Prac13

from tensorflow import keras

from keras.datasets import mnist

from matplotlib import pyplot

import numpy as np

from keras.utils import to\_categorical

from keras.models import Sequential

from keras.layers import Dense, Dropout

# -------------------- Loading and Visualizing Data --------------------

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

# Display first 9 images from training set

for i in range(0, 9):

pyplot.subplot(330 + 1 + i)

pyplot.imshow(X\_train[i], cmap=pyplot.get\_cmap('gray'))

pyplot.show()

# -------------------- Preprocessing Data --------------------

seed = 7

np.random.seed(seed)

# Flatten 28x28 images to 784-dimensional vectors

num\_pixels = X\_train.shape[1] \* X\_train.shape[2]

X\_train = X\_train.reshape(X\_train.shape[0], num\_pixels).astype('float32')

X\_test = X\_test.reshape(X\_test.shape[0], num\_pixels).astype('float32')

# Normalize pixel values to 0-1

X\_train /= 255

X\_test /= 255

# One-hot encode target labels

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

num\_classes = y\_test.shape[1]

# -------------------- Defining the Model --------------------

def baseline\_model():

model = Sequential()

model.add(Dense(num\_pixels, input\_dim=num\_pixels, kernel\_initializer='normal', activation='relu'))

model.add(Dense(num\_classes, kernel\_initializer='normal', activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

# -------------------- Training the Model --------------------

model = baseline\_model()

model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=10, batch\_size=200)

# -------------------- Evaluating the Model --------------------

scores = model.evaluate(X\_test, y\_test, verbose=0)

print("Baseline Error: %.2f%%" % (100 - scores[1] \* 100))