#CNN on MNIST

#1a. Load and process MNIST

from tensorflow import keras

MNISTDB = keras.datasets.mnist

(xtrain,ytrain),(xtest,ytest) = MNISTDB.load\_data()

print("Size of Xtrain:",xtrain.shape)

print("Size of Ytrain:",ytrain.shape)

print("Size of Xtest:",xtest.shape)

print("Size of Ytest:",ytest.shape)

#1b. Display any one image from the dataset

import matplotlib.pyplot as plt

plt.imshow(xtrain[0],cmap='binary')

#1c. Convert to 1 channel

xtrain = xtrain.reshape((60000,28,28,1))

xtest = xtest.reshape((10000,28,28,1))

#1d. Print new size

print("Size of Xtrain:",xtrain.shape)

print("Size of Xtest:",xtest.shape)

#2. Create CNN layers

convnet=keras.models.Sequential()

convnet.add(keras.layers.Conv2D(32,(3,3),activation="relu",input\_shape=xtrain.shape[1:]))

convnet.add(keras.layers.Conv2D(64,(3,3),activation="relu"))

convnet.add(keras.layers.MaxPool2D(2,2))

convnet.add(keras.layers.Dropout(0.25))

convnet.add(keras.layers.Flatten())

convnet.add(keras.layers.Dense(128,activation="relu"))

convnet.add(keras.layers.Dropout(0.25))

convnet.add(keras.layers.Dense(10,activation="softmax"))

convnet.summary()

#3. Compile and test

convnet.compile(loss="sparse\_categorical\_crossentropy",optimizer="adam",metrics="accuracy")

convnet.fit(xtrain,ytrain,epochs=2,batch\_size=17)

testloss,testaccuracy=convnet.evaluate(xtest,ytest)

print("Test Loss =",testloss)

print("Test Accuracy =",testaccuracy)