#1. Load and Process MNIST

#a. import keras

from tensorflow import keras

MNDB=keras.datasets.mnist

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

print("Xtrain size:",xtrain.shape)

print("Ytrain size:",ytrain.shape)

print("Xtest size:",xtest.shape)

print("Ytest size:",ytest.shape)

import matplotlib.pyplot as plt

plt.imshow(xtrain[59999],cmap='binary')

#b. conversion to one channel

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

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

print("Xtrain size:",xtrain.shape)

print("Xtest size:",xtest.shape)

#c. create CNN layers

cnn=keras.models.Sequential()

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

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

cnn.add(keras.layers.BatchNormalization())

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

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

cnn.add(keras.layers.Flatten())

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

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

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

cnn.summary()

#3. compile and test

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

es=keras.callbacks.EarlyStopping(monitor='loss',patience=10,restore\_best\_weights=True)

cp=keras.callbacks.ModelCheckpoint("/content/Sreetu.h5",monitor='val\_loss')

cnn.fit(xtrain,ytrain,epochs=2,batch\_size=16,callbacks=[es,cp])