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

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[0])

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 [==============================] - 192s 102ms/step - loss: 0.2245 - accuracy: 0.9518 - val\_loss: 0.1058 - val\_accuracy: 0.9701

Epoch 2/5

1875/1875 [==============================] - 197s 105ms/step - loss: 0.0685 - accuracy: 0.9788 - val\_loss: 0.0962 - val\_accuracy: 0.9752

Epoch 3/5

1875/1875 [==============================] - 190s 101ms/step - loss: 0.0468 - accuracy: 0.9854 - val\_loss: 0.0900 - val\_accuracy: 0.9749

Epoch 4/5

1875/1875 [==============================] - 190s 102ms/step - loss: 0.0351 - accuracy: 0.9891 - val\_loss: 0.0993 - val\_accuracy: 0.9748

Epoch 5/5

1875/1875 [==============================] - 191s 102ms/step - loss: 0.0270 - accuracy: 0.9917 - val\_loss: 0.1005 - val\_accuracy: 0.9764

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

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

print(metrics)

Metrics (Test Loss & Test Accuracy):

[0.10052110999822617, 0.9764000177383423]

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

print(prediction)

1/1 [==============================] - 0s 92ms/step

[[1.5678695e-09 1.6640128e-14 2.0494097e-12 1.5698962e-08 5.4015579e-15

3.6338055e-13 2.2240399e-20 1.0000000e+00 2.9577885e-08 1.9005494e-08]

[5.8188578e-09 1.2512093e-10 9.9999821e-01 7.4831279e-09 1.0770124e-10

2.9252167e-18 1.6483800e-06 1.5410843e-14 1.2811967e-07 3.3103555e-12]

[1.2689595e-09 9.9028254e-01 3.9091717e-08 1.3732340e-10 9.6216686e-03

2.9094124e-07 1.9340013e-10 4.5208512e-07 9.5003670e-05 2.4108826e-10]

[1.0000000e+00 7.3556976e-16 3.5439882e-12 4.7910155e-14 3.2022885e-12

1.5000925e-12 1.5939531e-11 4.1566353e-14 7.7353792e-12 1.2456662e-09]]

I

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")

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

**from** matplotlib **import** pyplot

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

print('X\_train:' **+**str(X\_train**.**shape))

print('y\_train:' **+**str(y\_train**.**shape))

print('X\_test:' **+**str(X\_test**.**shape))

print('y\_test:' **+**str(y\_test**.**shape))

**from** matplotlib **import** pyplot

**for** i **in** range(9):

pyplot**.**subplot(330**+**1**+**i)

pyplot**.**imshow(X\_train[i],cmap**=**pyplot**.**get\_cmap('gray'))

pyplot**.**show()

X\_train:(60000, 28, 28)

y\_train:(60000,)

X\_test:(10000, 28, 28)

y\_test:(10000,)