**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

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

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

**from** tensorflow.keras.models **import** Sequential

**from** tensorflow.keras.layers **import** Conv2D, Dense, Flatten

**from** tensorflow.keras.optimizers **import** Adam

**from** tensorflow.keras.models **import** load\_model

**from** PIL **import** Image, ImageOps

**import** numpy

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

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

11490434/11490434 [==============================] - 1s 0us/step

I

print(X\_train**.**shape)

print(X\_test**.**shape)

(60000, 28, 28)

(10000, 28, 28)

X\_train[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],

[ 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, 3,

18, 18, 18, 126, 136, 175, 26, 166, 255, 247, 127, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 30, 36, 94, 154, 170,

253, 253, 253, 253, 253, 225, 172, 253, 242, 195, 64, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 49, 238, 253, 253, 253, 253,

253, 253, 253, 253, 251, 93, 82, 82, 56, 39, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 18, 219, 253, 253, 253, 253,

253, 198, 182, 247, 241, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 80, 156, 107, 253, 253,

205, 11, 0, 43, 154, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 14, 1, 154, 253,

90, 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, 139, 253,

190, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 11, 190,

253, 70, 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, 35,

241, 225, 160, 108, 1, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

81, 240, 253, 253, 119, 25, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 45, 186, 253, 253, 150, 27, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 16, 93, 252, 253, 187, 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, 249, 253, 249, 64, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 46, 130, 183, 253, 253, 207, 2, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 39,

148, 229, 253, 253, 253, 250, 182, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 24, 114, 221,

253, 253, 253, 253, 201, 78, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 23, 66, 213, 253, 253,

253, 253, 198, 81, 2, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 18, 171, 219, 253, 253, 253, 253,

195, 80, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 55, 172, 226, 253, 253, 253, 253, 244, 133,

11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 136, 253, 253, 253, 212, 135, 132, 16, 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[0]

5

plt**.**imshow(X\_train[1])

X\_train **=** X\_train**.**reshape(60000, 28, 28, 1)**.**astype('float32')

X\_test **=** X\_test**.**reshape(10000, 28, 28, 1)**.**astype('float32')

number\_of\_classes **=** 10

Y\_train **=** np\_utils**.**to\_categorical(y\_train, number\_of\_classes)

Y\_test **=** np\_utils**.**to\_categorical(y\_test, number\_of\_classes)

Y\_train[0]

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

model **=** Sequential()

model**.**add(Conv2D(64, (3, 3), input\_shape**=**(28, 28, 1), activation**=**"relu"))

model**.**add(Conv2D(32, (3, 3), activation**=**"relu"))

model**.**add(Flatten())

model**.**add(Dense(number\_of\_classes, activation**=**"softmax"))

model**.**compile(loss**=**'categorical\_crossentropy', optimizer**=**"Adam", metrics**=**["accuracy"])

model**.**fit(X\_train, Y\_train, batch\_size**=**32, epochs**=**5, validation\_data**=**(X\_test,Y\_test))

Epoch 1/5

1875/1875 [==============================] - 194s 103ms/step - loss: 0.2567 - accuracy: 0.9506 - val\_loss: 0.0980 - val\_accuracy: 0.9693

Epoch 2/5

1875/1875 [==============================] - 196s 105ms/step - loss: 0.0695 - accuracy: 0.9791 - val\_loss: 0.0983 - val\_accuracy: 0.9735

Epoch 3/5

1875/1875 [==============================] - 196s 105ms/step - loss: 0.0494 - accuracy: 0.9842 - val\_loss: 0.0906 - val\_accuracy: 0.9755

Epoch 4/5

1875/1875 [==============================] - 192s 102ms/step - loss: 0.0375 - accuracy: 0.9882 - val\_loss: 0.0913 - val\_accuracy: 0.9787

Epoch 5/5

1875/1875 [==============================] - 196s 104ms/step - loss: 0.0306 - accuracy: 0.9903 - val\_loss: 0.1032 - val\_accuracy: 0.9743

metrics **=** model**.**evaluate(X\_test, Y\_test, verbose**=**0)

print("Metrics (Test Loss & Test Accuracy): ")

print(metrics)

Metrics (Test Loss & Test Accuracy):

[0.10322817414999008, 0.9743000268936157]

prediction **=** model**.**predict(X\_test[:4])

print(prediction)

1/1 [==============================] - 0s 64ms/step

[[4.77358049e-11 1.26020884e-14 2.23637656e-07 2.59297366e-07

1.53105145e-18 1.41474479e-13 2.73819453e-19 9.99999523e-01

5.75746352e-12 1.40723442e-08]

[3.92702641e-05 3.63764530e-09 9.99928832e-01 1.10518204e-06

3.28396650e-11 1.87219923e-13 3.02575540e-06 4.75269130e-12

2.79003762e-05 1.17118581e-09]

[3.37602168e-11 9.99982953e-01 7.10459869e-09 3.63090309e-13

1.67968246e-05 6.36366426e-09 4.59948364e-11 2.65287614e-09

2.72516672e-07 1.53049936e-12]

[9.99999762e-01 1.02759820e-17 6.89465485e-10 4.13503087e-14

3.53135576e-12 2.56500203e-11 6.89072754e-09 4.50628203e-14

8.74276596e-10 1.82247064e-07]]

print(numpy**.**argmax(prediction, axis**=**1))

print(Y\_test[:4])

[7 2 1 0]

[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]

[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]

[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]

[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

model**.**save("model.h5")

model**=**load\_model("model.h5")