

NNDL - Assignment - 6 ICP_Basics in Keras

Use Case Description:

Predicting the diabetes disease

Programming elements:

Keras Basics

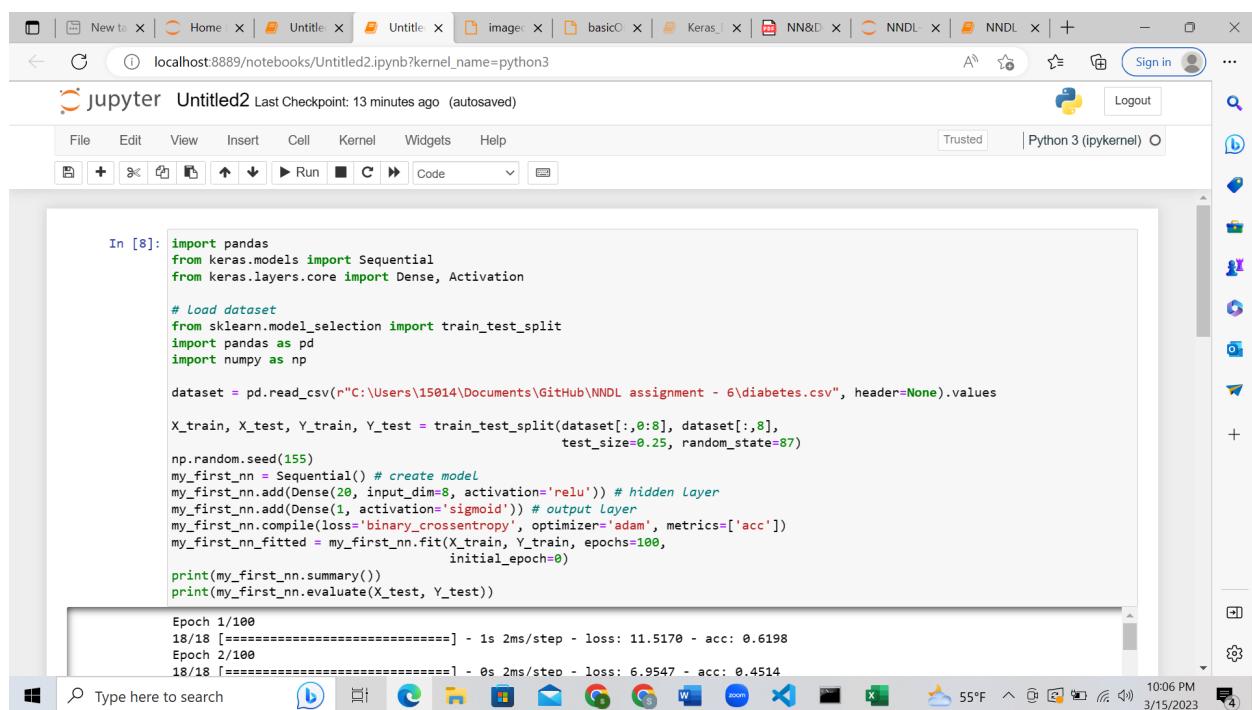
In class programming:

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 -



```
In [8]: import pandas
from keras.models import Sequential
from keras.layers.core import Dense, Activation

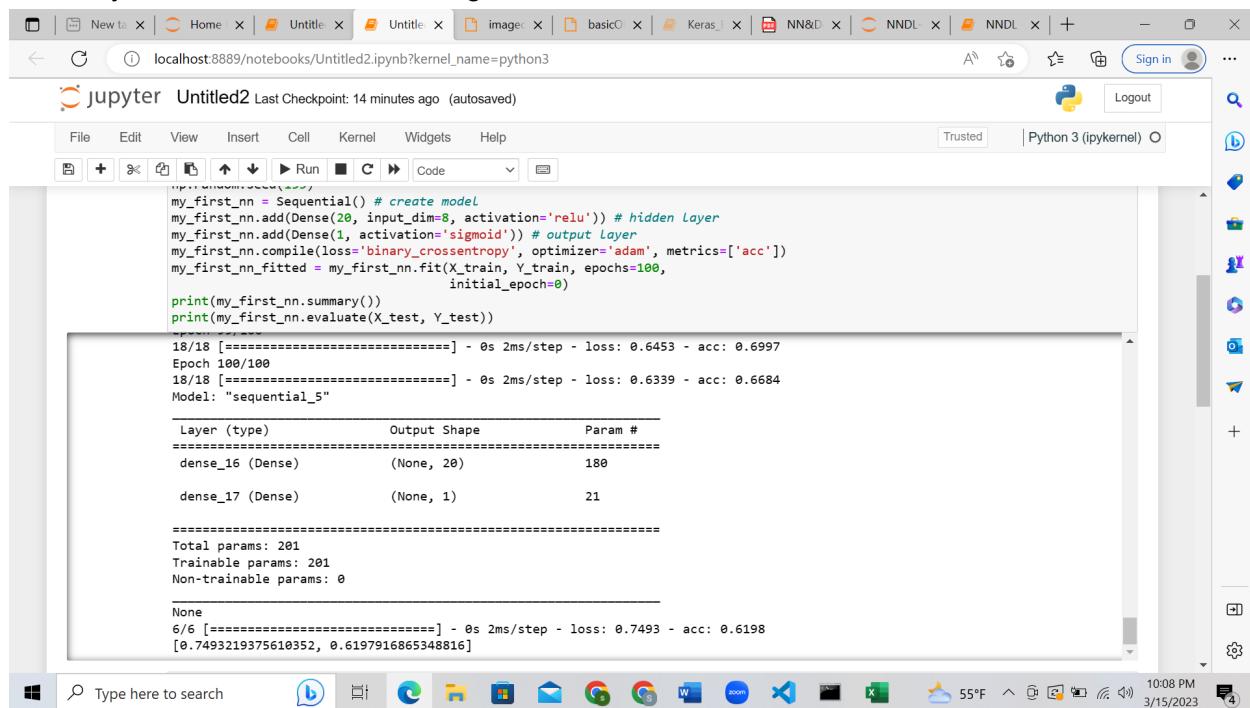
# Load dataset
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

dataset = pd.read_csv(r"C:\Users\15014\Documents\GitHub\NNDL_assignment - 6\diabetes.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))

Epoch 1/100
18/18 [=====] - 1s 2ms/step - loss: 11.5170 - acc: 0.6198
Epoch 2/100
18/18 [=====] - 0s 2ms/step - loss: 6.9547 - acc: 0.4514
```

Accuracy of the model before adding the code is -



```
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))

18/18 [=====] - 0s 2ms/step - loss: 0.6453 - acc: 0.6997
Epoch 100/100
18/18 [=====] - 0s 2ms/step - loss: 0.6339 - acc: 0.6684
Model: "sequential_5"

Layer (type)          Output Shape         Param #
=====
dense_16 (Dense)      (None, 20)          180
dense_17 (Dense)      (None, 1)           21
=====

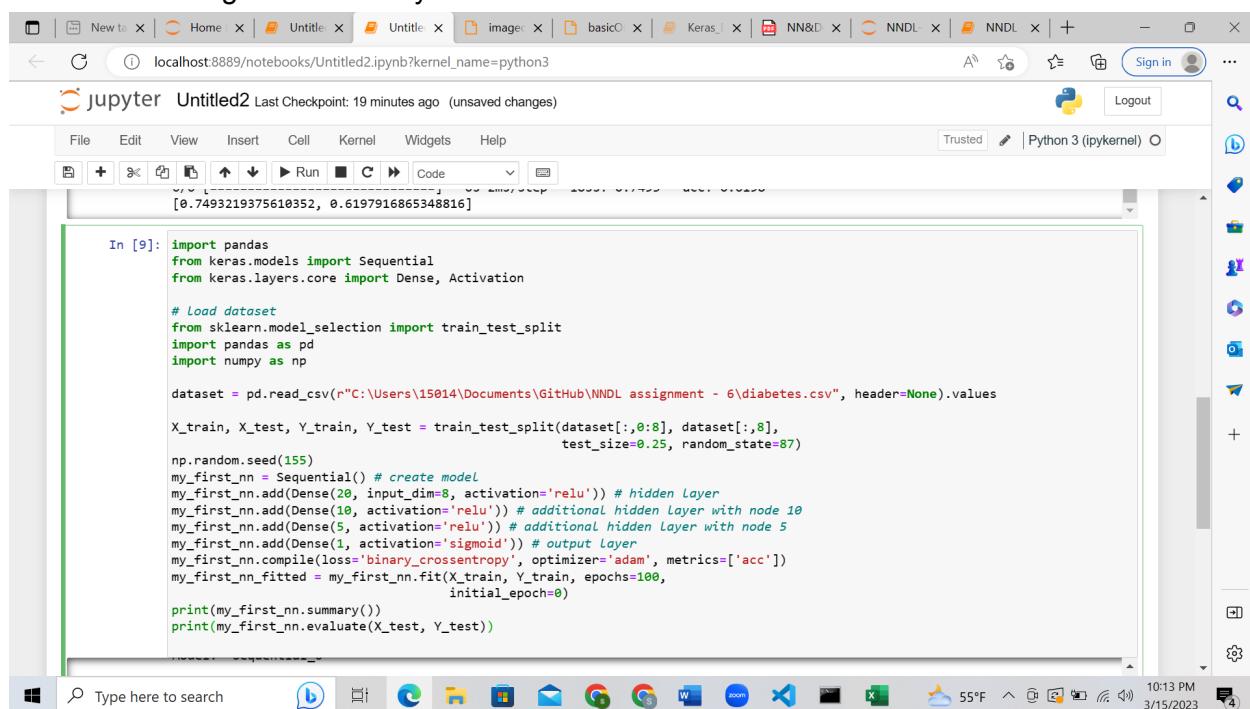
Total params: 201
Trainable params: 201
Non-trainable params: 0

None
6/6 [=====] - 0s 2ms/step - loss: 0.7493 - acc: 0.6198
[0.7493219375610352, 0.6197916865348816]
```

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 -



```
In [9]: import pandas
from keras.models import Sequential
from keras.layers.core import Dense, Activation

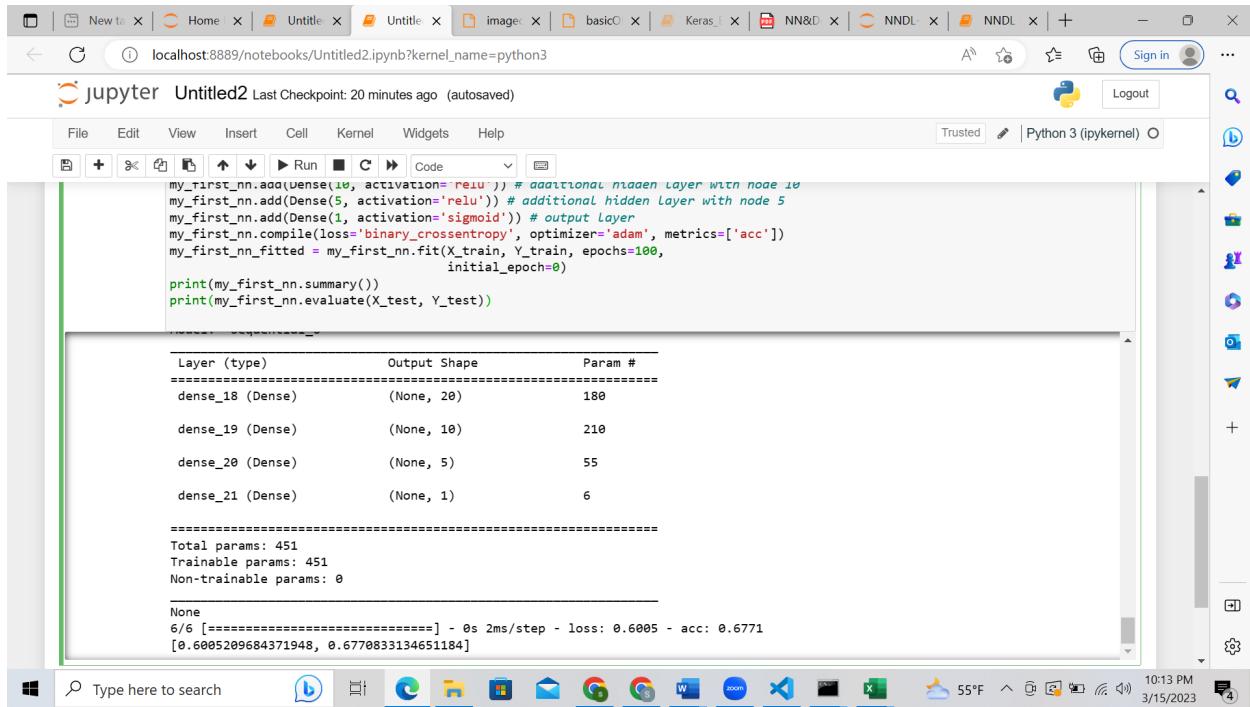
# Load dataset
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

dataset = pd.read_csv(r"C:\Users\15014\Documents\GitHub\NNDL assignment - 6\diabetes.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))

[0.7493219375610352, 0.6197916865348816]
```

Accuracy of the model after adding the additional dense layers is



```
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))

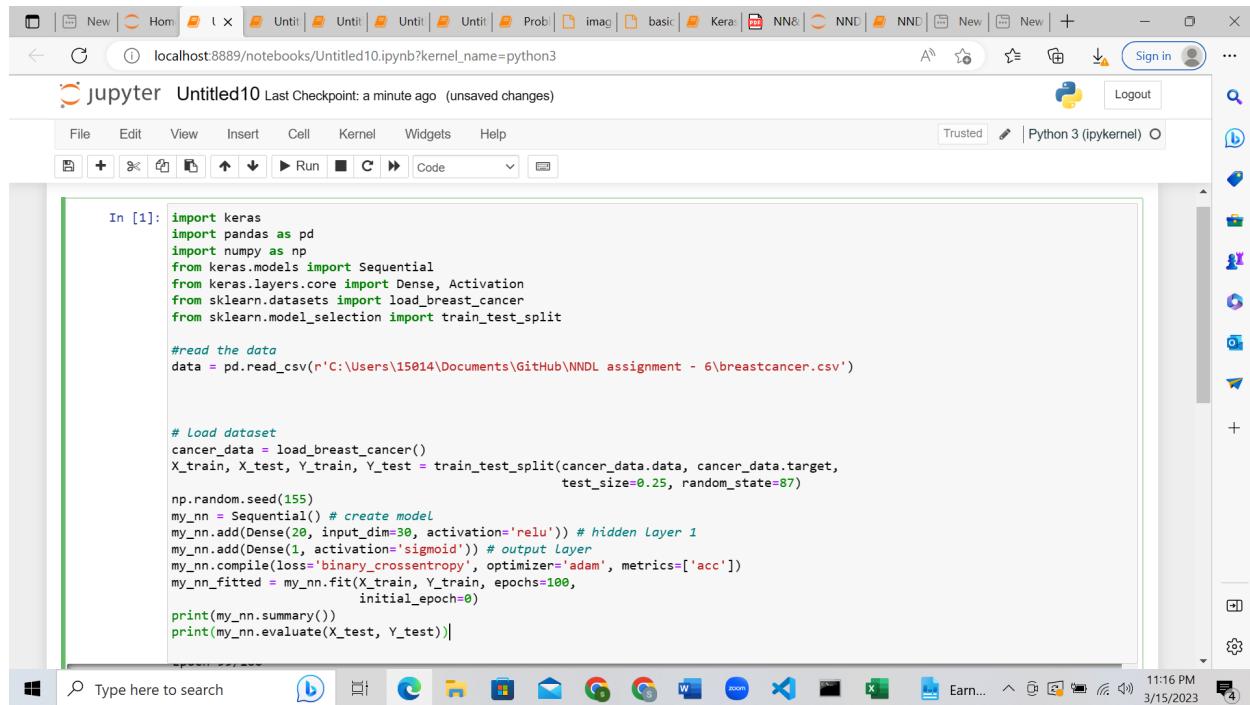
Layer (type)          Output Shape         Param #
=====
dense_18 (Dense)      (None, 20)          180
dense_19 (Dense)      (None, 10)          210
dense_20 (Dense)      (None, 5)           55
dense_21 (Dense)      (None, 1)           6
=====
Total params: 451
Trainable params: 451
Non-trainable params: 0
=====
None
6/6 [=====] - 0s 2ms/step - loss: 0.6005 - acc: 0.6771
[0.6005209684371948, 0.6770833134651184]
```

Accuracy of the model increased from 0.6198 to 0.6771 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



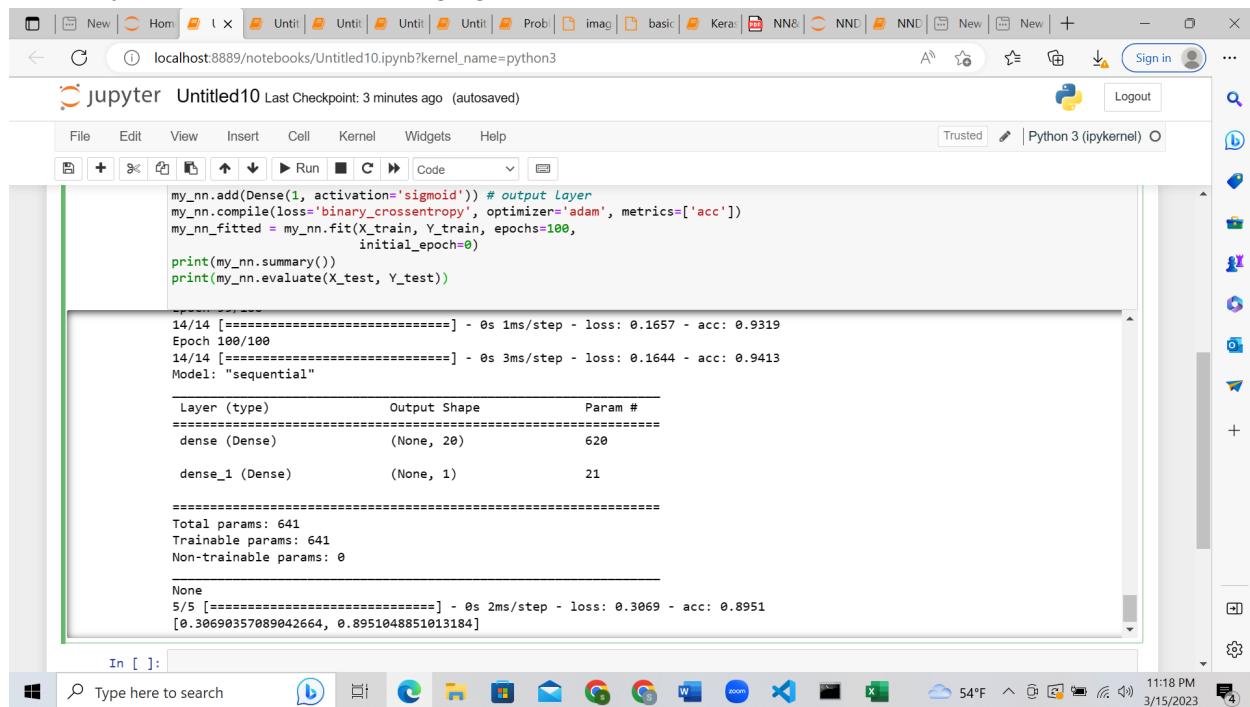
In [1]:

```
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

#read the data
data = pd.read_csv(r'C:\Users\15014\Documents\GitHub\NNDL assignment - 6\breastcancer.csv')

# Load dataset
cancer_data = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                    test_size=0.25, random_state=87)
np.random.seed(1555)
my_nn = Sequential() # create model
my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden Layer 1
my_nn.add(Dense(1, activation='sigmoid')) # output Layer
my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                         initial_epoch=0)
print(my_nn.summary())
print(my_nn.evaluate(X_test, Y_test))
```

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



```
my_nn.add(Dense(1, activation='sigmoid')) # output Layer
my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                         initial_epoch=0)
print(my_nn.summary())
print(my_nn.evaluate(X_test, Y_test))

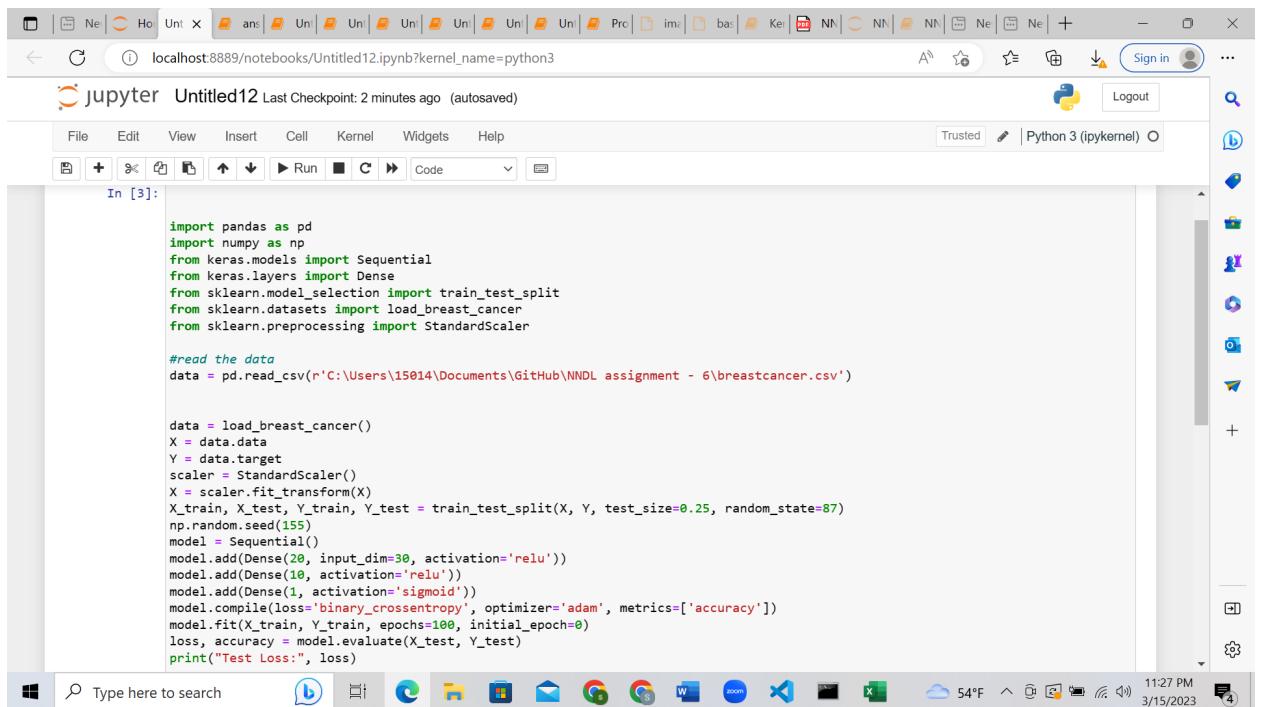
14/14 [=====] - 0s 1ms/step - loss: 0.1657 - acc: 0.9319
Epoch 100/100
14/14 [=====] - 0s 3ms/step - loss: 0.1644 - acc: 0.9413
Model: "sequential"

Layer (type)          Output Shape         Param #
=====
dense (Dense)         (None, 20)          620
dense_1 (Dense)       (None, 1)           21
=====
Total params: 641
Trainable params: 641
Non-trainable params: 0
=====
None
5/5 [=====] - 0s 2ms/step - loss: 0.3069 - acc: 0.8951
[0.30690357089042664, 0.8951048851013184]
```

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)

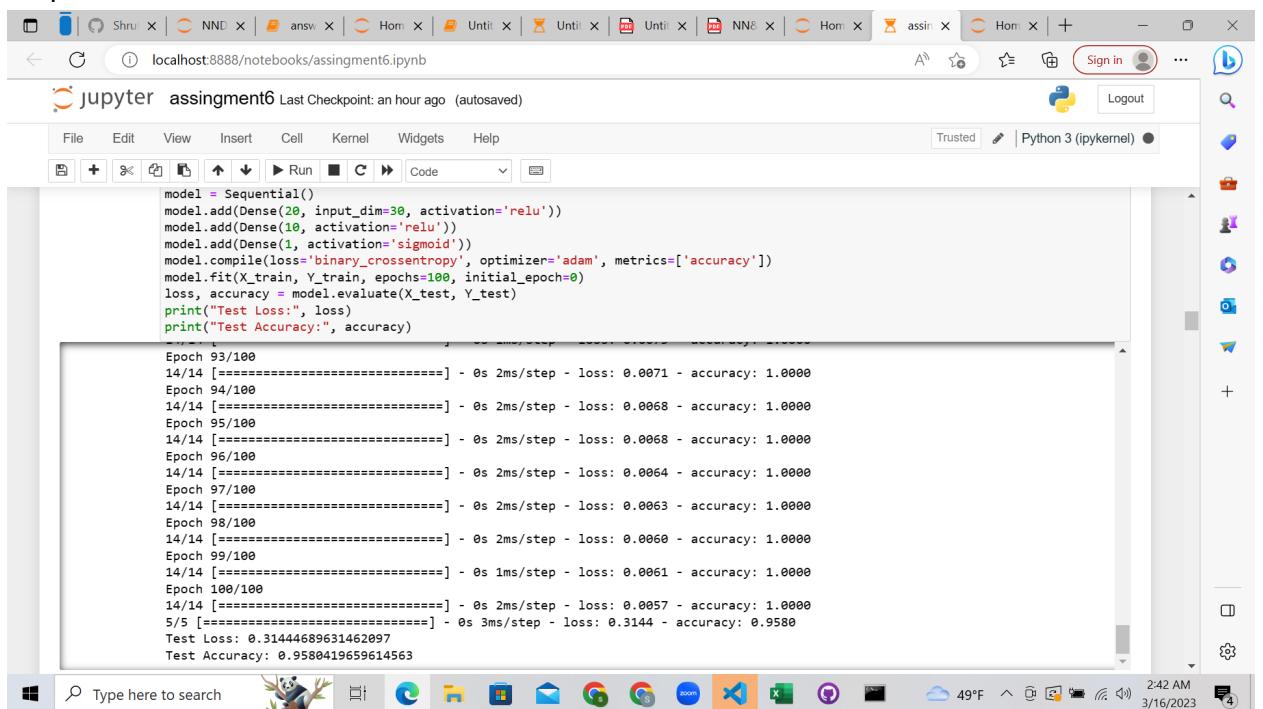


```
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler

#read the data
data = pd.read_csv(r'C:\Users\15014\Documents\GitHub\NNDL assignment - 6\breastcancer.csv')

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



```
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)

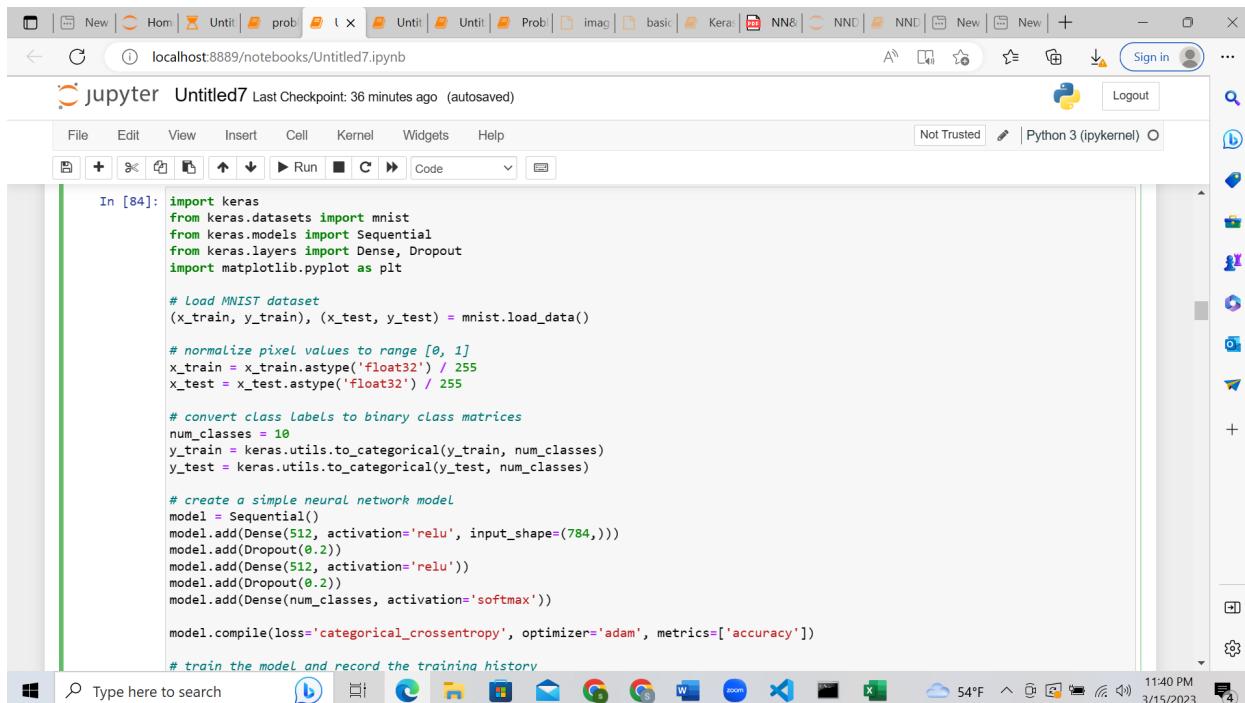
Epoch 93/100
14/14 [=====] - 0s 2ms/step - loss: 0.0071 - accuracy: 1.0000
Epoch 94/100
14/14 [=====] - 0s 2ms/step - loss: 0.0068 - accuracy: 1.0000
Epoch 95/100
14/14 [=====] - 0s 2ms/step - loss: 0.0068 - accuracy: 1.0000
Epoch 96/100
14/14 [=====] - 0s 2ms/step - loss: 0.0064 - accuracy: 1.0000
Epoch 97/100
14/14 [=====] - 0s 2ms/step - loss: 0.0063 - accuracy: 1.0000
Epoch 98/100
14/14 [=====] - 0s 2ms/step - loss: 0.0060 - accuracy: 1.0000
Epoch 99/100
14/14 [=====] - 0s 1ms/step - loss: 0.0061 - accuracy: 1.0000
Epoch 100/100
14/14 [=====] - 0s 2ms/step - loss: 0.0057 - accuracy: 1.0000
5/5 [=====] - 0s 3ms/step - loss: 0.3144 - accuracy: 0.9580
Test Loss: 0.3144468963146209
Test Accuracy: 0.9580419659614563
```

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

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

Ans)

Code for the above code is



```
In [84]: 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()

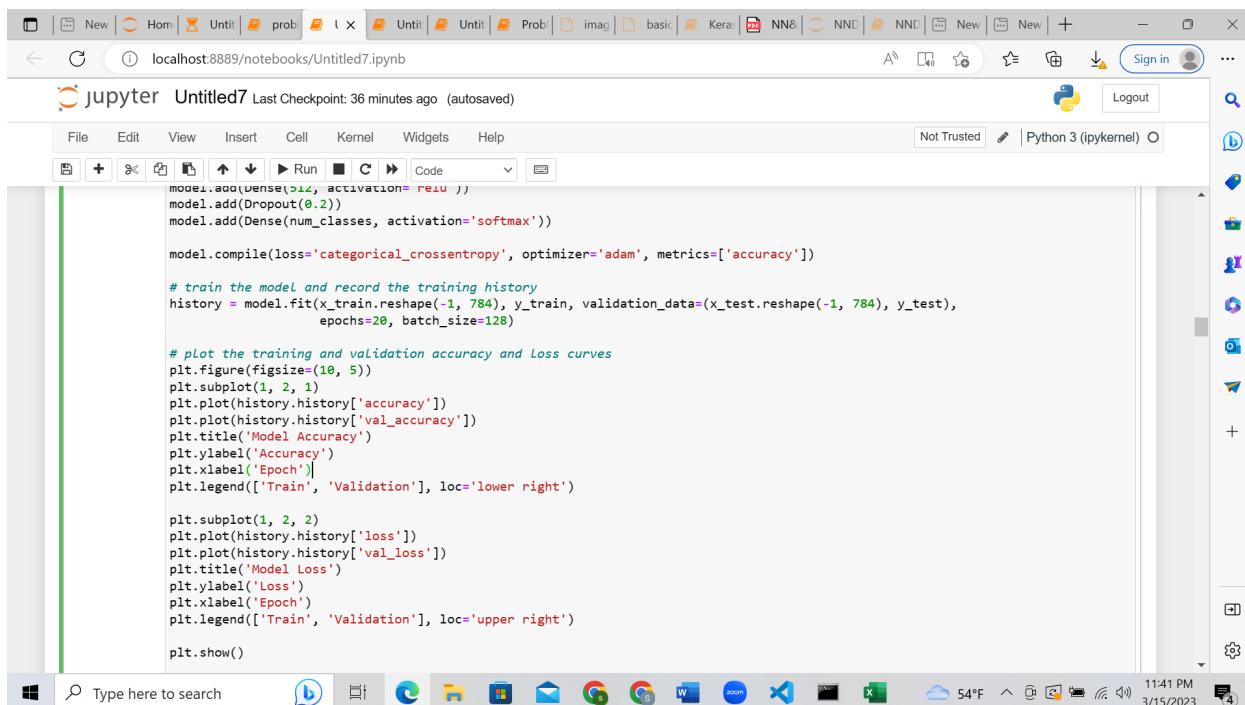
# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# 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
```



```
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
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                     epochs=20, batch_size=128)

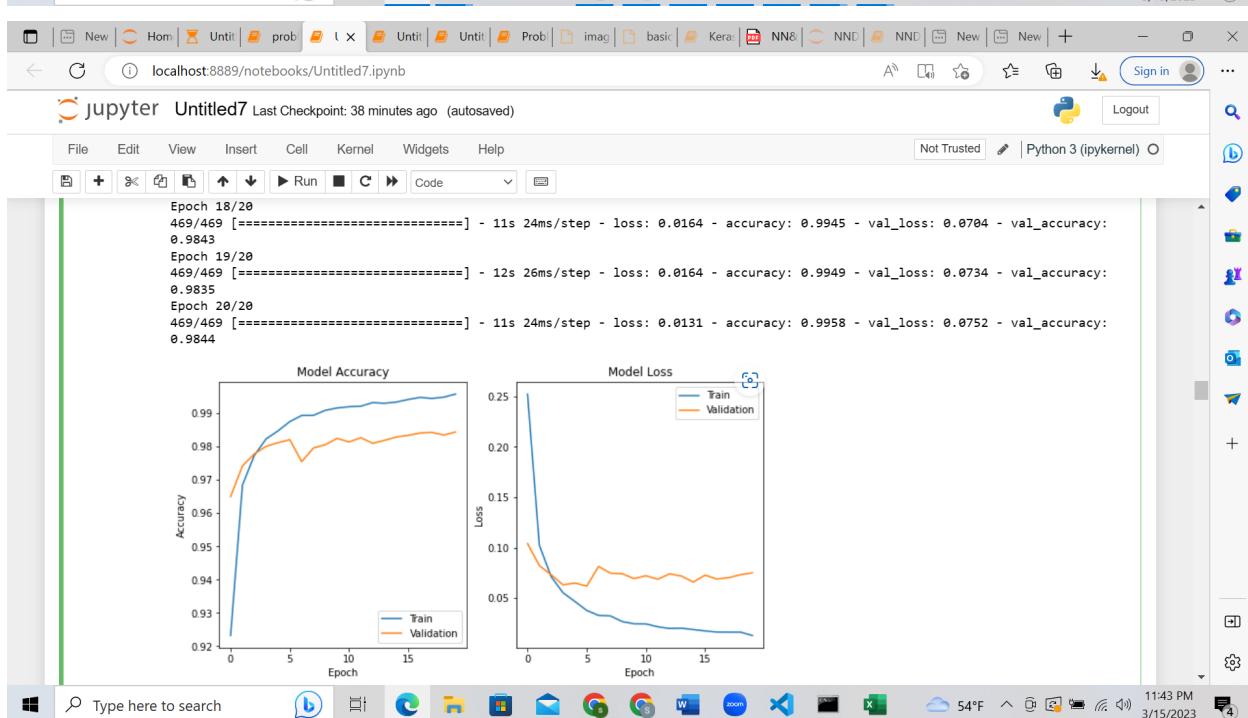
# plot the training and validation accuracy and loss curves
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='lower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

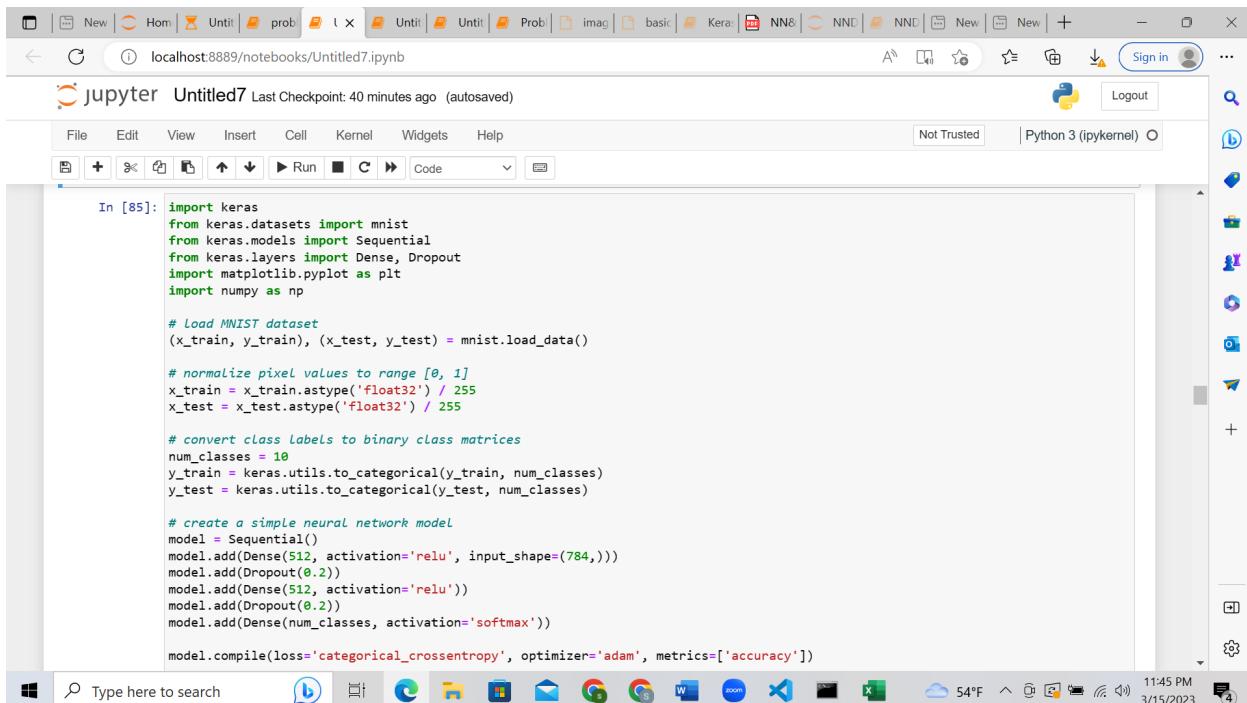
plt.show()
```

Output of the code

```
469/469 [=====] - 16s 27ms/step - loss: 0.2524 - accuracy: 0.9232 - val_loss: 0.1042 - val_accuracy: 0.9650
Epoch 2/20
469/469 [=====] - 17s 36ms/step - loss: 0.1024 - accuracy: 0.9684 - val_loss: 0.0823 - val_accuracy: 0.9742
Epoch 3/20
469/469 [=====] - 14s 29ms/step - loss: 0.0713 - accuracy: 0.9773 - val_loss: 0.0733 - val_accuracy: 0.9778
Epoch 4/20
469/469 [=====] - 13s 28ms/step - loss: 0.0554 - accuracy: 0.9823 - val_loss: 0.0632 - val_accuracy: 0.9801
Epoch 5/20
469/469 [=====] - 12s 25ms/step - loss: 0.0468 - accuracy: 0.9847 - val_loss: 0.0651 - val_accuracy: 0.9812
Epoch 6/20
469/469 [=====] - 12s 25ms/step - loss: 0.0379 - accuracy: 0.9875 - val_loss: 0.0620 - val_accuracy: 0.9821
Epoch 7/20
469/469 [=====] - 13s 28ms/step - loss: 0.0330 - accuracy: 0.9894 - val_loss: 0.0815 - val_accuracy: 0.9755
Epoch 8/20
469/469 [=====] - 12s 25ms/step - loss: 0.0325 - accuracy: 0.9894 - val_loss: 0.0749 - val_accuracy: 0.9796
Epoch 9/20
469/469 [=====] - 15s 31ms/step - loss: 0.0269 - accuracy: 0.9909 - val_loss: 0.0743 - val_accuracy: 0.9806
Epoch 10/20
469/469 [=====] - 12s 27ms/step - loss: 0.0247 - accuracy: 0.9916 - val_loss: 0.0694 - val_accuracy: 0.9825
```



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.



```
In [85]: 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

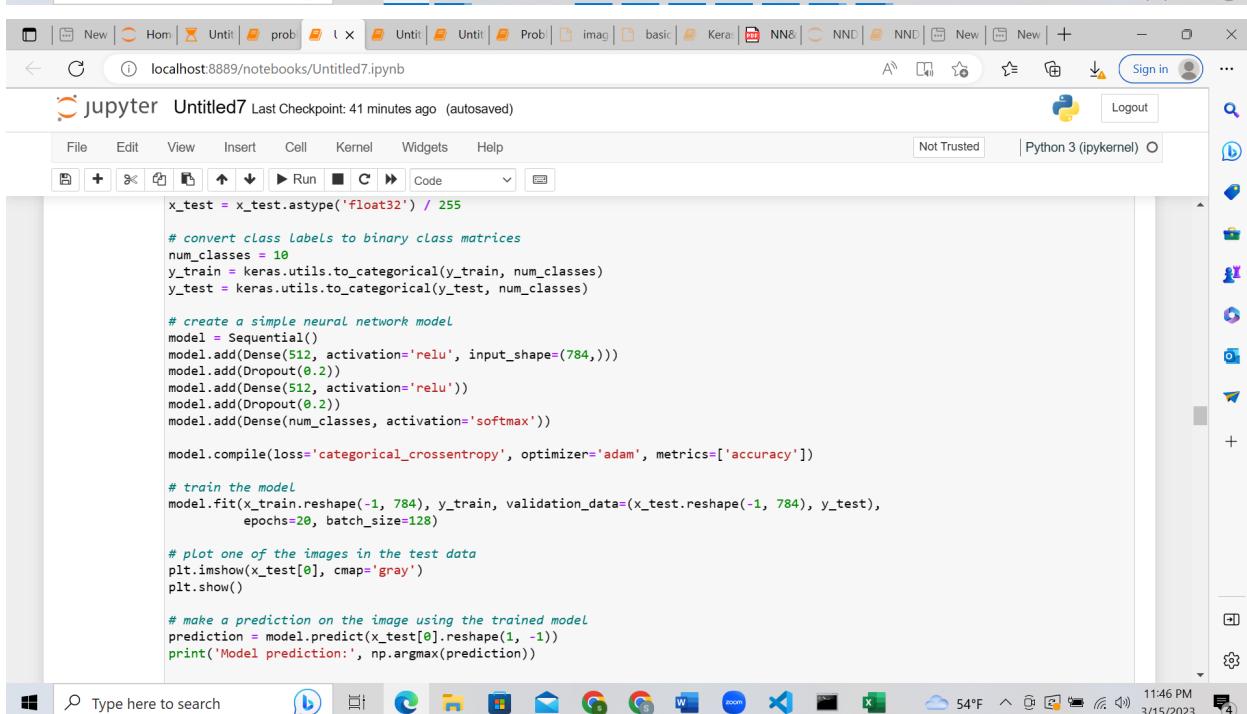
# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# 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'])
```



```
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'])

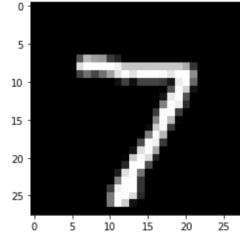
# train the model
model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
          epochs=20, batch_size=128)

# 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))
```

```
Epoch 1/20
469/469 [=====] - 12s 22ms/step - loss: 0.2488 - accuracy: 0.9253 - val_loss: 0.1118 - val_accuracy: 0.9652
Epoch 2/20
469/469 [=====] - 11s 24ms/step - loss: 0.1016 - accuracy: 0.9684 - val_loss: 0.0742 - val_accuracy: 0.9769
Epoch 3/20
469/469 [=====] - 15s 31ms/step - loss: 0.0713 - accuracy: 0.9779 - val_loss: 0.0695 - val_accuracy: 0.9784
Epoch 4/20
469/469 [=====] - 14s 30ms/step - loss: 0.0561 - accuracy: 0.9819 - val_loss: 0.0702 - val_accuracy: 0.9779
Epoch 5/20
469/469 [=====] - 11s 24ms/step - loss: 0.0455 - accuracy: 0.9855 - val_loss: 0.0615 - val_accuracy: 0.9822
Epoch 6/20
469/469 [=====] - 11s 24ms/step - loss: 0.0381 - accuracy: 0.9873 - val_loss: 0.0648 - val_accuracy: 0.9818
Epoch 7/20
469/469 [=====] - 11s 24ms/step - loss: 0.0337 - accuracy: 0.9891 - val_loss: 0.0732 - val_accuracy: 0.9810
Epoch 8/20
469/469 [=====] - 12s 25ms/step - loss: 0.0296 - accuracy: 0.9905 - val_loss: 0.0675 - val_accuracy: 0.9811
Epoch 9/20
469/469 [=====] - 12s 25ms/step - loss: 0.0279 - accuracy: 0.9905 - val_loss: 0.0756 - val_accuracy: 0.9799
Epoch 10/20
469/469 [=====] - 12s 26ms/step - loss: 0.0248 - accuracy: 0.9915 - val_loss: 0.0804 - val_accuracy: 0.9823
```

```
0.9823
Epoch 18/20
469/469 [=====] - 10s 22ms/step - loss: 0.0162 - accuracy: 0.9948 - val_loss: 0.0800 - val_accuracy: 0.9822
Epoch 19/20
469/469 [=====] - 11s 24ms/step - loss: 0.0145 - accuracy: 0.9951 - val_loss: 0.0873 - val_accuracy: 0.9820
Epoch 20/20
469/469 [=====] - 11s 24ms/step - loss: 0.0140 - accuracy: 0.9957 - val_loss: 0.0807 - val_accuracy: 0.9841
```

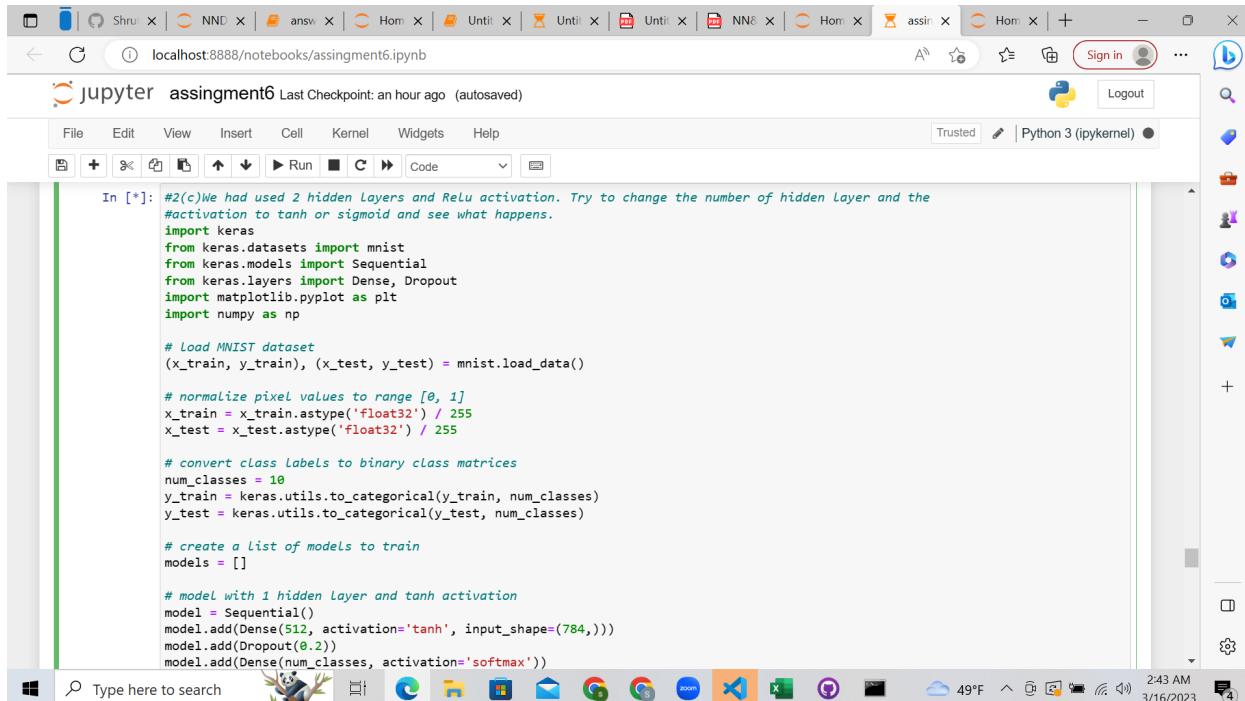


```
1/1 [=====] - 0s 120ms/step
Model prediction: 7
```

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



```
In [*]: #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.
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

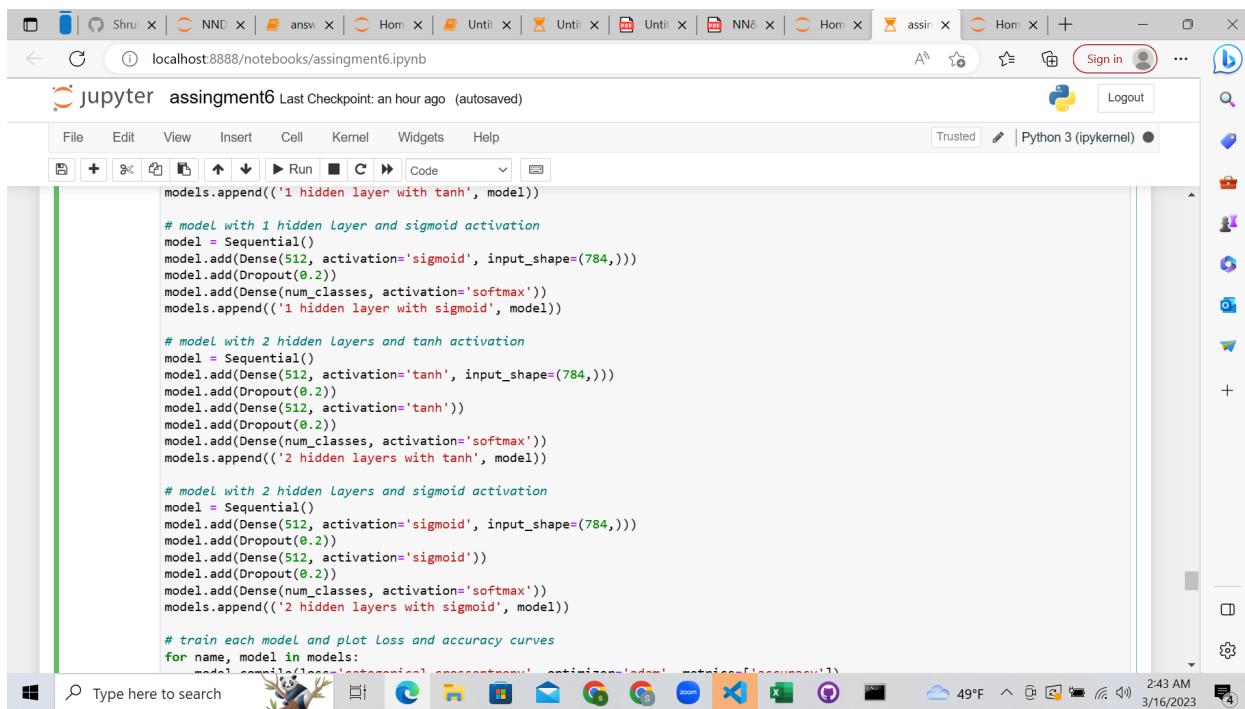
# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# 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('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'])
```

jupyter assingment6 Last Checkpoint: an hour ago (autosaved)

```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) ●
```

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))

2:44 AM 3/16/2023

Output

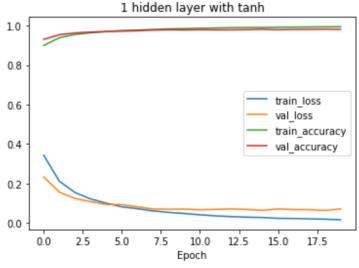
jupyter answers Last Checkpoint: 3 hours ago (autosaved)

```
File Edit View Insert Cell Kernel Widgets Help Not Connected Not Trusted Python 3 (ipykernel) ●
```

plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.title(name)
plt.xlabel('Epoch')
plt.legend()
plt.show()

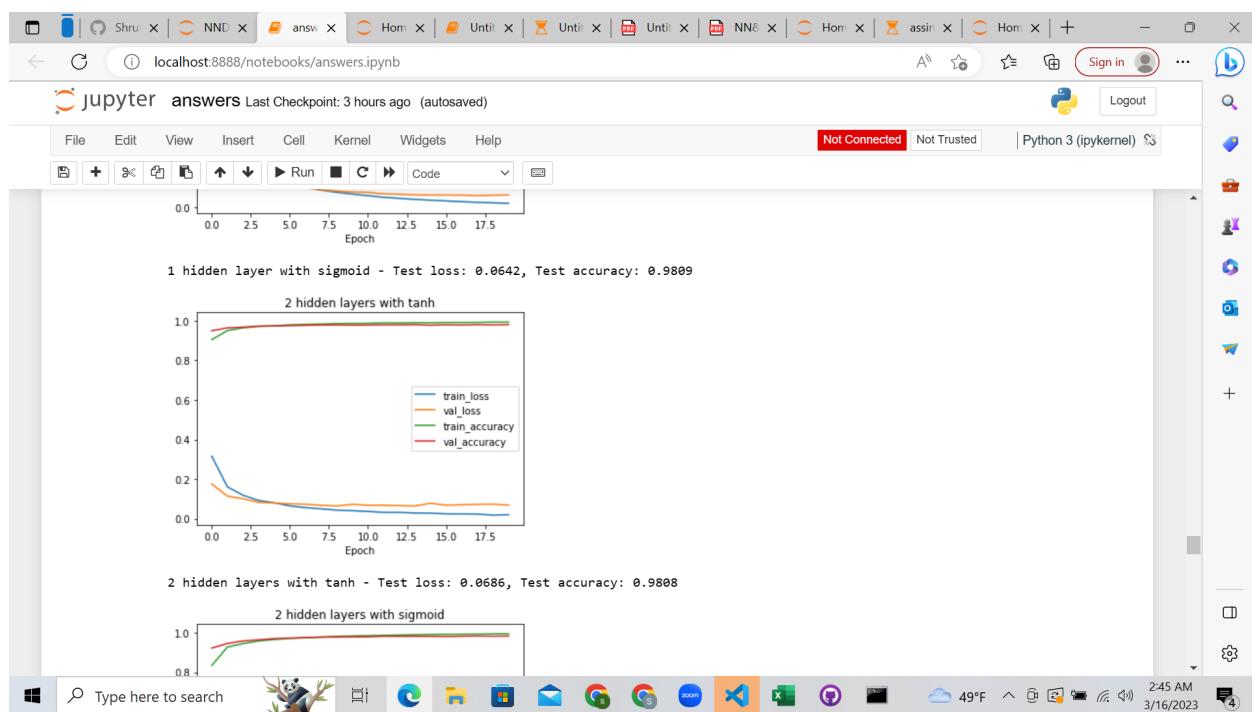
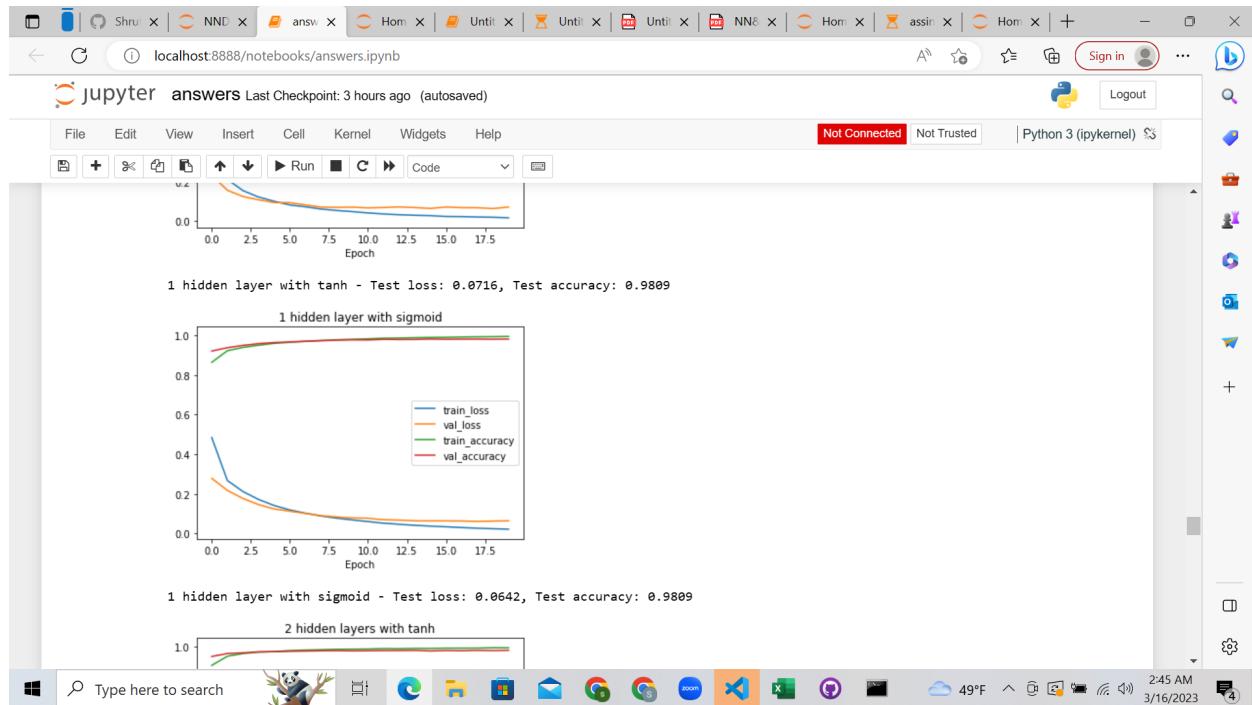
evaluate the model on test data
loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))

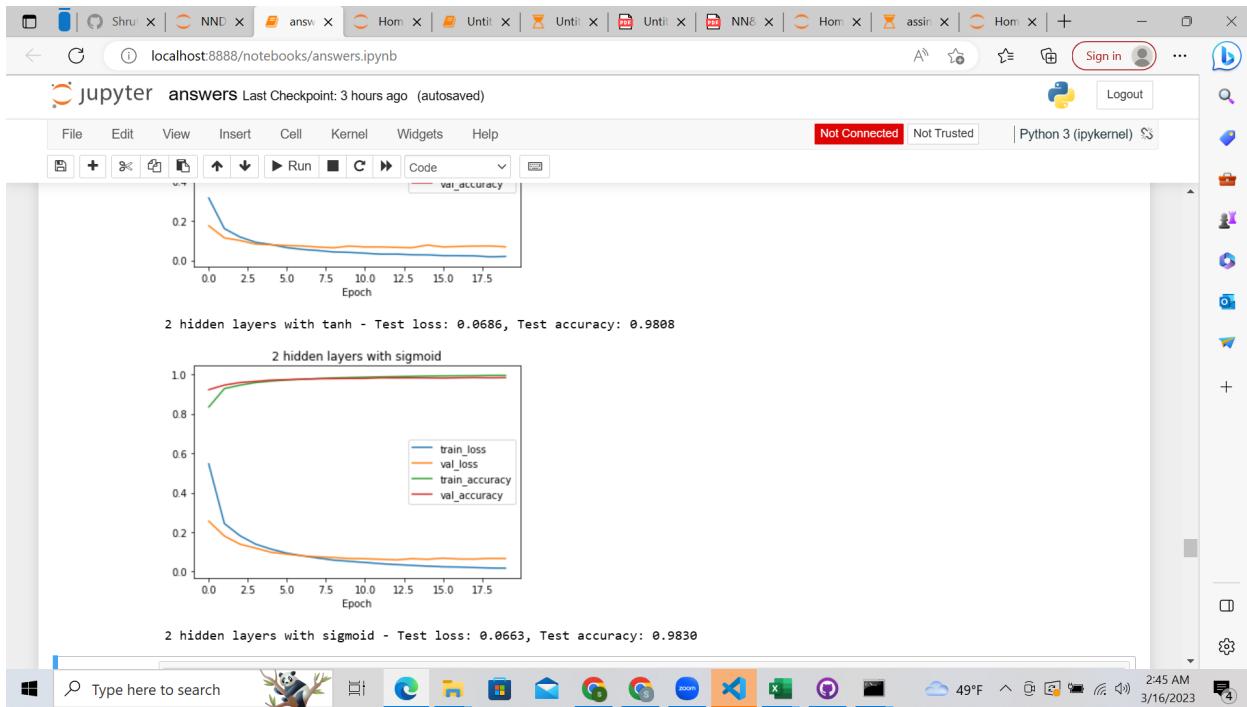
1 hidden layer with tanh



1 hidden layer with tanh - Test loss: 0.0716, Test accuracy: 0.9809

2:45 AM 3/16/2023





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

Code

```
In [*]: #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()
```

jupyter assingment6 Last Checkpoint: an hour ago (autosaved)

```
# 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),
```

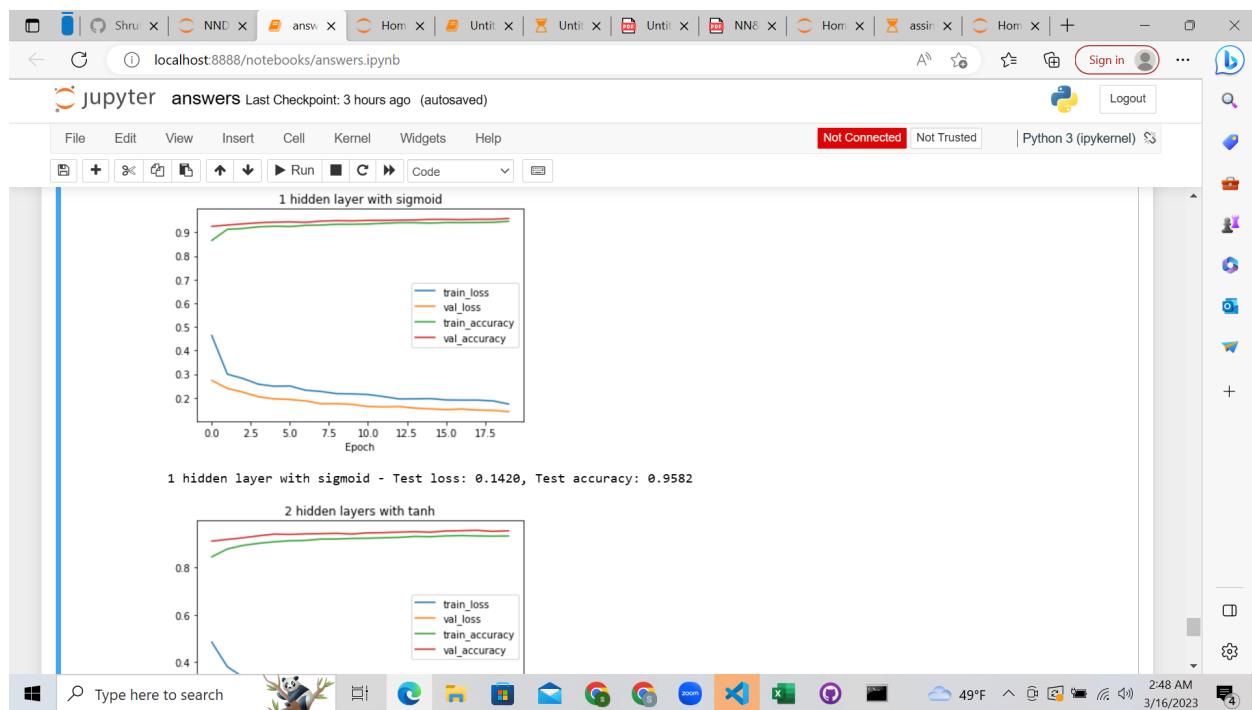
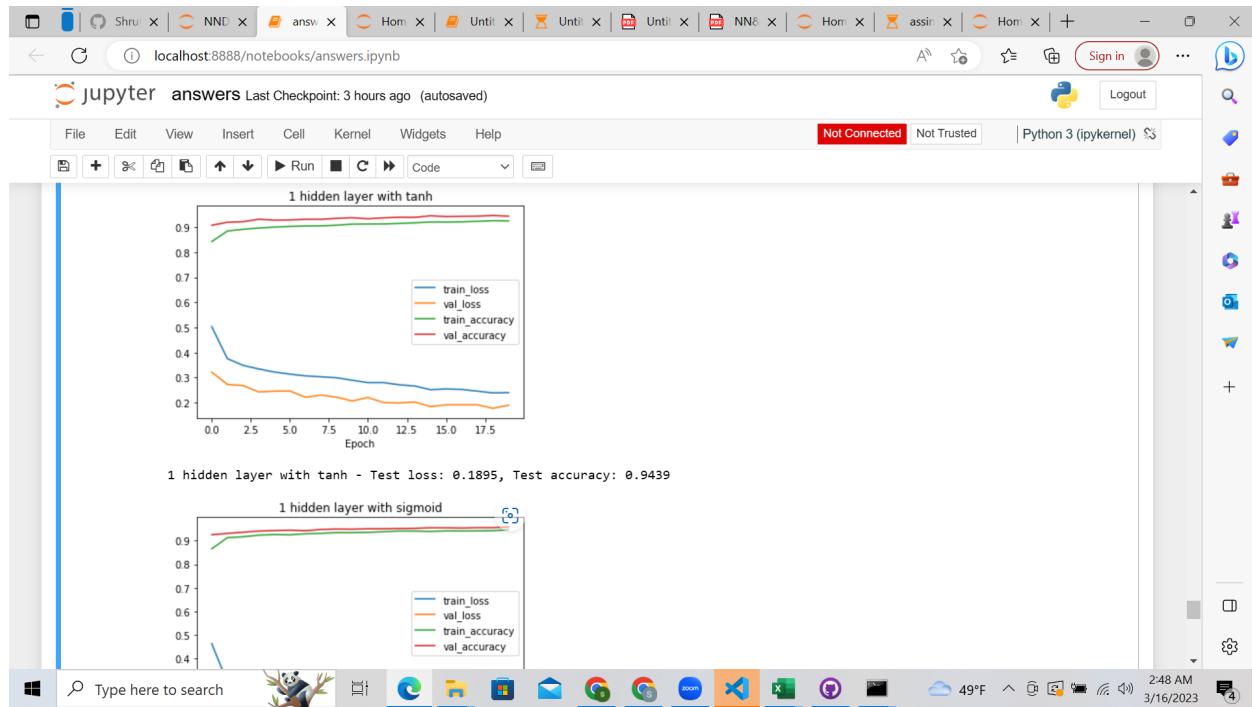
jupyter assingment6 Last Checkpoint: an hour ago (autosaved)

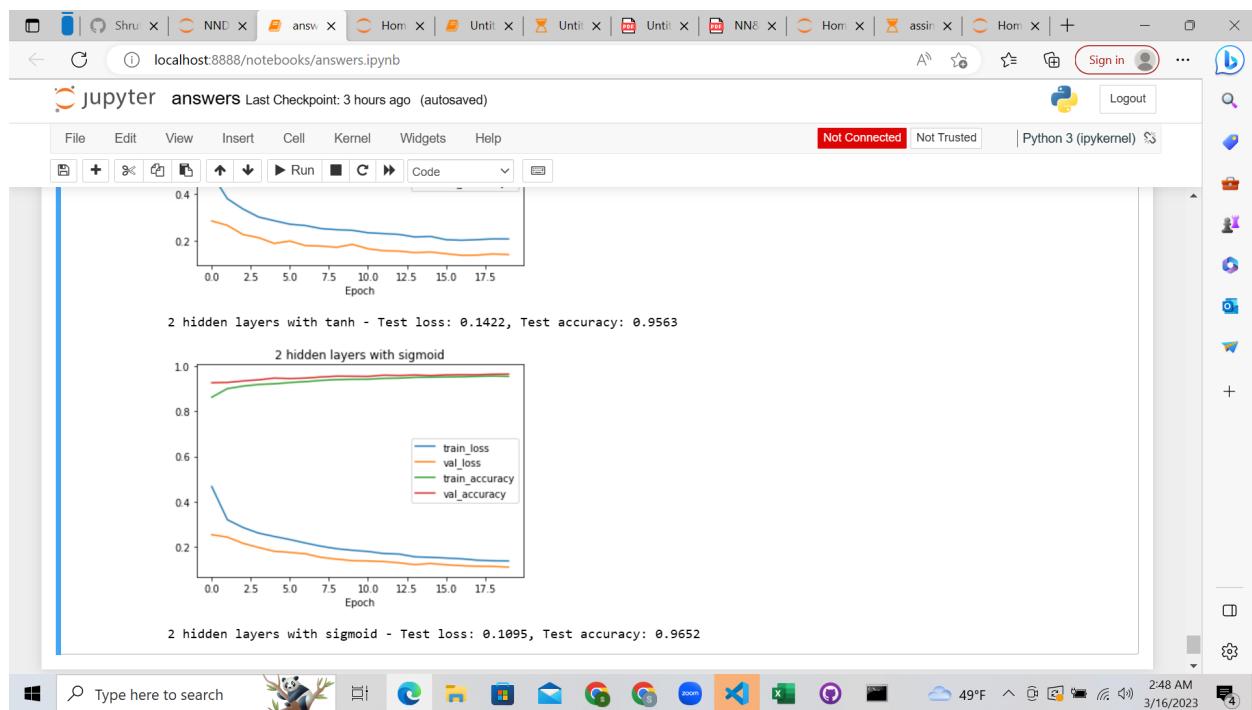
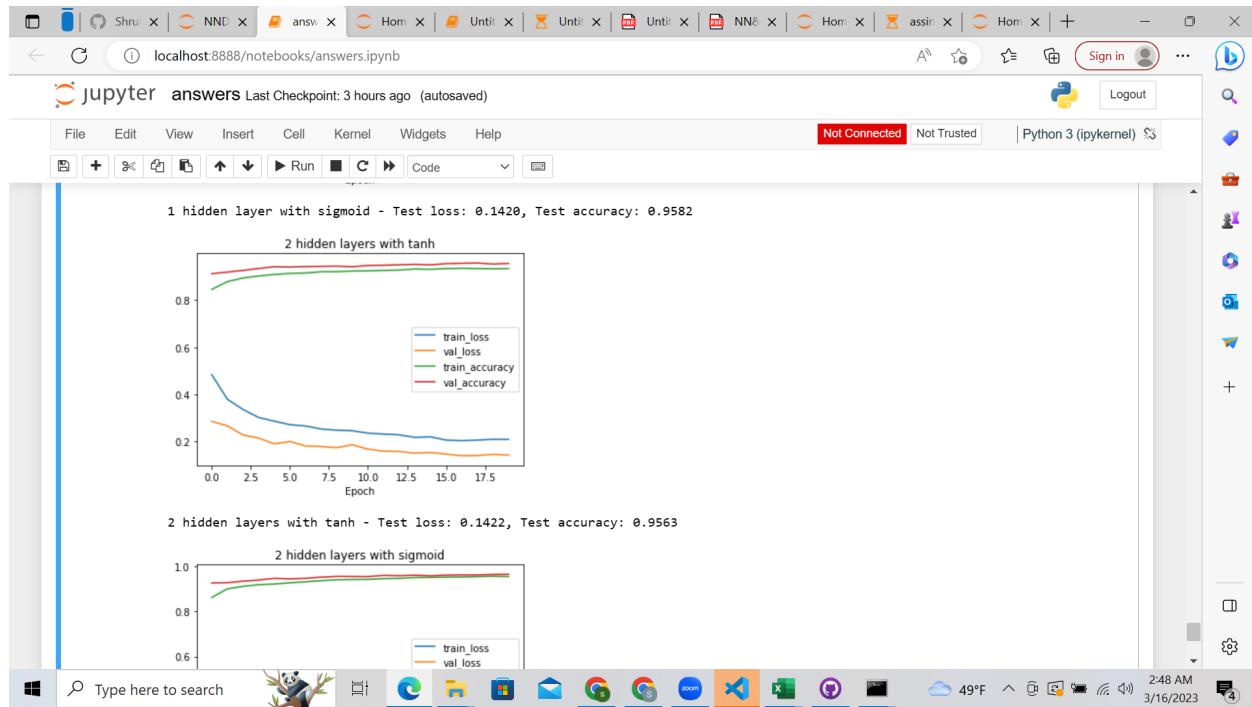
```
# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append('2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

Output





Git repo link : <https://github.com/ShruthiVallapReddy/NNDL---Assignment---6.git>