Task

On

CNN

**Course**: Artificial Intelligence

(Machine Learning & Deep Learning)

Name: Abbas Shafi

Father Name: Dr Muhammad Shafi

Week: 22

Submitted to: Sir Syed Nazir Afridi



National Vocational & Technical Training Commission

National Center For Big Data & Cloud Computing

University of Engineering & Technology

Peshawar

[10]:

**from keras.datasets import** mnist

#Loads the mnist dataset

(x\_train,y\_train),(x\_test,y\_test)=mnist.load\_data()

[11]:

[12]:

(60000, 28, 28)

*#printing the number of Samples, Labels in datasets*

print('Initial shape or dimension of X\_train',str(x\_train.shape))

print('number of samples in our training data',str(len(x\_train))) print('number of samples in our test data',str(len(x\_test)))

print('number of labels in our training data',str(len(y\_train))) print('number of labels in our test data',str(len(y\_test)))

[13]:

Initial shape or dimension of X\_train (60000, 28, 28) number of samples in our training data 60000

number of samples in our test data 10000 number of labels in our training data 60000 number of labels in our test data 10000

*#Lets take some of images in our dataset*

**import cv2**

**import numpy as np**

**from google.colab.patches import** cv2\_imshow

**for** i **in** range(0,6): random\_num=np.random.randint(0,len(x\_train)) img=x\_train[random\_num] window\_name="Random\_Sample #"+str(i) cv2\_imshow(img)

cv2.waitKey(0)

cv2.destroyAllWindows()

[14]:

[14]:

[15]:

*#Prepare our dataset for training* img\_rows=x\_train[0].shape[0] img\_cols=x\_train[1].shape[0]

*#getting data in the right shape needed for keras #we need to add 4th dimension to our data*

*#changing our orignal image to [60000,28,28] to [60000,28,28,1]*

x\_train=x\_train.reshape(x\_train.shape[0],img\_rows,img\_cols,1)

*#Lets plot it using matplotlib*

# import matplotlib.pyplot as plt

plt.subplot(331) random\_num=np.random.randint(0,len(x\_train)) plt.imshow(x\_train[random\_num],cmap=plt.get\_cmap('gray'))

plt.subplot(332) random\_num=np.random.randint(0,len(x\_train)) plt.imshow(x\_train[random\_num],cmap=plt.get\_cmap('gray'))

plt.subplot(333) random\_num=np.random.randint(0,len(x\_train)) plt.imshow(x\_train[random\_num],cmap=plt.get\_cmap('gray'))

plt.subplot(334) random\_num=np.random.randint(0,len(x\_train)) plt.imshow(x\_train[random\_num],cmap=plt.get\_cmap('gray'))

plt.subplot(335) random\_num=np.random.randint(0,len(x\_train)) plt.imshow(x\_train[random\_num],cmap=plt.get\_cmap('gray'))

plt.subplot(336) random\_num=np.random.randint(0,len(x\_train)) plt.imshow(x\_train[random\_num],cmap=plt.get\_cmap('gray'))

<matplotlib.image.AxesImage at 0x20d91eb0970>

x\_test=x\_test.reshape(x\_test.shape[0],img\_rows,img\_cols,1)

*#store the shape of a single image*

input\_image=(img\_rows,img\_cols,1)

*#because keras expected the float32 so we change it float 32* x\_train=x\_train.astype('float32') x\_test=x\_test.astype('float32')

*#normalize the range by changing the data from (0-255) to (0-1)*

x\_train /=255 x\_test /=255

print("X\_train Shape:", x\_train.shape) print(x\_train.shape[0],"train Samples") print(x\_test.shape[0],"test samples")

[16]:

X\_train Shape: (60000, 28, 28, 1)

60000 train Samples

10000 test samples

*#onehot encoding*

**from keras.utils import** np\_utils

y\_train=np\_utils.to\_categorical(y\_train) y\_test=np\_utils.to\_categorical(y\_test)

print("Number of classes:",str(y\_test.shape[1])) num\_classes=y\_test.shape[1] num\_of\_pixels=x\_train.shape[1] \* x\_train.shape[2]

print(num\_of\_pixels)

Number of classes: 10 784

[17]:

[17]: array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.], dtype=float32)

[18]:

**from keras.layers import** Conv2D, MaxPooling2D

**from keras import** backend **as** K

**from keras.optimizers import** gradient\_descent\_v2

# from tensorflow.keras.optimizers import SGD,Adam

*#create Model*

model = Sequential() model.

*‹→*add(Conv2D(32,kernel\_size=(3,3),activation='relu',input\_shape=input\_image))

model.add(Conv2D(64, (3, 3), activation="relu")) model.add(MaxPooling2D(pool\_size=(2,2))) model.add(Dropout(0.25))

model.add(Flatten()) model.add(Dense(128,activation="relu")) model.add(Dropout(0.5)) model.add(Dense(num\_classes,activation="softmax"))

*#set optimizers*

opt = tf.optimizers.SGD(learning\_rate=0.01)

model.

*‹→*compile(loss='categorical\_crossentropy',optimizer=opt,metrics=['accuracy'])

*#opt = tf.keras.optimizers.Adam(learning\_rate=0.01) #model.compile(loss='categorical\_crossentropy', optimizer=opt)* print(model.summary())

Model: "sequential"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| conv2d (Conv2D) | (None, 26, 26, 32) | | | 320 |
| conv2d\_1 (Conv2D) | (None, 24, 24, 64) | | | 18496 |
| max\_pooling2d (MaxPooling2D  ) | (None, 12, 12, 64) | | | 0 |
| dropout (Dropout) | (None, | 12, 12, | 64) | 0 |
| flatten (Flatten) | (None, | 9216) |  | 0 |
| dense (Dense) | (None, | 128) |  | 1179776 |
| dropout\_1 (Dropout) | (None, | 128) |  | 0 |

[ ]:

dense\_1 (Dense) (None, 10) 1290

=================================================================

Total params: 1,199,882

Trainable params: 1,199,882

Non-trainable params: 0 None