

NNDL - Assignment - 6 ICP_Basics in Keras

Use Case Description: Predicting the diabetes disease

Programming elements: Keras Basics

In class programming:

1) Use the use case in the class:

a) Add more Dense layers to the existing code and check how the accuracy changes

Ans

Code before adding the dense layers is -

The screenshot shows a Google Colab interface. At the top, there's a menu bar with options like Chrome, File, Edit, View, History, Bookmarks, Profiles, Tab, Window, Help, and a search bar. Below the menu is a toolbar with icons for back, forward, refresh, and other navigation. The main area is a Jupyter notebook cell titled 'Assignment2_q1.ipynb'. The cell contains Python code for a neural network. The code imports necessary libraries (drive, keras, pandas, Sequential, Dense, Activation), loads a dataset from a CSV file, splits it into training and testing sets, creates a Sequential model with one hidden layer (20 units, ReLU activation) and one output layer (1 unit, sigmoid activation), compiles the model with binary crossentropy loss and Adam optimizer, fits the model to the training data, and evaluates it on the test data. The output of the cell shows the model's summary and evaluation results. The bottom status bar indicates the notebook is mounted at /content/gdrive and shows epoch information.

```
#1 (a)#Use the use case in the class:  
from google.colab import drive  
drive.mount('/content/gdrive')  
path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/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))  
  
Mounted at /content/gdrive  
Epoch 1/100  
18/18 [=====] - 1s 2ms/step - loss: 6.1983 - acc: 0.5747
```

Accuracy of the model before adding the code is -

The screenshot shows a Google Colab notebook titled "Assignment2_q1.ipynb". The code cell displays training logs for a sequential model with 100 epochs. The logs show increasing accuracy from ~0.56 to ~0.73. A table below the logs shows the model's architecture with two dense layers of 20 and 1 nodes respectively, totaling 201 parameters. The notebook interface includes tabs for "Code" and "Text", and a toolbar with various icons.

```
Epoch 92/100
18/18 [=====] - 0s 3ms/step - loss: 0.5645 - acc: 0.7326
Epoch 93/100
18/18 [=====] - 0s 3ms/step - loss: 0.5596 - acc: 0.7205
Epoch 94/100
18/18 [=====] - 0s 3ms/step - loss: 0.5558 - acc: 0.7326
Epoch 95/100
18/18 [=====] - 0s 2ms/step - loss: 0.5501 - acc: 0.7257
Epoch 96/100
18/18 [=====] - 0s 2ms/step - loss: 0.5612 - acc: 0.7274
Epoch 97/100
18/18 [=====] - 0s 2ms/step - loss: 0.5625 - acc: 0.7274
Epoch 98/100
18/18 [=====] - 0s 2ms/step - loss: 0.5691 - acc: 0.7274
Epoch 99/100
18/18 [=====] - 0s 2ms/step - loss: 0.5626 - acc: 0.7257
Epoch 100/100
18/18 [=====] - 0s 2ms/step - loss: 0.5525 - acc: 0.7396
Model: "sequential"

Layer (type)      Output Shape         Param #
dense (Dense)    (None, 20)           180
dense_1 (Dense)  (None, 1)            21
=====
Total params: 201
Trainable params: 201
Non-trainable params: 0
```

f11. #1(a)... Add more Dense layers to the existing code and check how the accuracy changes.

Adding more dense layers to the code -

- 1) We can easily add more dense layers to the existing code , we can simply add more dense layers to the sequential model.
- 2) Here I have added two additional dense layers with 10 and 5 nodes.

Code after adding the dense layers is -

The screenshot shows a Google Colab notebook titled "Assignment2_q1.ipynb". The code in cell [1] adds more dense layers to an existing model:

```
[1] #1(a). Add more Dense layers to the existing code and check how the accuracy changes.
# added more dense layers to the above existing code
from google.colab import drive
drive.mount('/content/gdrive')
path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/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(10,activation='relu'))#additional hidden layer with node 10
my_first_nn.add(Dense(5,activation='relu'))#additional hidden layer with node 5
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))
```

Accuracy of the model after adding the additional dense layers is

The screenshot shows the output of the trained neural network. The accuracy has improved from 0.7604 to 0.7726.

```
18/18 [=====] - 0s 2ms/step - loss: 0.4881 - acc: 0.7656
[1] Epoch 94/100
18/18 [=====] - 0s 2ms/step - loss: 0.4843 - acc: 0.7691
Epoch 95/100
18/18 [=====] - 0s 2ms/step - loss: 0.4894 - acc: 0.7604
Epoch 96/100
18/18 [=====] - 0s 2ms/step - loss: 0.4787 - acc: 0.7639
Epoch 97/100
18/18 [=====] - 0s 2ms/step - loss: 0.4810 - acc: 0.7760
Epoch 98/100
18/18 [=====] - 0s 2ms/step - loss: 0.4877 - acc: 0.7552
Epoch 99/100
18/18 [=====] - 0s 2ms/step - loss: 0.5002 - acc: 0.7552
Epoch 100/100
18/18 [=====] - 0s 3ms/step - loss: 0.4739 - acc: 0.7726
Model: "sequential"
-----  

Layer (type)          Output Shape         Param #
-----  

dense (Dense)          (None, 20)           180  

dense_1 (Dense)        (None, 10)            210  

dense_2 (Dense)        (None, 5)             55  

dense_3 (Dense)        (None, 1)              6  

-----  

Total params: 451
Trainable params: 451
Non-trainable params: 0
-----  

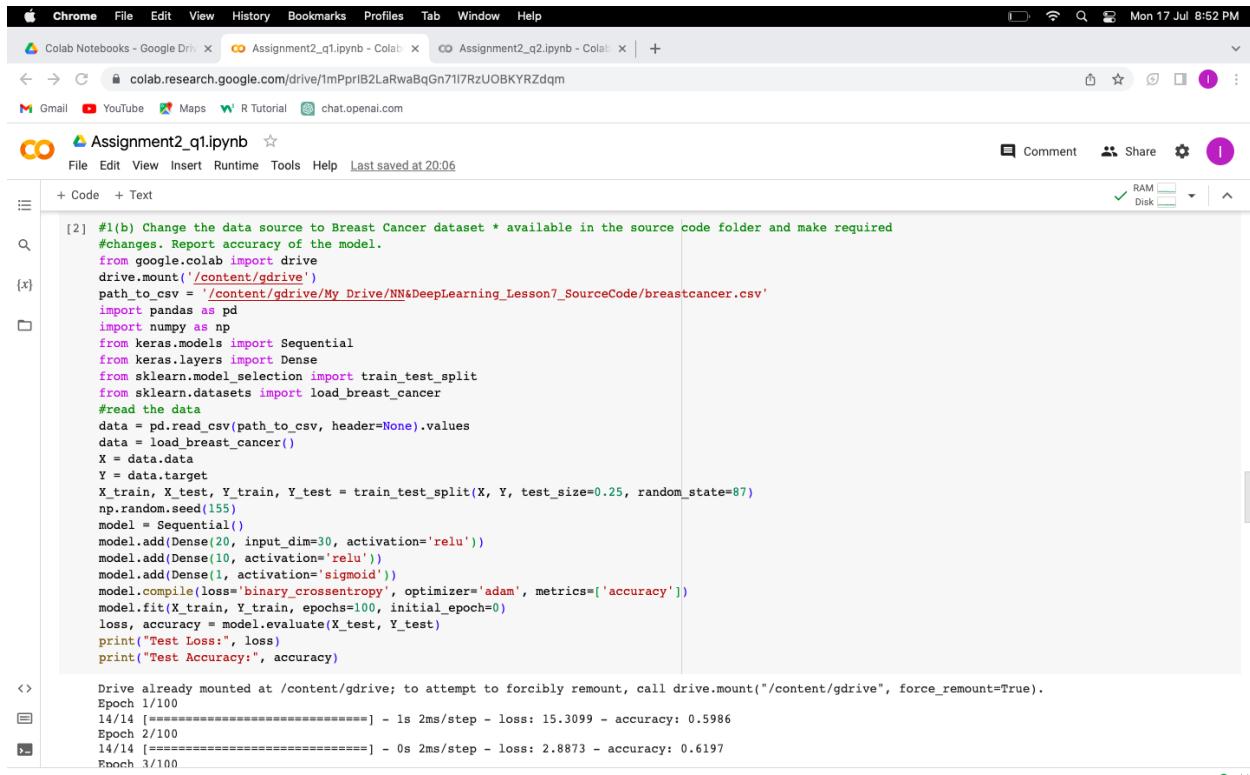
None
6/6 [=====] - 0s 3ms/step - loss: 0.5920 - acc: 0.7448
[0.5919528603553772, 0.7447916865348816]
```

Accuracy of the model increased from 0.6562 to 0.7448 after adding 2 more dense layers. Therefore, by adding additional layers the model has more parameters to learn from the training data which may lead to better accuracy.

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

Ans)

Code change after changing the data source to Breast Cancer dataset



The screenshot shows a Google Colab notebook titled "Assignment2_q1.ipynb". The code cell contains Python code to change the data source to a local CSV file and train a neural network. The output cell shows the training logs and evaluation results.

```
[2] #1(b) Change the data source to Breast Cancer dataset * available in the source code folder and make required changes. Report accuracy of the model.
from google.colab import drive
drive.mount('/content/gdrive')
path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/breastcancer.csv'
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
#read the data
data = pd.read_csv(path_to_csv, header=None).values
data = load_breast_cancer()
X = data.data
Y = data.target
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)
np.random.seed(155)
model = Sequential()
model.add(Dense(20, input_dim=30, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=100, initial_epoch=0)
loss, accuracy = model.evaluate(X_test, Y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)

<> Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
Epoch 1/100
14/14 [=====] - 1s 2ms/step - loss: 15.3099 - accuracy: 0.5986
Epoch 2/100
14/14 [=====] - 0s 2ms/step - loss: 2.8873 - accuracy: 0.6197
Epoch 3/100
```

Accuracy of the model after changing the data source to the breast cancer :

```

Assignment2_q1.ipynb
File Edit View Insert Runtime Tools Help Last saved at 20:06
+ Code + Text
[2] 14/14 [=====] - 0s 3ms/step - loss: 0.1792 - accuracy: 0.9319
14/14 [=====] - 0s 3ms/step - loss: 0.1538 - accuracy: 0.9343
14/14 [=====] - 0s 3ms/step - loss: 0.1610 - accuracy: 0.9319
14/14 [=====] - 0s 3ms/step - loss: 0.1498 - accuracy: 0.9531
14/14 [=====] - 0s 3ms/step - loss: 0.1727 - accuracy: 0.9343
14/14 [=====] - 0s 3ms/step - loss: 0.2580 - accuracy: 0.9108
14/14 [=====] - 0s 3ms/step - loss: 0.1653 - accuracy: 0.9413
14/14 [=====] - 0s 3ms/step - loss: 0.2499 - accuracy: 0.9178
14/14 [=====] - 0s 3ms/step - loss: 0.1655 - accuracy: 0.9272
14/14 [=====] - 0s 3ms/step - loss: 0.1604 - accuracy: 0.9460
14/14 [=====] - 0s 3ms/step - loss: 0.1800 - accuracy: 0.9249
14/14 [=====] - 0s 3ms/step - loss: 0.1590 - accuracy: 0.9413
14/14 [=====] - 0s 4ms/step - loss: 0.1645 - accuracy: 0.9390
14/14 [=====] - 0s 3ms/step - loss: 0.1425 - accuracy: 0.9343
14/14 [=====] - 0s 5ms/step - loss: 0.1478 - accuracy: 0.9437
14/14 [=====] - 0s 3ms/step - loss: 0.1510 - accuracy: 0.9413
<> 100/100
14/14 [=====] - 0s 3ms/step - loss: 0.1752 - accuracy: 0.9202
5/5 [=====] - 0s 4ms/step - loss: 0.2583 - accuracy: 0.9161
Test Loss: 0.258388090133667
Test Accuracy: 0.9160839319229126

```

c) 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()
Ans)

```

```

Assignment2_q1.ipynb
File Edit View Insert Runtime Tools Help Last saved at 20:06
+ Code + Text
[4] #1(c) 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()

from google.colab import drive
drive.mount('/content/gdrive')
path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/breastcancer.csv'
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler

#read the data
data = pd.read_csv(path_to_csv, header=None).values
data = load_breast_cancer()
X = data.data
Y = data.target
scaler = StandardScaler()
X = scaler.fit_transform(X)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)
np.random.seed(155)
model = Sequential()
model.add(Dense(20, input_dim=30, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=100, initial_epoch=0)
loss, accuracy = model.evaluate(X_test, Y_test)
print("Test Loss:", loss)

```

Output:

Chrome File Edit View History Bookmarks Profiles Tab Window Help

colab.research.google.com/drive/1mPpriB2LaRwaBqGn7Ii7RzUOBKYRZdqm

Gmail YouTube Maps R Tutorial chat.openai.com

Assignment2_q1.ipynb

File Edit View Insert Runtime Tools Help Last saved at 20:06

+ Code + Text

```
[4]: loss, accuracy = model.evaluate(X_test, Y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)

[4]: [=====] - 0s 3ms/step - loss: 0.0114 - accuracy: 0.9977
Epoch 74/100
[4]: [=====] - 0s 3ms/step - loss: 0.0111 - accuracy: 0.9977
Epoch 75/100
[4]: [=====] - 0s 3ms/step - loss: 0.0108 - accuracy: 0.9977
Epoch 76/100
[4]: [=====] - 0s 3ms/step - loss: 0.0107 - accuracy: 0.9977
Epoch 77/100
[4]: [=====] - 0s 3ms/step - loss: 0.0107 - accuracy: 0.9977
Epoch 78/100
[4]: [=====] - 0s 3ms/step - loss: 0.0102 - accuracy: 0.9977
Epoch 79/100
[4]: [=====] - 0s 3ms/step - loss: 0.0099 - accuracy: 0.9977
Epoch 80/100
[4]: [=====] - 0s 3ms/step - loss: 0.0095 - accuracy: 0.9977
Epoch 81/100
[4]: [=====] - 0s 3ms/step - loss: 0.0094 - accuracy: 0.9977
Epoch 82/100
[4]: [=====] - 0s 3ms/step - loss: 0.0089 - accuracy: 0.9977
Epoch 83/100
[4]: [=====] - 0s 3ms/step - loss: 0.0091 - accuracy: 0.9977
Epoch 84/100
[4]: [=====] - 0s 3ms/step - loss: 0.0089 - accuracy: 0.9977
Epoch 85/100
[4]: [=====] - 0s 3ms/step - loss: 0.0086 - accuracy: 0.9977
Epoch 86/100
[4]: [=====] - 0s 3ms/step - loss: 0.0084 - accuracy: 0.9977
Epoch 87/100
[4]: [=====] - 0s 3ms/step - loss: 0.0081 - accuracy: 0.9977
Epoch 88/100
[4]: [=====] - 0s 3ms/step - loss: 0.0077 - accuracy: 1.0000
Epoch 89/100
[4]: [=====] - 0s 3ms/step - loss: 0.0079 - accuracy: 1.0000
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help

colab.research.google.com/drive/1mPpriB2LaRwaBqGn7Ii7RzUOBKYRZdqm

Gmail YouTube Maps R Tutorial chat.openai.com

Assignment2_q1.ipynb

File Edit View Insert Runtime Tools Help Last saved at 20:06

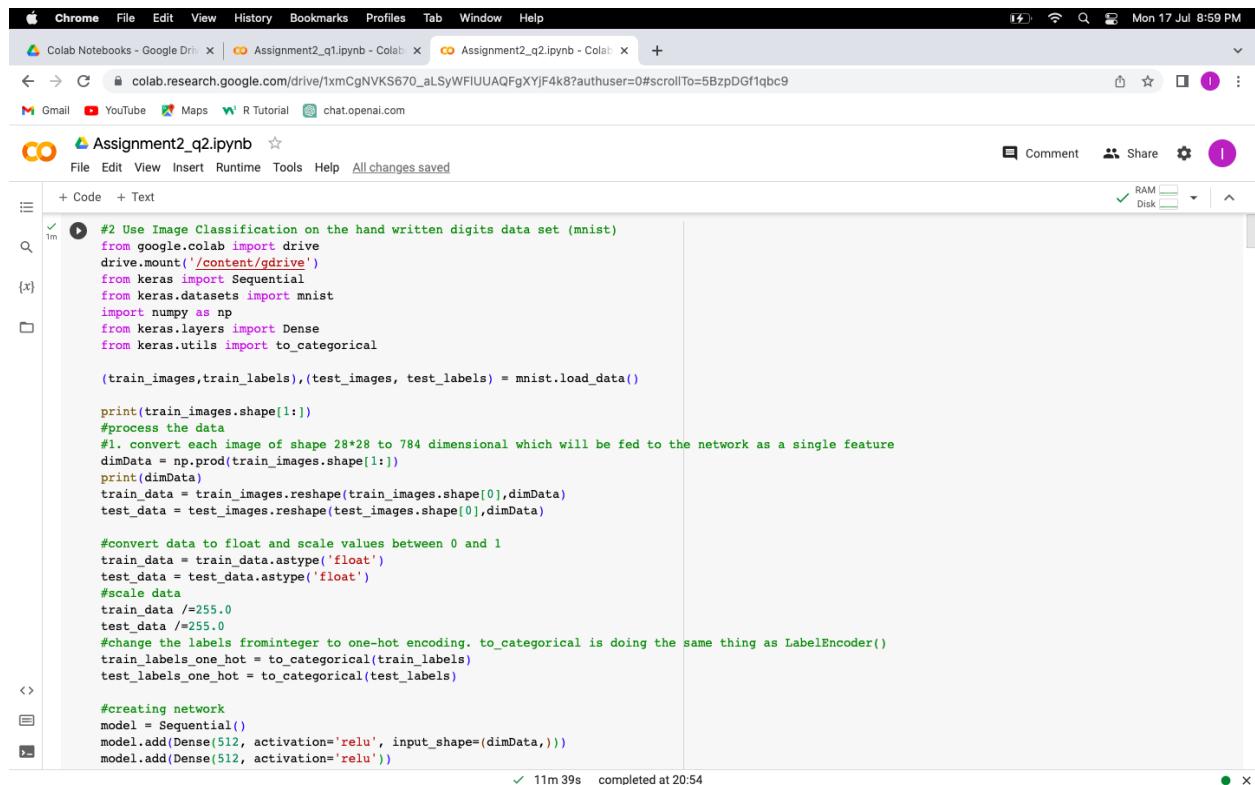
+ Code + Text

```
[4]: Epoch 87/100
[4]: [=====] - 0s 3ms/step - loss: 0.0081 - accuracy: 0.9977
Epoch 88/100
[4]: [=====] - 0s 3ms/step - loss: 0.0077 - accuracy: 1.0000
Epoch 89/100
[4]: [=====] - 0s 3ms/step - loss: 0.0079 - accuracy: 1.0000
Epoch 90/100
[4]: [=====] - 0s 3ms/step - loss: 0.0074 - accuracy: 0.9977
Epoch 91/100
[4]: [=====] - 0s 4ms/step - loss: 0.0072 - accuracy: 0.9977
Epoch 92/100
[4]: [=====] - 0s 3ms/step - loss: 0.0067 - accuracy: 0.9977
Epoch 93/100
[4]: [=====] - 0s 3ms/step - loss: 0.0066 - accuracy: 1.0000
Epoch 94/100
[4]: [=====] - 0s 3ms/step - loss: 0.0064 - accuracy: 1.0000
Epoch 95/100
[4]: [=====] - 0s 3ms/step - loss: 0.0062 - accuracy: 1.0000
Epoch 96/100
[4]: [=====] - 0s 3ms/step - loss: 0.0061 - accuracy: 1.0000
Epoch 97/100
[4]: [=====] - 0s 3ms/step - loss: 0.0059 - accuracy: 1.0000
Epoch 98/100
[4]: [=====] - 0s 3ms/step - loss: 0.0057 - accuracy: 1.0000
Epoch 99/100
[4]: [=====] - 0s 4ms/step - loss: 0.0055 - accuracy: 1.0000
Epoch 100/100
[4]: [=====] - 0s 3ms/step - loss: 0.0054 - accuracy: 1.0000
5/5 [=====] - 0s 4ms/step - loss: 0.1458 - accuracy: 0.9720
Test Loss: 0.1457810401916504
Test Accuracy: 0.9720279574394226
```

2) Use Image Classification on the handwritten digits data set (mnist)

Ans)

Code for the above is



The screenshot shows a Google Colab notebook titled "Assignment2_q2.ipynb". The code cell contains Python code for image classification using the Keras Sequential model on the MNIST dataset. The code includes importing libraries, loading the dataset, processing the data (reshaping and scaling), and creating a simple neural network with two layers. The code cell has a green checkmark indicating it is saved.

```
#2 Use Image Classification on the hand written digits data set (mnist)
from google.colab import drive
drive.mount('/content/gdrive')
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'))
```

Output:

The screenshot shows a Google Colab notebook titled "Assignment2_q2.ipynb". The code cell contains Python code for building a sequential model with three layers and compiling it with RMSprop optimizer and categorical crossentropy loss. The history object is used to fit the model on training data and validate it on test data. The output shows the training progress over 235 epochs, with metrics like loss, accuracy, val_loss, and val_accuracy printed to the console. The execution time was 11m 39s, completed at 20:54.

```
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))

Mounting at /content/gdrive
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [=====] - 0s 0us/step
(28, 28)
784
Epoch 1/10
235/235 [=====] - 9s 37ms/step - loss: 0.2893 - accuracy: 0.9113 - val_loss: 0.1420 - val_accuracy: 0.9517
Epoch 2/10
235/235 [=====] - 10s 44ms/step - loss: 0.1002 - accuracy: 0.9691 - val_loss: 0.1278 - val_accuracy: 0.9575
Epoch 3/10
235/235 [=====] - 8s 33ms/step - loss: 0.0628 - accuracy: 0.9800 - val_loss: 0.0740 - val_accuracy: 0.9778
Epoch 4/10
235/235 [=====] - 7s 32ms/step - loss: 0.0443 - accuracy: 0.9863 - val_loss: 0.0670 - val_accuracy: 0.9794
Epoch 5/10
235/235 [=====] - 7s 30ms/step - loss: 0.0306 - accuracy: 0.9900 - val_loss: 0.0775 - val_accuracy: 0.9775
Epoch 6/10
235/235 [=====] - 7s 31ms/step - loss: 0.0224 - accuracy: 0.9926 - val_loss: 0.0590 - val_accuracy: 0.9825
Epoch 7/10
235/235 [=====] - 7s 30ms/step - loss: 0.0157 - accuracy: 0.9952 - val_loss: 0.0772 - val_accuracy: 0.9775
Epoch 8/10
235/235 [=====] - 7s 29ms/step - loss: 0.0138 - accuracy: 0.9955 - val_loss: 0.0637 - val_accuracy: 0.9833
Epoch 9/10
235/235 [=====] - 8s 34ms/step - loss: 0.0085 - accuracy: 0.9973 - val_loss: 0.0870 - val_accuracy: 0.9787
Epoch 10/10
235/235 [=====] - 7s 28ms/step - loss: 0.0070 - accuracy: 0.9978 - val_loss: 0.0685 - val_accuracy: 0.9833
```

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

Ans:

Code is

A screenshot of a Google Colab notebook titled "Assignment2_q2.ipynb". The code cell contains Python code for a neural network. The code imports necessary libraries, loads MNIST data, normalizes pixel values, converts class labels to binary matrices, creates a sequential model with two hidden layers and two dropout layers, compiles it with categorical crossentropy loss, Adam optimizer, and accuracy metric, and trains the model for 20 epochs. The status bar at the bottom indicates the process completed at 20:54.

```

#2(a) Plot the loss and accuracy for both training data and validation data using the history object in the source
#code
from google.colab import drive
drive.mount('/content/gdrive')
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 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(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# train the model and record the training history

```

Output:

A screenshot of a Google Colab notebook titled "Assignment2_q2.ipynb". The code cell contains Python code to fit the model and plot training and validation accuracy and loss curves. The output shows the model's performance over 20 epochs, with accuracy increasing from ~0.9270 to ~0.9799 and loss decreasing from ~0.2444 to ~0.0555. The status bar at the bottom indicates the process completed at 20:54.

```

# 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)

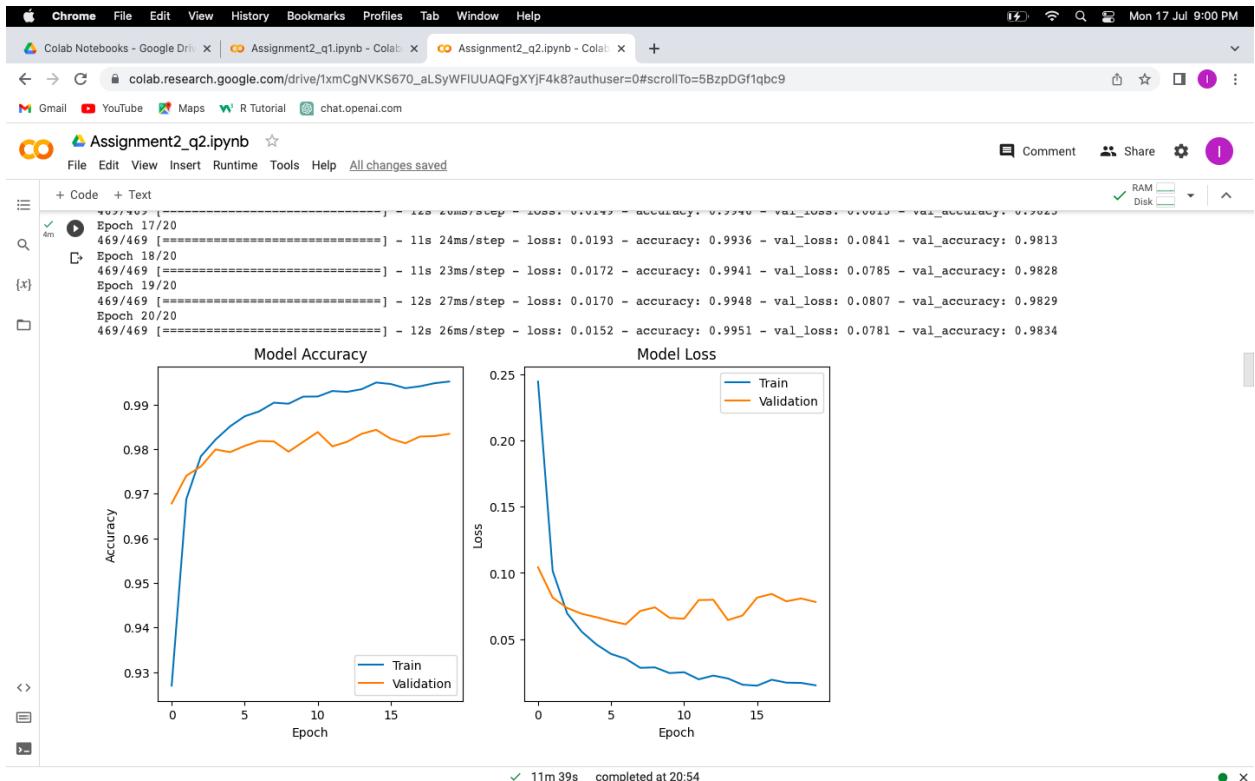
# plot the training and validation accuracy and loss curves
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='lower right')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.show()


```

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
Epoch 1/20
469/469 [=====] - 15s 30ms/step - loss: 0.2444 - accuracy: 0.9270 - val_loss: 0.1042 - val_accuracy: 0.9678
Epoch 2/20
469/469 [=====] - 12s 25ms/step - loss: 0.1015 - accuracy: 0.9687 - val_loss: 0.0813 - val_accuracy: 0.9740
Epoch 3/20
469/469 [=====] - 11s 24ms/step - loss: 0.0693 - accuracy: 0.9784 - val_loss: 0.0735 - val_accuracy: 0.9761
Epoch 4/20
469/469 [=====] - 11s 23ms/step - loss: 0.0555 - accuracy: 0.9821 - val_loss: 0.0691 - val_accuracy: 0.9799
Epoch 5/20



b). 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.

The screenshot shows a Google Colab notebook titled "Assignment2_q2.ipynb". The notebook contains Python code for loading the MNIST dataset and creating a simple neural network model.

```

#2(b)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.

from google.colab import drive
drive.mount('/content/gdrive')
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(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

```

A small inset image in the bottom right corner shows a handwritten digit from the MNIST dataset.

Output:

Chrome File Edit View History Bookmarks Profiles Tab Window Help

colab.research.google.com/drive/1xmCgNVKS670_aLSyWFIUUAQFgXYjF4k8?authuser=0#scrollTo=5BzpDGf1qbc9

Gmail YouTube Maps R Tutorial chat.openai.com

Assignment2_q2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

4m

```
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)

# plot one of the images in the test data
plt.imshow(x_test[0], cmap='gray')
plt.show()

# 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))
```

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).

Epoch 1/20
469/469 [=====] - 13s 26ms/step - loss: 0.2477 - accuracy: 0.9259 - val_loss: 0.1050 - val_accuracy: 0.9659
Epoch 2/20
469/469 [=====] - 12s 26ms/step - loss: 0.0998 - accuracy: 0.9691 - val_loss: 0.0770 - val_accuracy: 0.9754
Epoch 3/20
469/469 [=====] - 13s 27ms/step - loss: 0.0718 - accuracy: 0.9775 - val_loss: 0.0655 - val_accuracy: 0.9794
Epoch 4/20
469/469 [=====] - 11s 24ms/step - loss: 0.0567 - accuracy: 0.9824 - val_loss: 0.0631 - val_accuracy: 0.9805
Epoch 5/20
469/469 [=====] - 13s 28ms/step - loss: 0.0440 - accuracy: 0.9853 - val_loss: 0.0635 - val_accuracy: 0.9819
Epoch 6/20
469/469 [=====] - 12s 26ms/step - loss: 0.0384 - accuracy: 0.9874 - val_loss: 0.0717 - val_accuracy: 0.9792
Epoch 7/20
469/469 [=====] - 12s 26ms/step - loss: 0.0349 - accuracy: 0.9884 - val_loss: 0.0654 - val_accuracy: 0.9806
Epoch 8/20
469/469 [=====] - 12s 26ms/step - loss: 0.0305 - accuracy: 0.9898 - val_loss: 0.0704 - val_accuracy: 0.9813
Epoch 9/20
469/469 [=====] - 11s 24ms/step - loss: 0.0277 - accuracy: 0.9906 - val_loss: 0.0795 - val_accuracy: 0.9791
Epoch 10/20
469/469 [=====] - 11s 23ms/step - loss: 0.0241 - accuracy: 0.9919 - val_loss: 0.0680 - val_accuracy: 0.9824

11m 39s completed at 20:54

Chrome File Edit View History Bookmarks Profiles Tab Window Help

colab.research.google.com/drive/1xmCgNVKS670_aLSyWFIUUAQFgXYjF4k8?authuser=0#scrollTo=5BzpDGf1qbc9

Gmail YouTube Maps R Tutorial chat.openai.com

Assignment2_q2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

4m

```
Epoch 16/20  
469/469 [=====] - 12s 25ms/step - loss: 0.0198 - accuracy: 0.9933 - val_loss: 0.0731 - val_accuracy: 0.9821  
Epoch 17/20  
469/469 [=====] - 12s 26ms/step - loss: 0.0178 - accuracy: 0.9937 - val_loss: 0.0698 - val_accuracy: 0.9836  
Epoch 18/20  
469/469 [=====] - 11s 23ms/step - loss: 0.0161 - accuracy: 0.9949 - val_loss: 0.0780 - val_accuracy: 0.9831  
Epoch 19/20  
469/469 [=====] - 12s 25ms/step - loss: 0.0150 - accuracy: 0.9950 - val_loss: 0.0714 - val_accuracy: 0.9839  
Epoch 20/20  
469/469 [=====] - 12s 26ms/step - loss: 0.0158 - accuracy: 0.9949 - val_loss: 0.0856 - val_accuracy: 0.9824
```

0

5

10

15

20

25

0 5 10 15 20 25

1/1 [=====] - 0s 88ms/step

Model prediction: 7

11m 39s completed at 20:54

c.) 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.

Ans)

Code

The screenshot shows a Google Colab notebook titled "Assignment2_q2.ipynb". The code cell contains the following Python script:

```
#2(c)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.

from google.colab import drive
drive.mount('/content/gdrive')
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(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(model)

11m 39s completed at 20:54
```

Mon 17 Jul 9:01 PM

colab.research.google.com/drive/1xmCgNVKS670_aLSyWFIUQAQFgXYjF4k8?authuser=0#scrollTo=5BzpDGf1qbc9

```
+ Code + Text
13m 1 hidden layer with tanh
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(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))

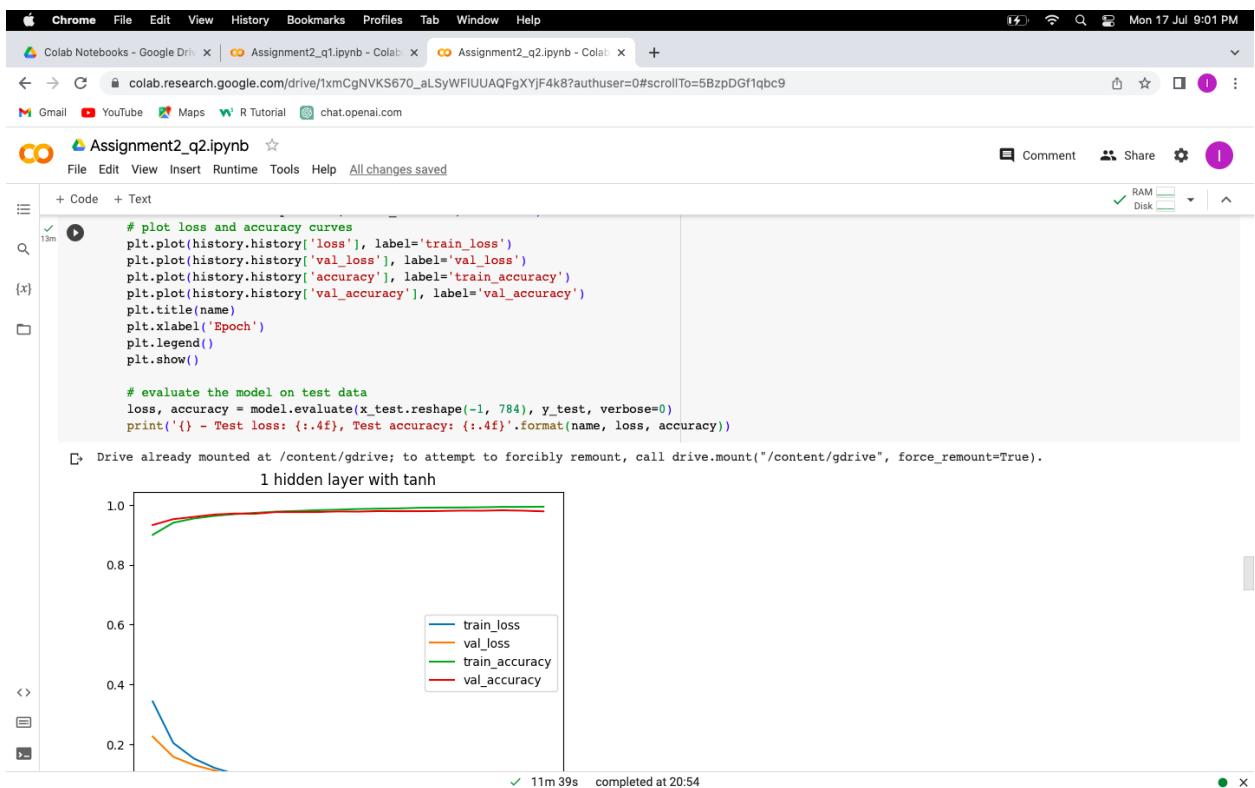
# model with 2 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))

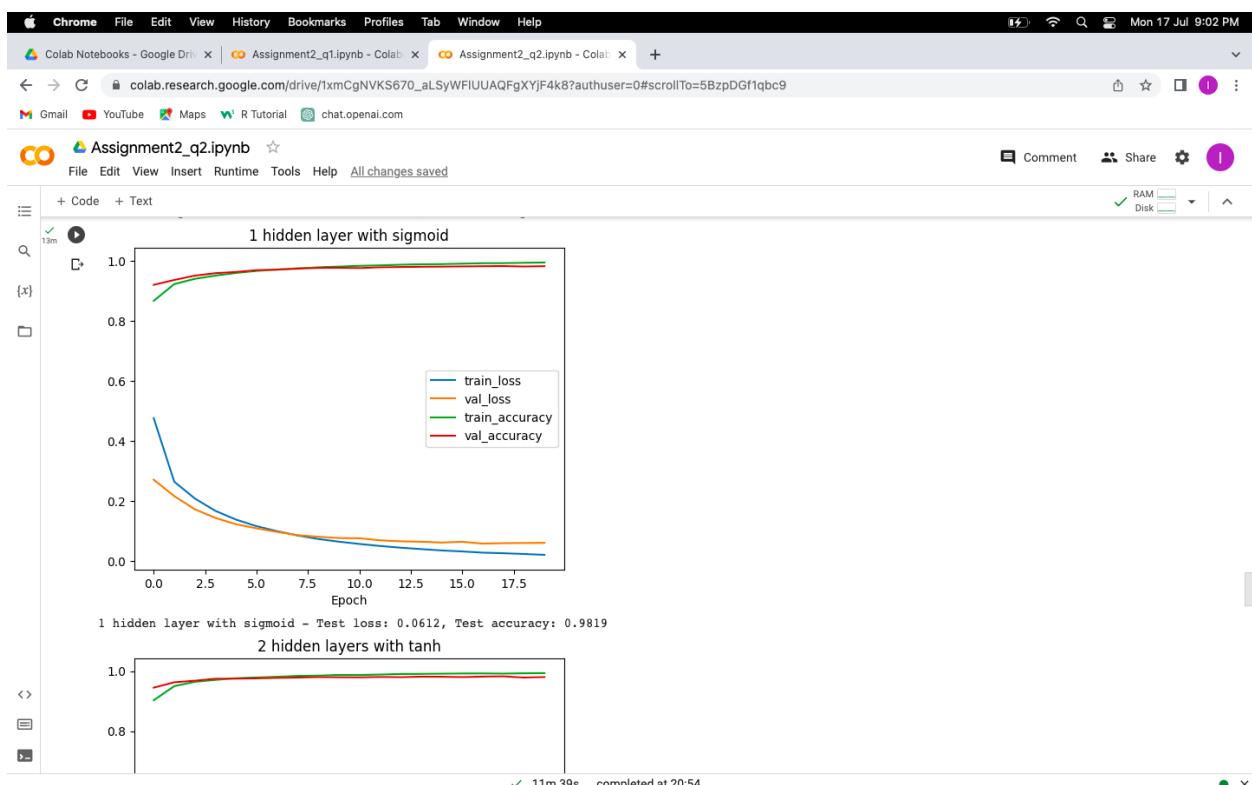
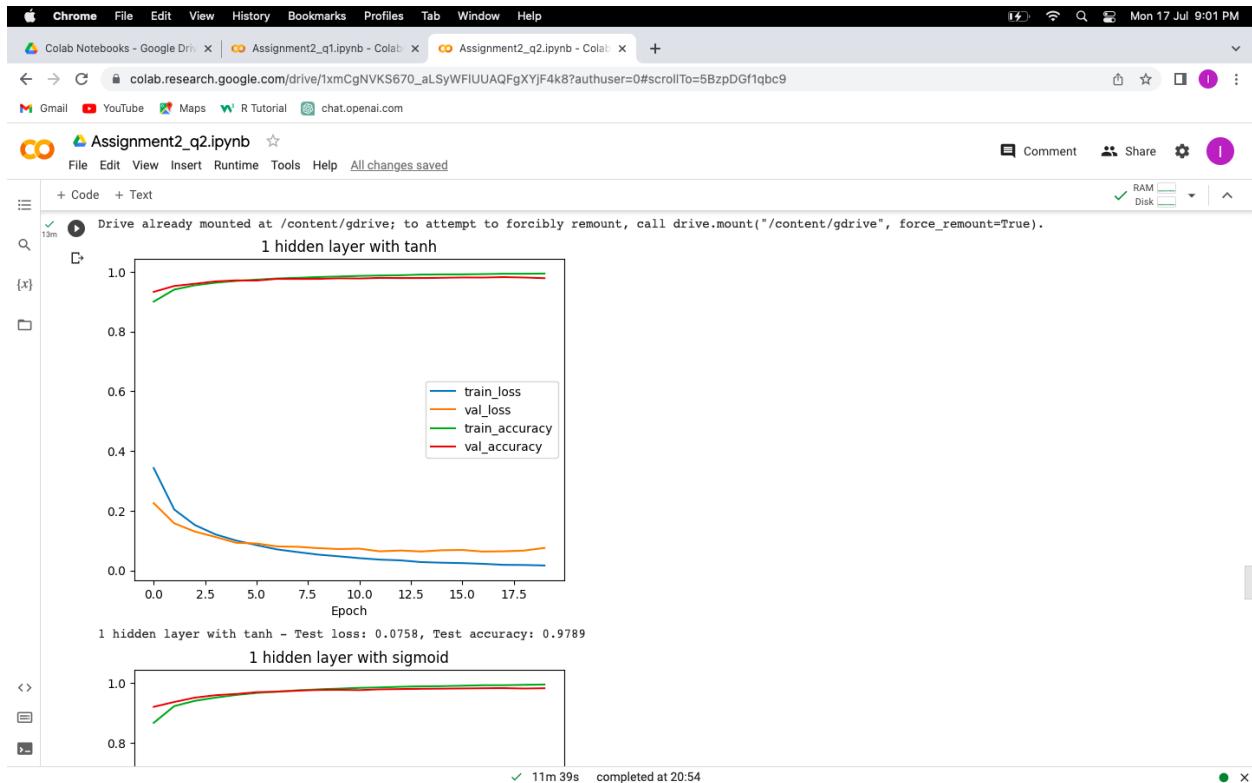
# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))

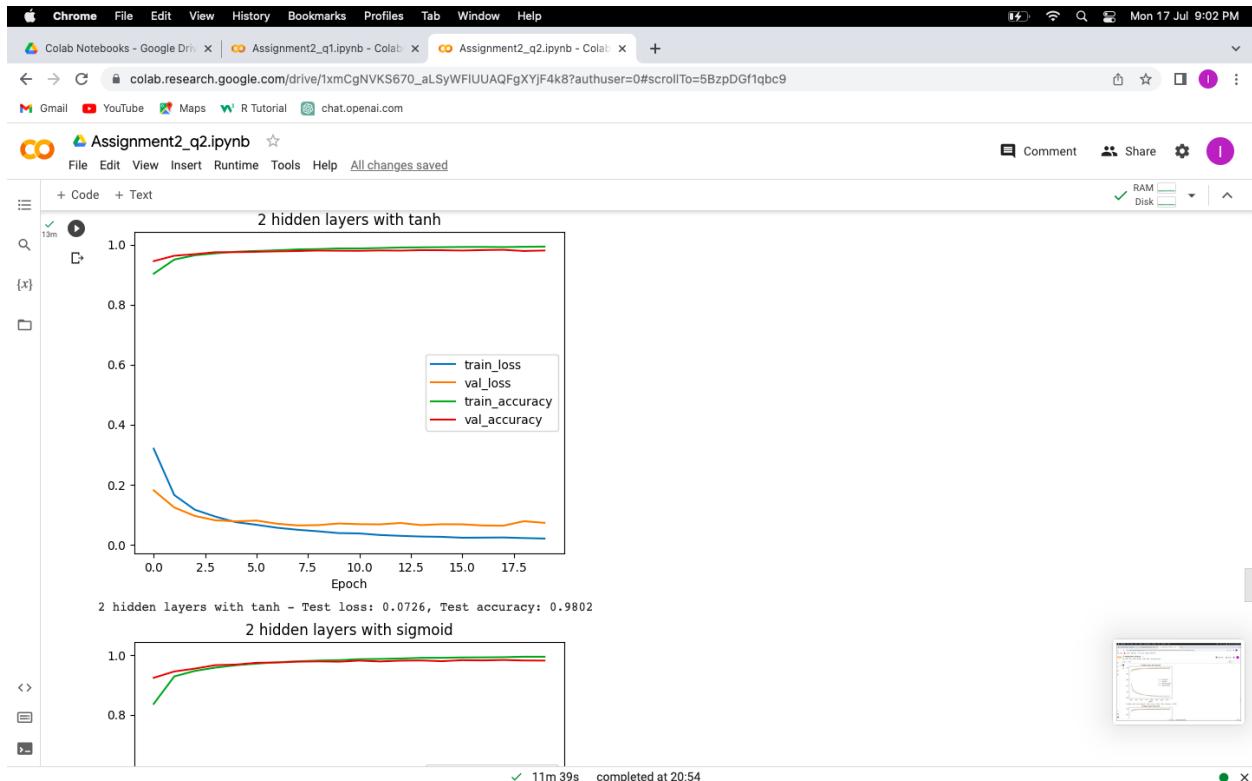
# train each model and plot loss and accuracy curves
for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                         epochs=20, batch_size=128, verbose=0)
    # plot loss and accuracy curves
```

✓ 11m 39s completed at 20:54

Output:







Chrome File Edit View History Bookmarks Profiles Tab Window Help

Colab Notebooks - Google Drive | Assignment2_q1.ipynb - Colab | Assignment2_q2.ipynb - Colab + Mon 17 Jul 9:02 PM

colab.research.google.com/drive/1xmCgNVKS670_aLSyWFIUUAQFgXYjF4k8?authuser=0#scrollTo=5BzpDGf1qbc9

Gmail YouTube Maps R Tutorial chat.openai.com

Assignment2_q2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Help

RAM Disk

2 hidden layers with sigmoid

Epoch	train_loss	val_loss	train_accuracy	val_accuracy
0.0	0.55	0.25	0.85	0.90
2.5	0.25	0.15	0.90	0.95
5.0	0.15	0.08	0.95	0.98
7.5	0.08	0.05	0.97	0.98
10.0	0.05	0.04	0.98	0.98
12.5	0.04	0.03	0.98	0.98
15.0	0.03	0.03	0.98	0.98
17.5	0.02	0.04	0.98	0.98

2 hidden layers with sigmoid - Test loss: 0.0707, Test accuracy: 0.9812

```
#2(d) 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
```

11m 39s completed at 20:54

d.) Run the same code without scaling the images and check the performance

Ans)

Code

The screenshot shows a Google Colab interface with a Jupyter notebook titled "Assignment2_q2.ipynb". The code cell contains Python code for building a neural network to classify MNIST digits. The code imports Keras, loads the MNIST dataset, converts class labels to binary matrices, creates a list of models, and defines two different model architectures: one with a tanh activation layer and another with a sigmoid activation layer. The code is annotated with comments explaining the steps.

```
#2(d)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(Dropout(0.2))
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(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Colab Notebooks - Google Drive | Assignment2_q1.ipynb - Colab | Assignment2_q2.ipynb - Colab +

colab.research.google.com/drive/1xmCgNVKS670_aLSyWFIUQAQFgXYjF4k8?authuser=0#scrollTo=5BzpDGf1qbc9

Gmail YouTube Maps R Tutorial chat.openai.com

Assignment2_q2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
# model with 2 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))

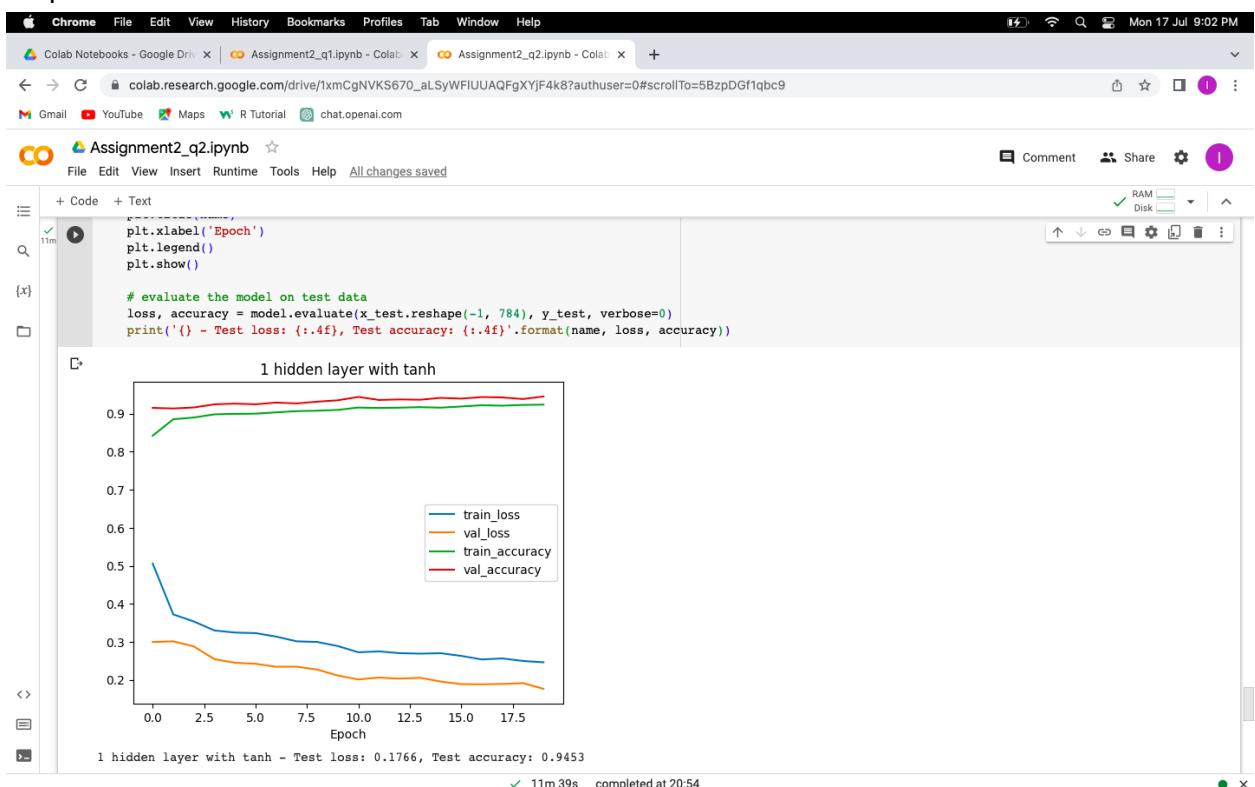
# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))

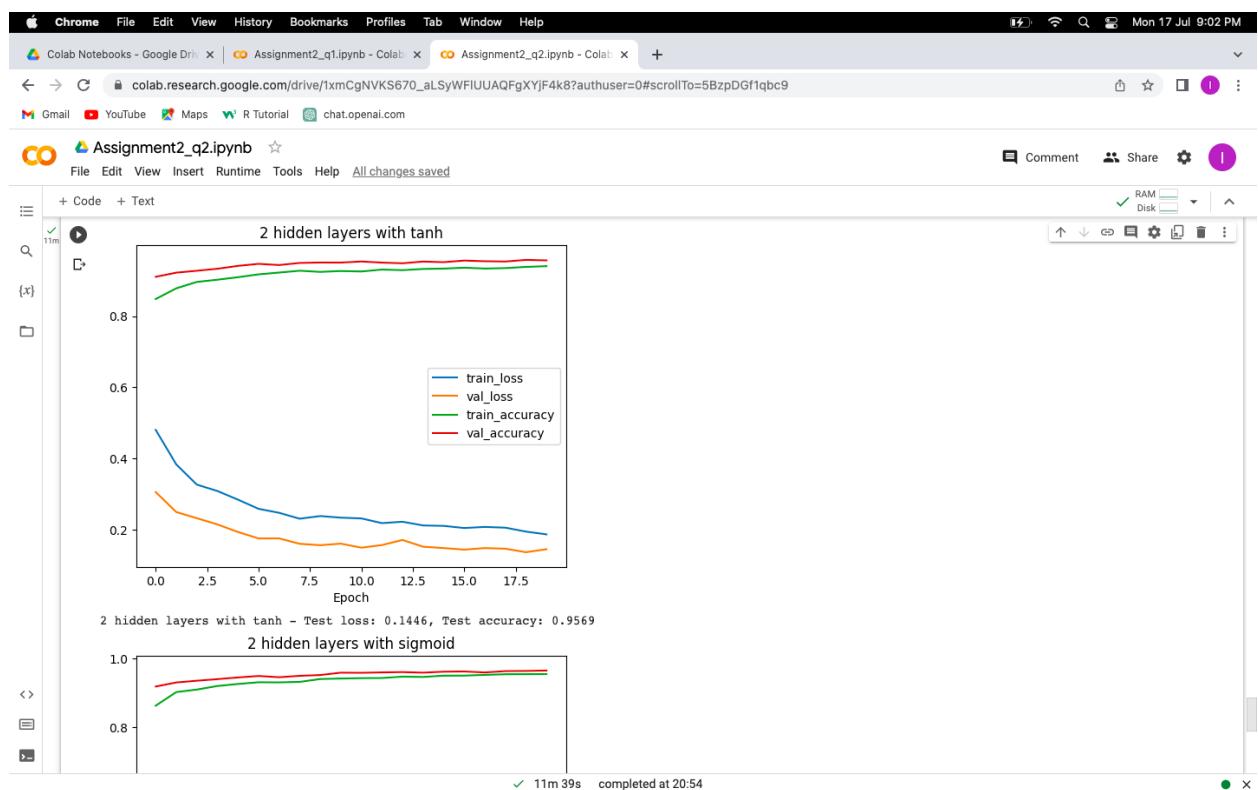
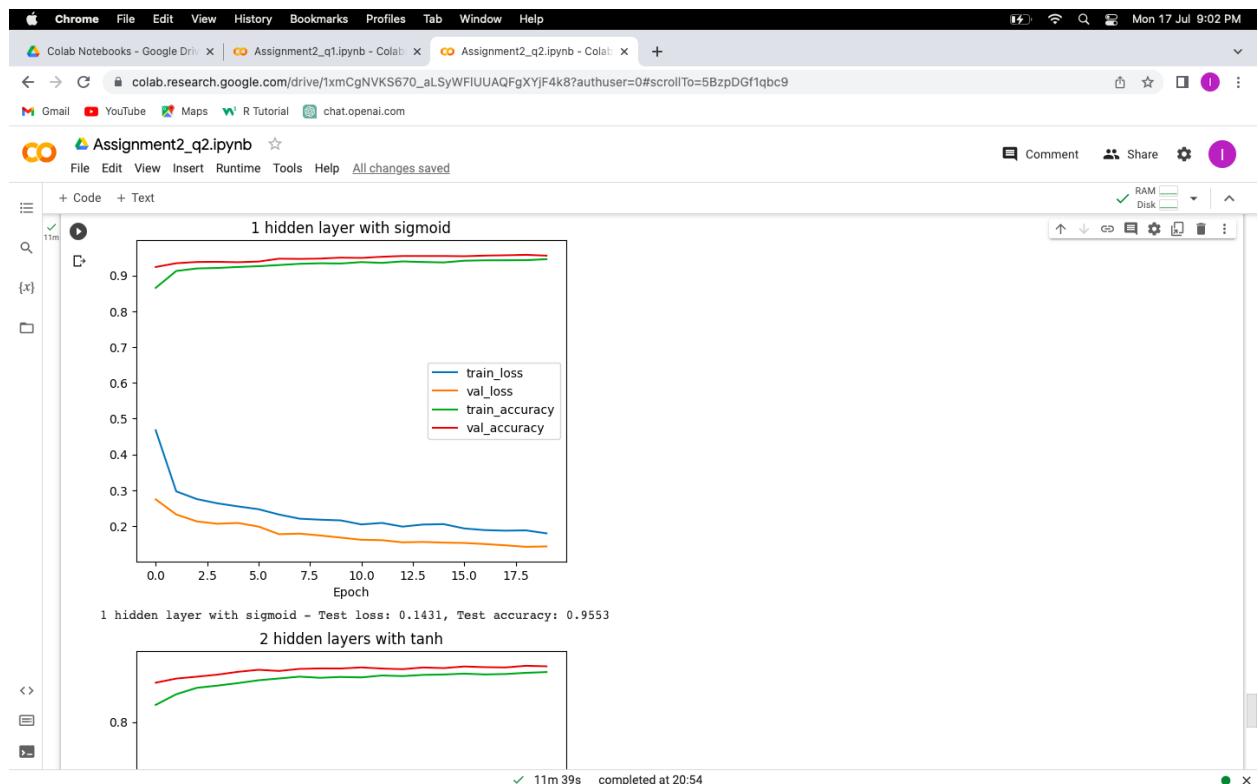
# train each model and plot loss and accuracy curves
for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                          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()

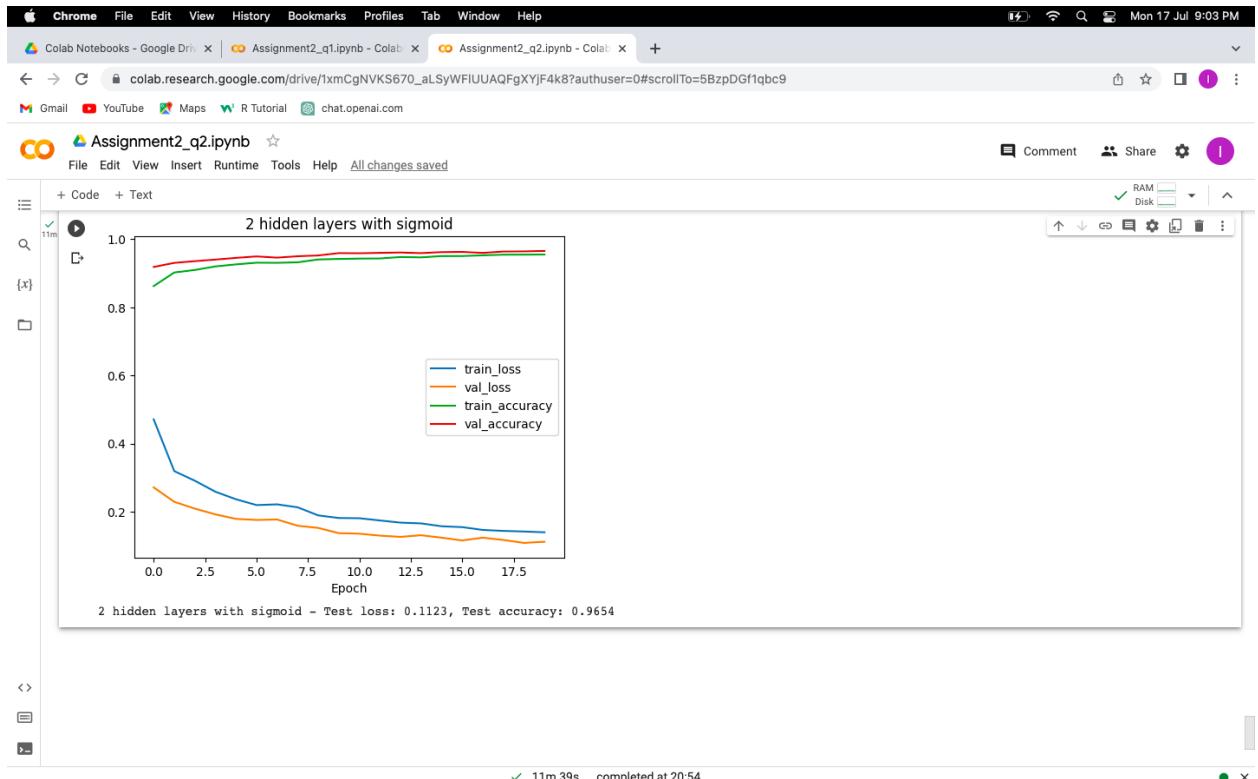
# evaluate the model on test data
loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```

11m 39s completed at 20:54

Output:







Github Link:

https://github.com/LahariKollipara/NNDL_Assignment2