Import necessary package

import numpy

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

Load data

(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

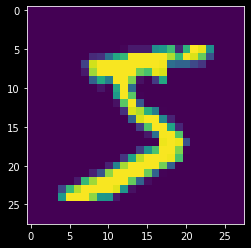
print(X\_train.shape)

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])

Data pre processing

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)

Create model

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 [==============================] - 209s 111ms/step - loss: 0.2373 - accuracy: 0.9528 - val\_loss: 0.0800 - val\_accuracy: 0.9747

Epoch 2/5

1875/1875 [==============================] - 198s 105ms/step - loss: 0.0706 - accuracy: 0.9784 - val\_loss: 0.0846 - val\_accuracy: 0.9752

Epoch 3/5

1875/1875 [==============================] - 195s 104ms/step - loss: 0.0484 - accuracy: 0.9851 - val\_loss: 0.0874 - val\_accuracy: 0.9771

Epoch 4/5

1875/1875 [==============================] - 196s 105ms/step - loss: 0.0359 - accuracy: 0.9888 - val\_loss: 0.0919 - val\_accuracy: 0.9779

Epoch 5/5

1875/1875 [==============================] - 195s 104ms/step - loss: 0.0292 - accuracy: 0.9912 - val\_loss: 0.1064 - val\_accuracy: 0.9775

Test the model

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

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

print(metrics)

Metrics (Test Loss & Test Accuracy):

[0.10643840581178665, 0.9775000214576721]

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

print(prediction)

1/1 [==============================] - 0s 89ms/step

[[2.32846579e-14 9.17158403e-23 6.90037508e-15 7.31810099e-11

4.10212473e-18 4.40370640e-19 9.66349185e-25 1.00000000e+00

2.73089168e-13 1.83711661e-13]

[9.68418488e-12 1.70741088e-09 1.00000000e+00 2.09152730e-14

3.54427834e-16 2.11336798e-16 1.38079270e-09 1.66195792e-21

1.10303775e-08 2.15770184e-21]

[1.18453893e-11 1.00000000e+00 2.75740764e-09 1.34126463e-16

2.06069490e-08 4.37439551e-09 5.86808380e-10 9.92376181e-10

3.11978390e-08 4.28370887e-14]

[9.99996424e-01 1.09464391e-12 2.31800234e-12 7.66664699e-14

1.10396303e-10 1.33016170e-13 3.56359237e-06 3.39002220e-12

3.03460881e-12 7.90741861e-10]]

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.]]