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,

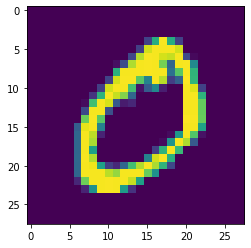
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0]], dtype=uint8)

y\_train[0]

5

plt.imshow(X\_train[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 [==============================] - 175s 93ms/step - loss: 0.2493 - accuracy: 0.9536 - val\_loss: 0.0964 - val\_accuracy: 0.9685

Epoch 2/5

1875/1875 [==============================] - 175s 93ms/step - loss: 0.0665 - accuracy: 0.9797 - val\_loss: 0.0905 - val\_accuracy: 0.9714

Epoch 3/5

1875/1875 [==============================] - 173s 92ms/step - loss: 0.0470 - accuracy: 0.9848 - val\_loss: 0.1015 - val\_accuracy: 0.9737

Epoch 4/5

1875/1875 [==============================] - 174s 93ms/step - loss: 0.0386 - accuracy: 0.9876 - val\_loss: 0.0875 - val\_accuracy: 0.9780

Epoch 5/5

1875/1875 [==============================] - 172s 92ms/step - loss: 0.0263 - accuracy: 0.9912 - val\_loss: 0.1246 - val\_accuracy: 0.9748

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

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

print(metrics)

Metrics (Test Loss & Test Accuracy):

[0.1246321052312851, 0.9747999906539917]

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

print(prediction)

1/1 [==============================] - 0s 95ms/step

[[9.69764113e-12 5.35128160e-18 6.73075276e-11 8.16666557e-09

6.05515796e-16 1.61785570e-12 8.17041761e-17 1.00000000e+00

8.81304404e-12 4.62230726e-10]

[4.07073875e-10 7.69928477e-09 1.00000000e+00 9.22982697e-12

3.13884536e-17 1.78578504e-20 1.23509398e-08 2.08939884e-17

3.13647155e-12 7.12971540e-23]

[4.91924723e-09 9.99989867e-01 7.83750487e-09 2.34650103e-12

3.13111173e-06 4.78714313e-09 1.26857791e-09 9.65541119e-11

7.08947118e-06 4.01666284e-10]

[1.00000000e+00 7.29500773e-15 2.82593549e-11 1.63891306e-16

1.03110545e-16 8.73841117e-15 1.64485969e-09 6.97776676e-17

1.71514617e-13 2.39570310e-14]]

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,)