Neural Network and Deep learning: ICP2

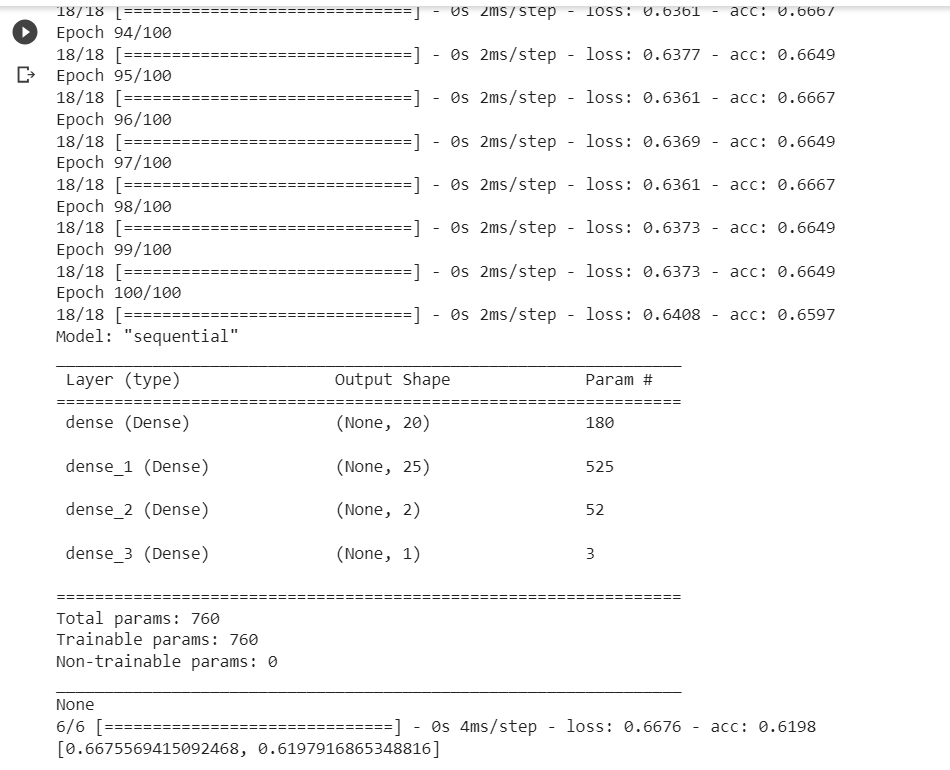
Mallika Mamidi

700746126

Github Link: <https://github.com/mallika76/NN-ICP2.git>

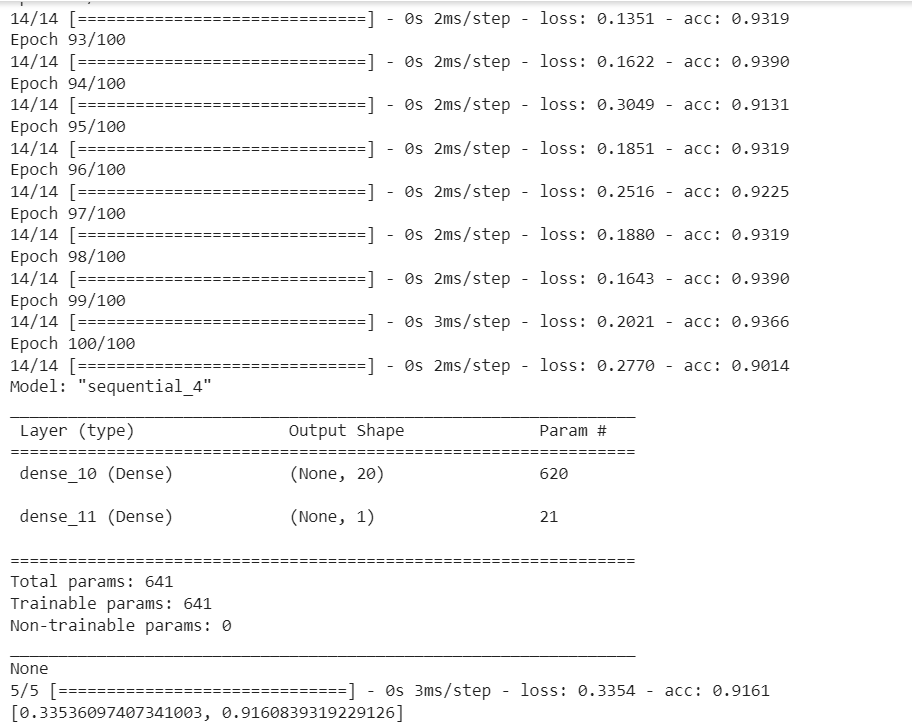
Problem 1: Use the use case in the class: a. Add more Dense layers to the existing code and check how the accuracy changes





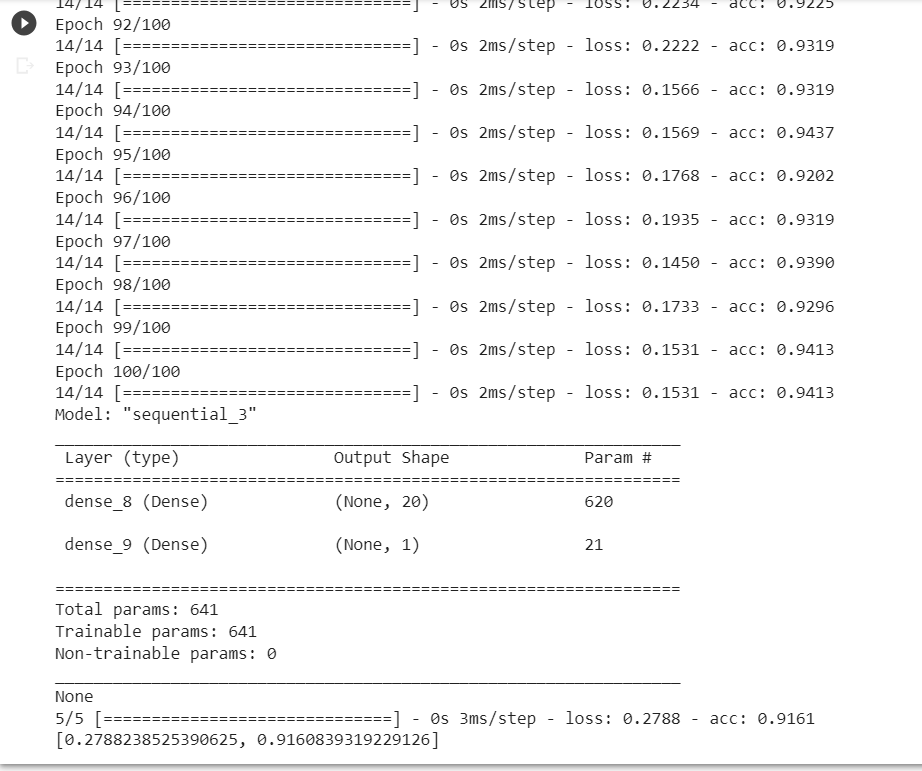
Change the data source to Breast Cancer dataset \* available in the source code folder and make required changes. Report accuracy of the model.





Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below). from sklearn.preprocessing import StandardScaler sc = StandardScaler()





Q2.

Plot the loss and accuracy for both training data and validation data using the history object in the source code.

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout

import matplotlib.pyplot as plt

# load MNIST dataset

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

# normalize pixel values to range [0, 1]

x\_train = x\_train.astype('float32') / 255

x\_test = x\_test.astype('float32') / 255

# Convert the labels to categorical

num\_classes = 10

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

# create a model

model = Sequential()

model.add(Dense(512, activation='relu', input\_shape=(784,)))

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

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

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

# train the model and record the training history

history = model.fit(x\_train.reshape(-1, 784), y\_train, validation\_data=(x\_test.reshape(-1, 784), y\_test),

epochs=20, batch\_size=128)

# plot the training and validation accuracy and loss curves

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.title('Model Loss')

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')

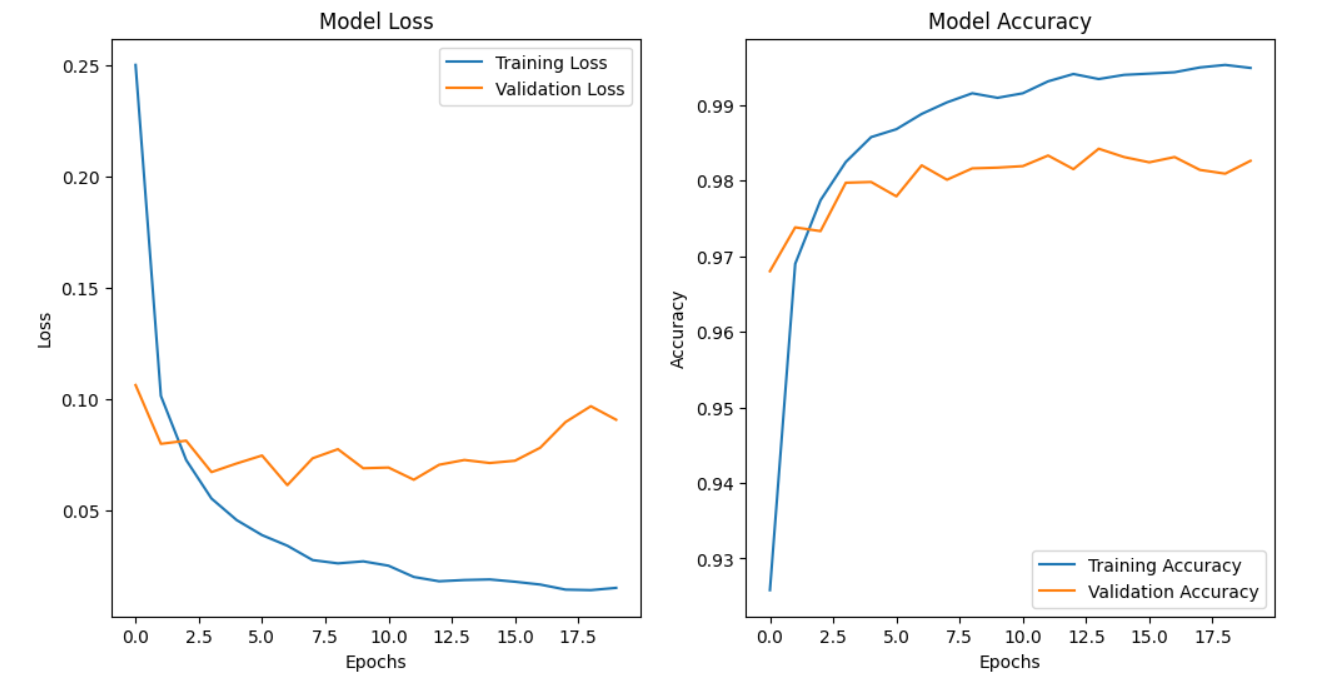
plt.ylabel('Accuracy')

plt.title('Model Accuracy')

plt.legend()

#plt.tight\_layout()

plt.show()



Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout

import matplotlib.pyplot as plt

import numpy as np

# load MNIST dataset

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

# normalize pixel values to range [0, 1]

x\_train = x\_train.astype('float32') / 255

x\_test = x\_test.astype('float32') / 255

# convert class labels to binary class matrices

num\_classes = 10

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

# create a simple neural network model

model = Sequential()

model.add(Dense(512, activation='relu', input\_shape=(784,)))

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

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

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

# train the model

model.fit(x\_train.reshape(-1, 784), y\_train, validation\_data=(x\_test.reshape(-1, 784), y\_test),

epochs=20, batch\_size=128)

# make a prediction on the image using the trained model

prediction = model.predict(x\_test[0].reshape(1, -1))

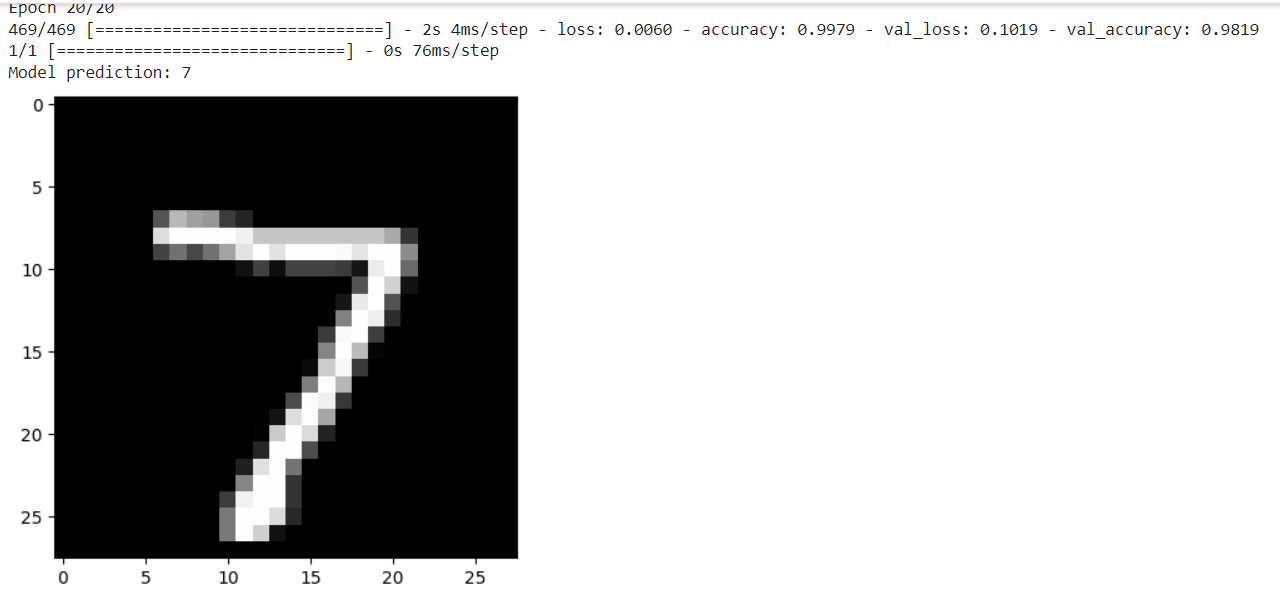
print('Model prediction:', np.argmax(prediction))

# plot one of the images in the test data

plt.imshow(x\_test[0], cmap='gray')

plt.figure(figsize=(10,6))

plt.show()



Q3

We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout

import matplotlib.pyplot as plt

import numpy as np

# load MNIST dataset

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

# normalize pixel values to range [0, 1]

x\_train = x\_train.astype('float32') / 255

x\_test = x\_test.astype('float32') / 255

# convert class labels to binary class matrices

num\_classes = 10

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

# create a list of models to train

models = []

# model with 1 hidden layer and tanh activation

model = Sequential()

model.add(Dense(512, activation='tanh', input\_shape=(784,)))

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

models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation

model = Sequential()

model.add(Dense(512, activation='sigmoid', input\_shape=(784,)))

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

models.append(('1 hidden layer with sigmoid', model))

# model with 2 hidden layers and sigmoid activation

model = Sequential()

model.add(Dense(512, activation='sigmoid', input\_shape=(784,)))

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

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

models.append(('2 hidden layers with sigmoid', model))

# train each model and plot loss and accuracy curves

for name, model in models:

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

history = model.fit(x\_train.reshape(-1, 784), y\_train, validation\_data=(x\_test.reshape(-1, 784), y\_test),

epochs=20, batch\_size=128, verbose=0)

# plot loss and accuracy curves

plt.plot(history.history['loss'], label='train\_loss')

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

plt.plot(history.history['accuracy'], label='train\_accuracy')

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

plt.title(name)

plt.xlabel('Epoch')

plt.legend()

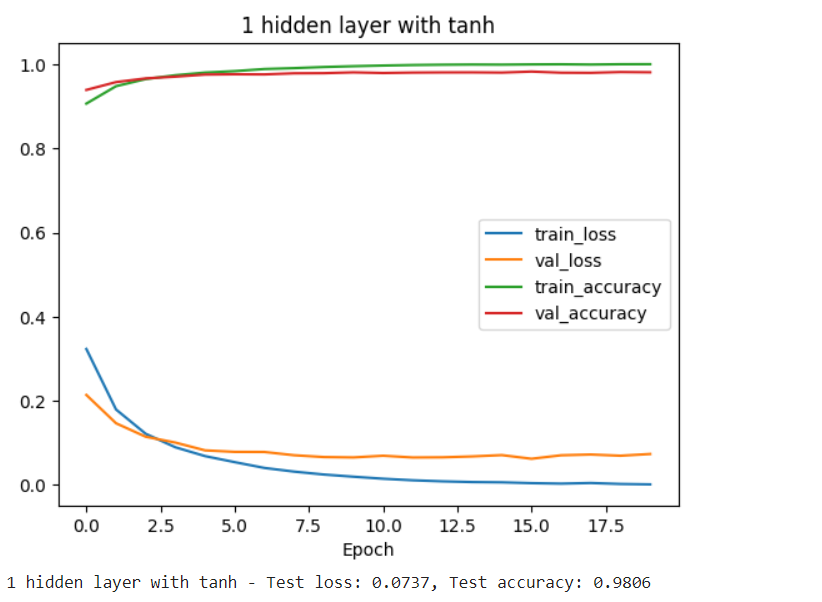
plt.show()

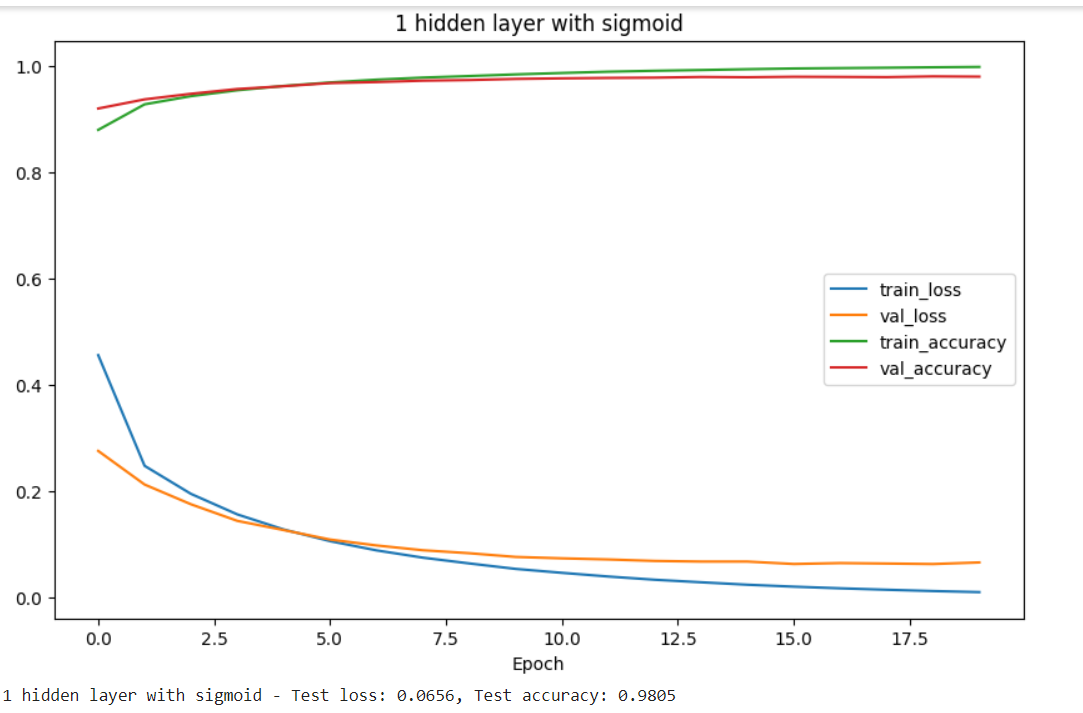
plt.figure(figsize=(10,6))

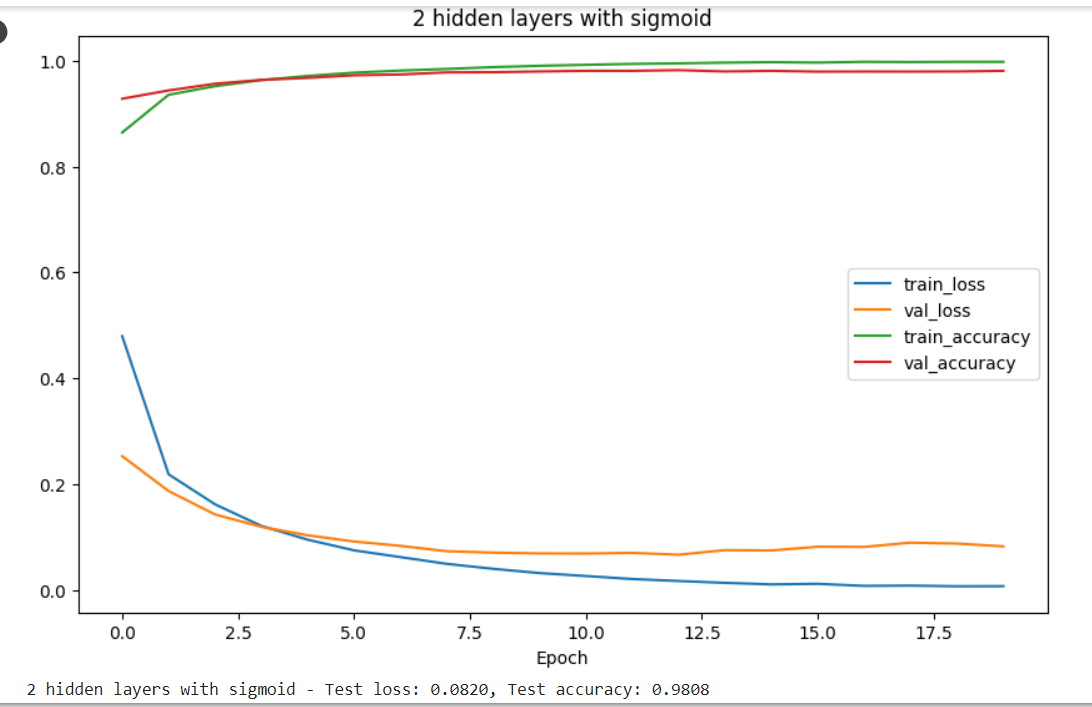
# evaluate the model on test data

loss, accuracy = model.evaluate(x\_test.reshape(-1, 784), y\_test, verbose=0)

print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))







Q4.

Run the same code without scaling the images and check the performance?

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout

import matplotlib.pyplot as plt

import numpy as np

# load MNIST dataset

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

# convert class labels to binary class matrices

num\_classes = 10

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

# create a list of models to train

models = []

# model with 1 hidden layer and tanh activation

model = Sequential()

model.add(Dense(512, activation='tanh', input\_shape=(784,)))

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

models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation

model = Sequential()

model.add(Dense(512, activation='sigmoid', input\_shape=(784,)))

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

models.append(('1 hidden layer with sigmoid', model))

# model with 2 hidden layers and sigmoid activation

model = Sequential()

model.add(Dense(512, activation='sigmoid', input\_shape=(784,)))

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

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

models.append(('2 hidden layers with sigmoid', model))

# train each model and plot loss and accuracy curves

for name, model in models:

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

history = model.fit(x\_train.reshape(-1, 784), y\_train, validation\_data=(x\_test.reshape(-1, 784), y\_test),

epochs=20, batch\_size=128, verbose=0)

# plot loss and accuracy curves

plt.plot(history.history['loss'], label='train\_loss')

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

plt.plot(history.history['accuracy'], label='train\_accuracy')

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

plt.title(name)

plt.xlabel('Epoch')

plt.legend()

plt.figure(figsize=(10,6))

plt.show()

# evaluate the model on test data

loss, accuracy = model.evaluate(x\_test.reshape(-1, 784), y\_test, verbose=0)

print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))

