from keras import Sequential

from keras.datasets import mnist

import numpy as np

from keras.layers import Dense

from keras.utils import to\_categorical

(train\_images,train\_labels),(test\_images, test\_labels) = mnist.load\_data()

print(train\_images.shape[1:])

#process the data

#1. convert each image of shape 28\*28 to 784 dimensional which will be fed to the network as a single feature

dimData = np.prod(train\_images.shape[1:])

print(dimData)

train\_data = train\_images.reshape(train\_images.shape[0],dimData)

test\_data = test\_images.reshape(test\_images.shape[0],dimData)

#convert data to float and scale values between 0 and 1

train\_data = train\_data.astype('float')

test\_data = test\_data.astype('float')

#scale data

train\_data /=255.0

test\_data /=255.0

#change the labels frominteger to one-hot encoding. to\_categorical is doing the same thing as LabelEncoder()

train\_labels\_one\_hot = to\_categorical(train\_labels)

test\_labels\_one\_hot = to\_categorical(test\_labels)

#creating network

model = Sequential()

model.add(Dense(512, activation='relu', input\_shape=(dimData,)))

model.add(Dense(512, activation='relu'))

model.add(Dense(10, activation='softmax'))

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

history = model.fit(train\_data, train\_labels\_one\_hot, batch\_size=256, epochs=10, verbose=1,

                   validation\_data=(test\_data, test\_labels\_one\_hot))

import matplotlib.pyplot as plt

# Plot training & validation loss values

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper right')

plt.show()

# Plot training & validation accuracy values

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='lower right')

plt.show()

from google.colab import drive

drive.mount('/content/gdrive')

path\_to\_csv = '/content/diabetes.csv'

import keras

import pandas

from keras.models import Sequential

from keras.layers.core import Dense, Activation

# load dataset

from sklearn.model\_selection import train\_test\_split

import pandas as pd

import numpy as np

dataset = pd.read\_csv(path\_to\_csv, header=None).values

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(dataset[:,0:8], dataset[:,8], test\_size=0.25, random\_state=87)

np.random.seed(155)

my\_first\_nn = Sequential() # create model

my\_first\_nn.add(Dense(20, input\_dim=8, activation='relu')) # hidden layer

my\_first\_nn.add(Dense(1, activation='sigmoid')) # output layer

my\_first\_nn.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['acc'])

my\_first\_nn\_fitted = my\_first\_nn.fit(X\_train, Y\_train, epochs=100,

                                     initial\_epoch=0)

print(my\_first\_nn.summary())

print(my\_first\_nn.evaluate(X\_test, Y\_test))

np.random.seed(155)

my\_second\_nn = Sequential() # create model

my\_second\_nn.add(Dense(20, input\_dim=8, activation='relu')) # hidden layer

my\_second\_nn.add(Dense(10, activation='relu')) # hidden layer 2

my\_second\_nn.add(Dense(5, activation='relu')) # hidden layer 3

my\_second\_nn.add(Dense(1, activation='sigmoid')) # output layer

my\_second\_nn.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['acc'])

my\_second\_nn\_fitted = my\_second\_nn.fit(X\_train, Y\_train, epochs=100,

                                     initial\_epoch=0)

print(my\_second\_nn.summary())

print(my\_second\_nn.evaluate(X\_test, Y\_test))