, kernel\_size=5, activation='relu')) model.add(MaxPooling2D(pool\_size=(3,3)))

model.add(Dropout(0.35)) model.add(Flatten()) model.add(Dense(64, activation='relu')) model.add(Dropout(0.35))

model.add(Dense(34, activation='softmax'))

########################################################################################

# 8. Compile model model.compile(loss='categorical\_crossentropy',

optimizer='adam', metrics=['accuracy'])

# 9. Fit model on training data model.fit(X\_train, y\_train,

batch\_size=32, nb\_epoch=10, verbose=1) #epochs = iterations(Nit)

# 10. Evaluate model on test data

score = model.evaluate(X\_test, y\_test, verbose=1) print('Testing accuracy - > ',score[1] \* 100)

ytested = model.predict\_classes(X\_test) for i in range(10):

print("The Predicted Testing image is =%s verify below" % ((list(labels\_name.keys())[lis gen\_image(X\_test[i]).show() # printing image vs the predicted image below



Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m (array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,

17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]), array([5

50, 18, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50]))

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl Instructions for updating:

Please use `rate` instead of `keep\_prob`. Rate should be set to `rate = 1 - keep\_prob WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:79

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow\_core/python Instructions for updating:

Use tf.where in 2.0, which has the same broadcast rule as np.where

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:143: UserWarning: The `n WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl Epoch 1/10

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl 1334/1334 [==============================] - 9s 7ms/step - loss: 3.1074 - acc: 0.1769

Epoch 2/10

1334/1334 [==============================] - 0s 179us/step - loss: 1.5432 - acc: 0.58

Epoch 3/10

1334/1334 [==============================] - 0s 184us/step - loss: 0.6982 - acc: 0.80

Epoch 4/10

1334/1334 [==============================] - 0s 183us/step - loss: 0.4639 - acc: 0.87

Epoch 5/10

1334/1334 [==============================] - 0s 179us/step - loss: 0.3151 - acc: 0.90

Epoch 6/10

1334/1334 [==============================] - 0s 171us/step - loss: 0.2524 - acc: 0.94

Epoch 7/10

1334/1334 [==============================] - 0s 187us/step - loss: 0.2108 - acc: 0.94

Epoch 8/10

1334/1334 [==============================] - 0s 166us/step - loss: 0.1607 - acc: 0.95

Epoch 9/10

1334/1334 [==============================] - 0s 178us/step - loss: 0.1553 - acc: 0.95

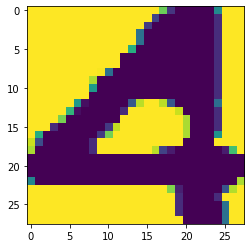
Epoch 10/10

1334/1334 [==============================] - 0s 165us/step - loss: 0.1322 - acc: 0.95

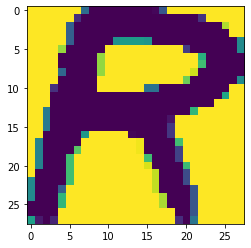
334/334 [==============================] - 0s 237us/step

Testing accuracy - > 98.20359281437125

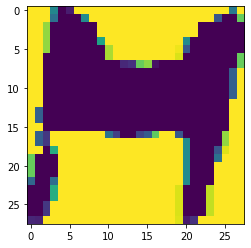
The Predicted Testing image is =4 verify below



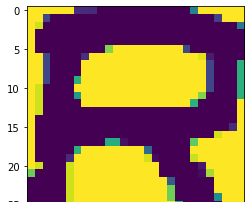
The Predicted Testing image is =R verify below



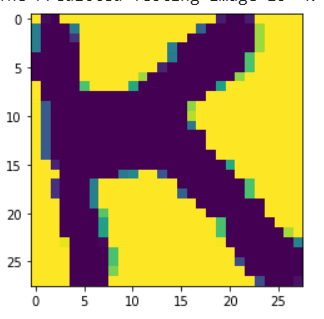
The Predicted Testing image is =M verify below



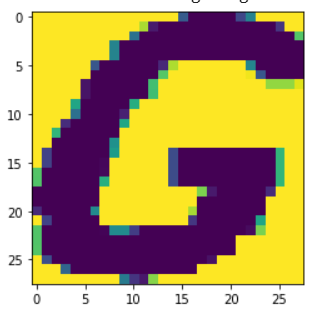
The Predicted Testing image is =R verify below



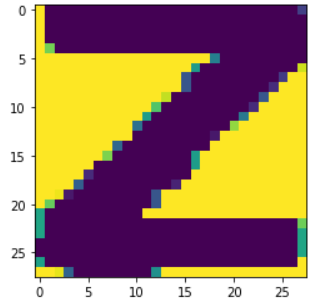
The Predicted Testing image is =K verify below



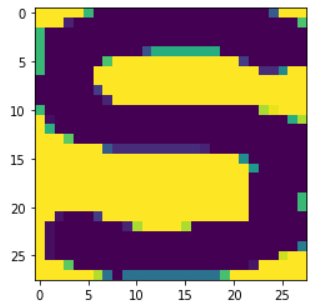
The Predicted Testing image is =G verify below



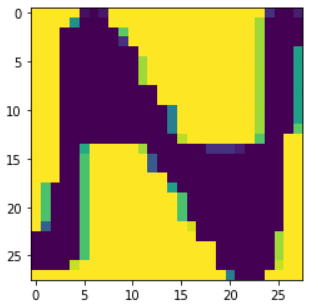
The Predicted Testing image is =Z verify below



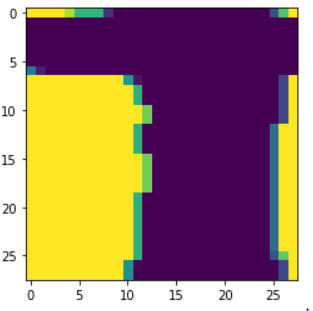
The Predicted Testing image is =5 verify below



The Predicted Testing image is =N verify below



The Predicted Testing image is =1 verify below



Explanation of code:

* First, we are importing the required files
* Then we use the “Lambda func” to sort the image and generate image. The lambda function is used for higher order functions.
* In generation of image we use tensor flow reshape function.
* Then we create path for importing the character file.
* Then we are creating python data structures for inputs like we are creating list and tuples.
* Then we print shuffle reshape and normalize the data.
* At the end we are calling different model layers to recognition the image.
* We are calling the convolution library file and passing 4 arguments and these arguments are “filter size, kernel size, activation function, input shape”.
* Generally, convolution layer has more than 10 arguments since we require only 4 arguments for our project, we don’t use the rest.
* Then we call the pooling library file with 1 argument and that argument is “pool size”.
* Similarly, the pooling function has 4 arguments and we use only one of those arguments.
* Then we add the dropout layer with one argument and that one argument corresponds to “rate”. The other two arguments are noise shape and seed
* Next layer that we add would be flatten with no arguments, however, flatten accepts one arguments so if we don’t specify it will by default take an argument.
* Then we add the dense layer with 2 arguments and these 2 are the “ hidden units and activation function”.
* The relu activation function is similar to that of the sigmoid function but faster.
* Similarly like the convolution and pooling layer dense also has more arguments but we only use two of those.
* We again call the dropout layer
* And finally, we call the dense layer with SoftMax activation function.
* The SoftMax function has input with tensor axis.
* Then we train and test the data for 10 epochs to recognize the image.

**CONCLUSION:**

In this assignment we run the code that is capable of recognizing the image with different model layers. These model layers are imported as a library file and are executed. We give necessary arguments to generate the and reshape the image. The image file is mentioned as a data path and then 10 epochs are executed to predict the image. The training and testing of the image are done with an Accuracy of 98.2%