Model\_Building TEAM\_ID - PNT2022TMID49379

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

(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],

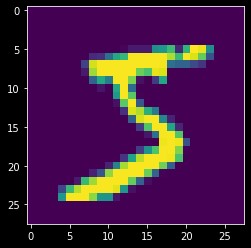
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

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

0, 0]], dtype=uint8)

Data pre processing

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)

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"))

Compiling the model

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 [==============================] - 196s 104ms/step - loss: 0.2114 - accuracy: 0.9530 - val\_loss: 0.0860 - val\_accuracy: 0.9739

Epoch 2/5

1875/1875 [==============================] - 202s 108ms/step - loss: 0.0678 - accuracy: 0.9796 - val\_loss: 0.1017 - val\_accuracy: 0.9751

Epoch 3/5

1875/1875 [==============================] - 197s 105ms/step - loss: 0.0480 - accuracy: 0.9847 - val\_loss: 0.0876 - val\_accuracy: 0.9790

Epoch 4/5

1875/1875 [==============================] - 202s 108ms/step - loss: 0.0368 - accuracy: 0.9890 - val\_loss: 0.0725 - val\_accuracy: 0.9812

Epoch 5/5

1875/1875 [==============================] - 196s 104ms/step - loss: 0.0317 - accuracy: 0.9903 - val\_loss: 0.1061 - val\_accuracy: 0.9749

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):

[0.10613072663545609, 0.9749000072479248] prediction = model.predict(X\_test[:4]) print(prediction)

1/1 [==============================] - 0s 93ms/step

[[9.9039808e-14 1.7801291e-17 2.4331301e-09 5.1562615e-10 2.6416533e-15

9.6221535e-16 1.8084108e-24 1.0000000e+00 3.2786626e-14 6.7406480e-10]

[5.2280425e-16 1.4731727e-12 1.0000000e+00 1.9810487e-11 3.6309416e-18

1.7567800e-18 4.1256623e-08 3.1735288e-19 3.5455400e-10 4.5615819e-23]

[3.9027423e-08 9.9997389e-01 4.7386902e-06 4.2051904e-12 1.8228963e-07

3.9746135e-07 1.9909536e-09 1.0973835e-11 2.0772874e-05 2.1399367e-13]

[9.9999607e-01 5.6394082e-13 2.5004380e-07 1.9577358e-12 9.8116532e-12

6.2192071e-11 2.3564528e-06 9.4811000e-14 1.3014859e-06 6.0907044e-09]] 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.]

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

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