#SPRINT 1 TEAM ID:PNT2022TMID24124

#Importing required Libraries

import numpy*#for numerical analysis*

import tensorflow*#open source ml tool by google*

from tensorflow.keras.datasets import mnist *#mnist dataset*

from tensorflow.keras.models import Sequential*# stack for layers* from tensorflow.keras import layers*#input,middle and output layers forcnn structure*

from tensorflow.keras.layers import Dense,Flatten*#dense and flatten layers*

from tensorflow.keras.layers import Conv2D*#convolutional layers* from tensorflow import keras*#library for building neural networks built on tensorflow*

from tensorflow.keras.optimizers import Adam*#optimizers* from keras.utils import np\_utils

#Loading dataset

Dataset is available in tensorflow dataset repository

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

Downloading data from [https://storage.googleapis.com/tensorflow/tf- keras-datasets/mnist.npz](https://storage.googleapis.com/tensorflow/tf-%20keras-datasets/mnist.npz)

11490434/11490434 [==============================] - 0s 0us/step

automatically data is splitted for train ,test -70:30 ratio

print(x\_train.shape) print(y\_train.shape)

(60000, 28, 28)

(60000,)

Training Dataset has 60000 images S testing has 10000 images

print(x\_test.shape) print(y\_test.shape)

(10000, 28, 28)

(10000,)

**Analyze the data**

x\_train[3]

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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, | 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, | 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, | 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, | 0, | 0, | 0, | 124, | 253, | 255, | 63, | 0, | 0, | 0, |
| 0, | 0], |  |  |  |  |  |  |  |  |  |  |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 96, | 244, | 251, | 253, | 62, | 0, | 0, | 0, |
| 0, | 0], |  |  |  |  |  |  |  |  |  |  |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 127, | 251, | 251, | 253, | 62, | 0, | 0, | 0, |
| 0, | 0], |  |  |  |  |  |  |  |  |  |  |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 68, | 236, | 251, | 211, | 31, | 8, | 0, | 0, | 0, |
| 0, | 0], |  |  |  |  |  |  |  |  |  |  |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 60, | 228, | 251, | 251, | 94, | 0, | 0, | 0, | 0, | 0, |

array([[ 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, |
| 253, | 253, | 189, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 251, | 235, | 66, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 251, | 126, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 184, | 15, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 23, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 32, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 151, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 48, | 221, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 234, | 251, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 253, | 251, |
| 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |

0,

0,

0,

0,

0,

0,

0,

0,

80,

0,

253,

0,

251,

0,

251,

0,

251,

0,

251,

[ 0, 0, 0,

0, 0, 155,

0, 0],

[ 0, 0, 0,

0, 20, 253,

0, 0],

[ 0, 0, 0,

32, 205, 253,

0, 0],

[ 0, 0, 0,

104, 251, 253,

0, 0],

[ 0, 0, 0,

240, 251, 193,

0, 0],

[ 0, 0, 0,

253, 253, 159,

0, 0],

[ 0, 0, 0,

251, 251, 39,

0, 0],

[ 0, 0, 0,

251, 172, 0,

0, 0],

[ 0, 0, 0,

196, 12, 0,

0, 0],

[ 0, 0, 0,

89, 0, 0,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | [ | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 159, | 255, | 253, |
| 253, |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 31, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0, | 0], |  |  |  |  |  |  |  |  |  |  |
|  | [ | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 48, | 228, | 253, | 247, |
| 140, |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 8, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0, | 0], |  |  |  |  |  |  |  |  |  |  |
|  | [ | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 64, | 251, | 253, | 220, |
| 0, |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0, | 0], |  |  |  |  |  |  |  |  |  |  |
|  | [ | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 64, | 251, | 253, | 220, |
| 0, |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, |
| 0, |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0, | 0], |  |  |  |  |  |  |  |  |  |  |
|  | [ | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 24, | 193, | 253, | 220, |
| 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, | 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], |  |  |  |  |  |  |  |  |  |  |
|  | [ | 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]], dtype=uint8) y\_train[36]

6

image in 36th position in training dataset import matplotlib.pyplot as plt plt.imshow(x\_train[36])

<matplotlib.image.AxesImage at 0x7fa70fb3b550>



#Reshaping the data

,As we are using Deep learning neural network, the input for this network to get trained on should be of higher dimensional. Our dataset is having three-dimensional images so we have to reshape them too higher dimensions

*#(batch,height,width,channel)* x\_train=x\_train.reshape(60000,28,28,1).astype('float32') x\_test=x\_test.reshape(10000,28,28,1).astype('float32')

#Applying one hot encoding

One hot encoding to convert numerical values to classes where 0 to 9 are 10 seperate classes if value is 5 class 5 is 1 else 0

no\_of\_classes=10 y\_train=np\_utils.to\_categorical(y\_train,no\_of\_classes) y\_test=np\_utils.to\_categorical(y\_test,no\_of\_classes)

y\_test[3]

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