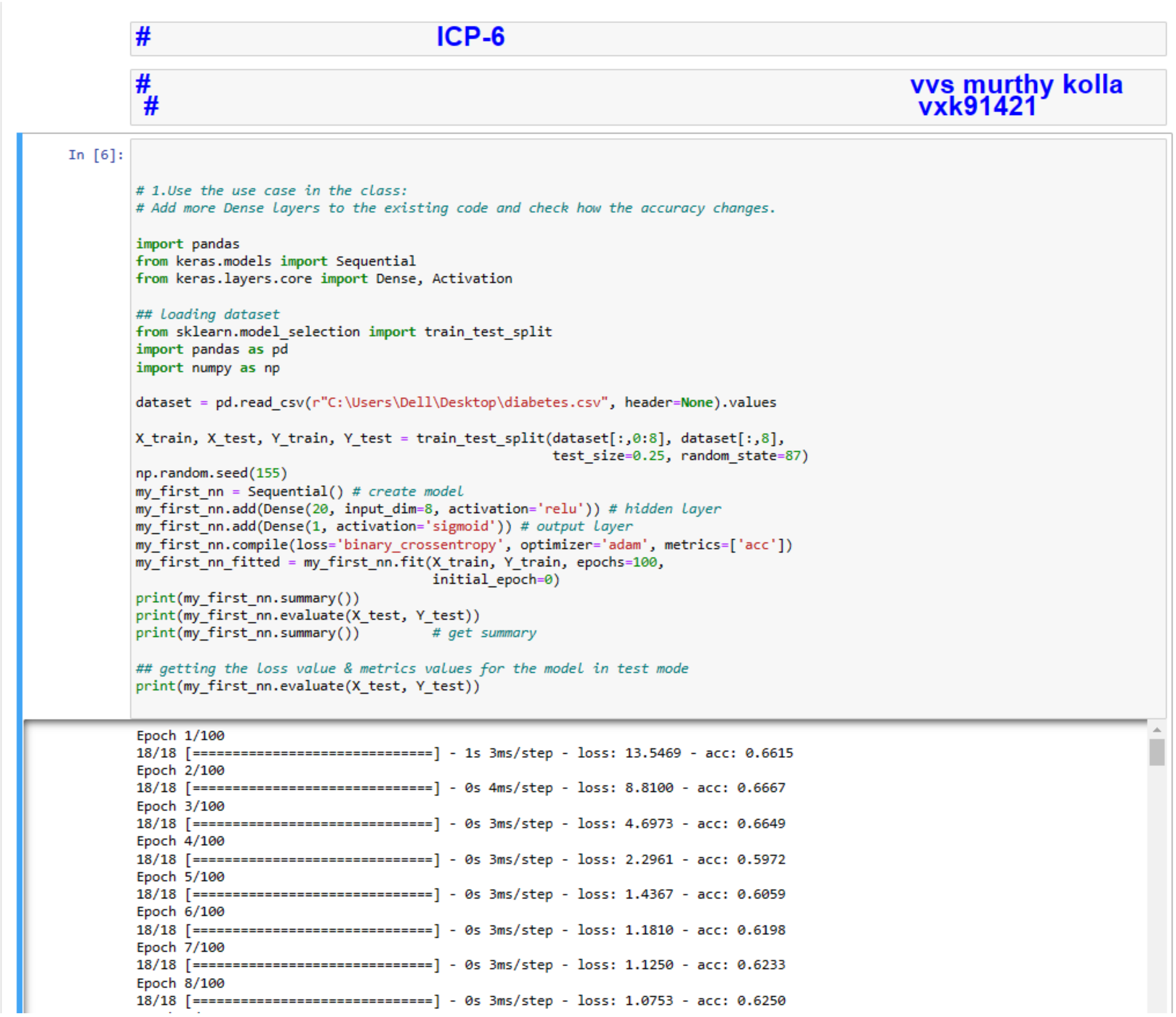
Assignment-6

vvsmurthy kolla
vxk91421

github link: https://github.com/murthykolla/ICP-6.git

video link:

https://drive.google.com/file/d/1NlWYDep06_L2YqsbTyeU_aKGwo767Mqm/view?usp=share_link



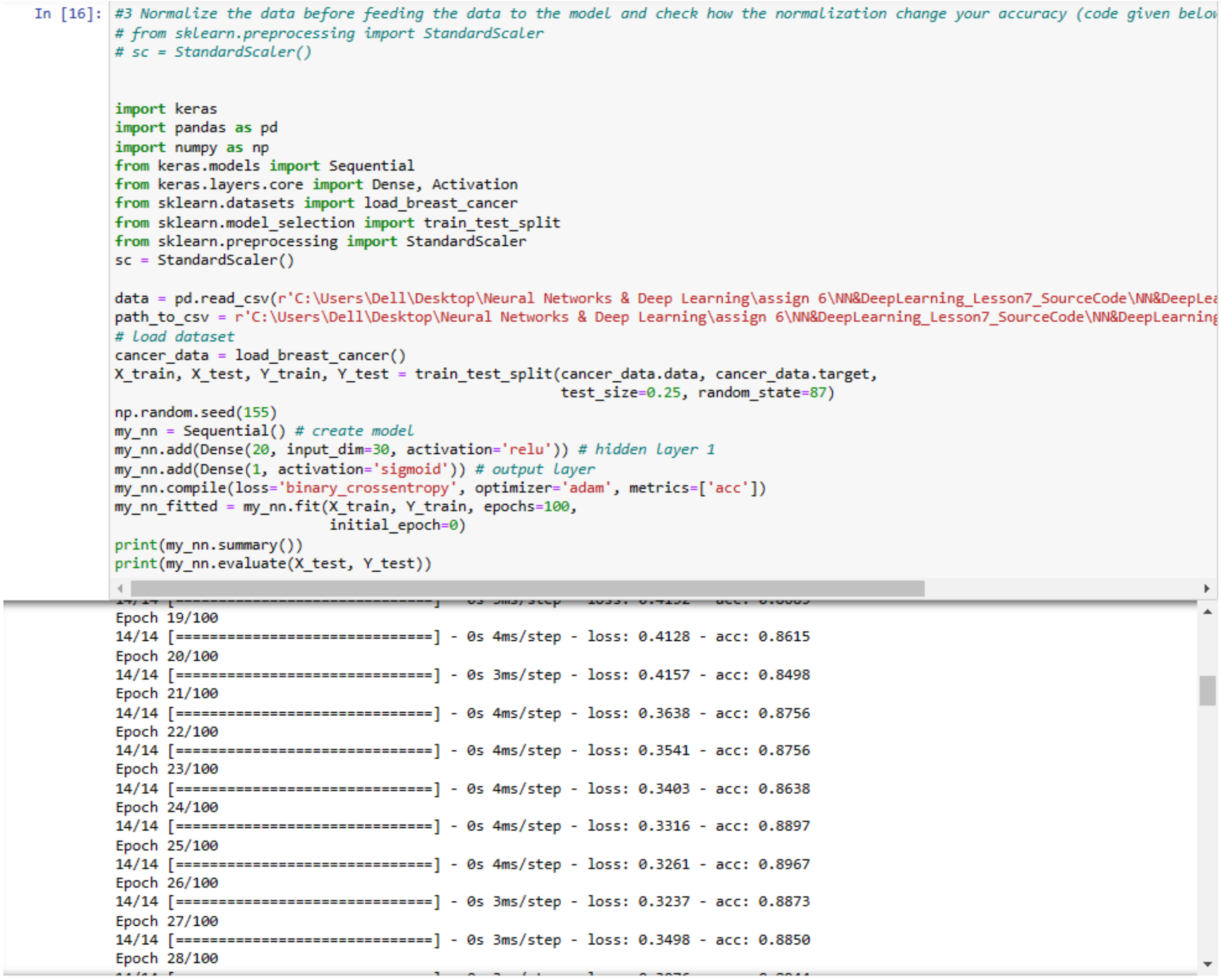
ICP-6

vvs murthy kolla
vxk91421

In [6]:

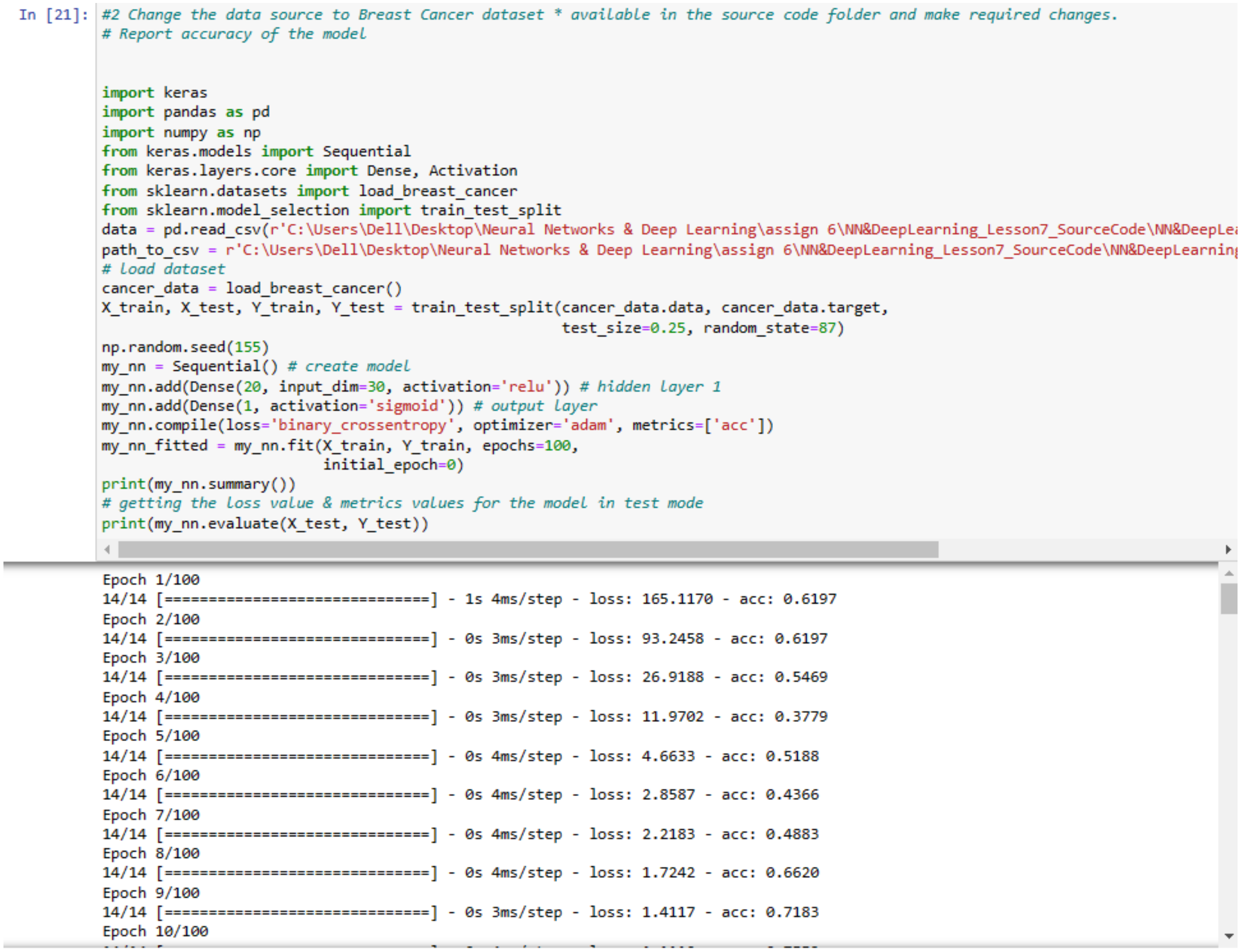
```
# 1.Use the use case in the class:
# Add more Dense layers to the existing code and check how the accuracy changes.

import pandas
from keras.models import Sequential
from keras.layers.core import Dense, Activation

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

dataset = pd.read_csv(r"C:\Users\Dell\Desktop\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))
print(my_first_nn.summary())         # get summary

## getting the loss value & metrics values for the model in test mode
print(my_first_nn.evaluate(X_test, Y_test))
```

```
Epoch 1/100
18/18 [==============================] - 1s 3ms/step - loss: 13.5469 - acc: 0.6615
Epoch 2/100
18/18 [==============================] - 0s 4ms/step - loss: 8.8100 - acc: 0.6667
Epoch 3/100
18/18 [==============================] - 0s 3ms/step - loss: 4.6973 - acc: 0.6649
Epoch 4/100
18/18 [==============================] - 0s 3ms/step - loss: 2.2961 - acc: 0.5972
Epoch 5/100
18/18 [==============================] - 0s 3ms/step - loss: 1.4367 - acc: 0.6059
Epoch 6/100
18/18 [==============================] - 0s 3ms/step - loss: 1.1810 - acc: 0.6198
Epoch 7/100
18/18 [==============================] - 0s 3ms/step - loss: 1.1250 - acc: 0.6233
Epoch 8/100
18/18 [==============================] - 0s 3ms/step - loss: 1.0753 - acc: 0.6250
```



```
In [16]: #3 Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given belo
         # from sklearn.preprocessing import StandardScaler
         # sc = StandardScaler()


         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
         from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()

         data = pd.read_csv(r'C:\Users\Dell\Desktop\Neural Networks & Deep Learning\assign 6\NN&DeepLearning_Lesson7_SourceCode\NN&DeepLea
         path_to_csv = r'C:\Users\Dell\Desktop\Neural Networks & Deep Learning\assign 6\NN&DeepLearning_Lesson7_SourceCode\NN&DeepLearning
         # 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(155)
         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))
```

```
Epoch 19/100
14/14 [==============================] - 0s 4ms/step - loss: 0.4128 - acc: 0.8615
Epoch 20/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4157 - acc: 0.8498
Epoch 21/100
14/14 [==============================] - 0s 4ms/step - loss: 0.3638 - acc: 0.8756
Epoch 22/100
14/14 [==============================] - 0s 4ms/step - loss: 0.3541 - acc: 0.8756
Epoch 23/100
14/14 [==============================] - 0s 4ms/step - loss: 0.3403 - acc: 0.8638
Epoch 24/100
14/14 [==============================] - 0s 4ms/step - loss: 0.3316 - acc: 0.8897
Epoch 25/100
14/14 [==============================] - 0s 4ms/step - loss: 0.3261 - acc: 0.8967
Epoch 26/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3237 - acc: 0.8873
Epoch 27/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3498 - acc: 0.8850
Epoch 28/100
```



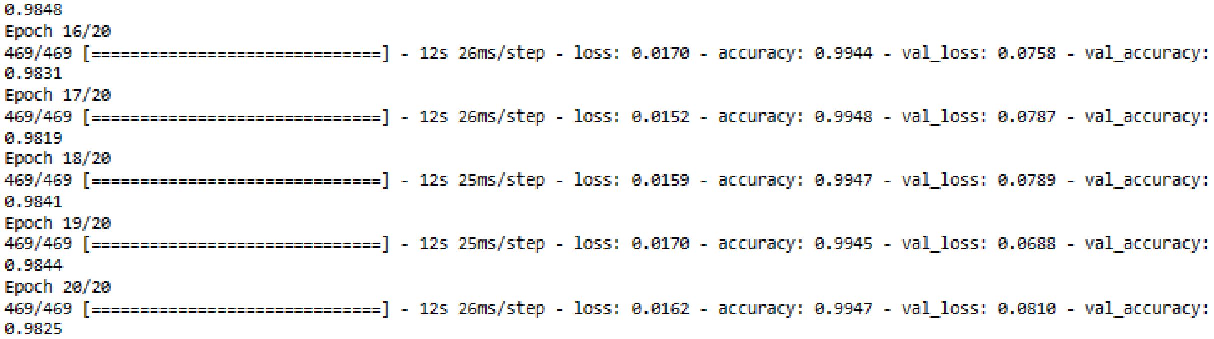
```
In [21]: #2 Change the data source to Breast Cancer dataset * available in the source code folder and make required changes.
         # Report accuracy of the model


         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
         data = pd.read_csv(r'C:\Users\Dell\Desktop\Neural Networks & Deep Learning\assign 6\NN&DeepLearning_Lesson7_SourceCode\NN&DeepLea
         path_to_csv = r'C:\Users\Dell\Desktop\Neural Networks & Deep Learning\assign 6\NN&DeepLearning_Lesson7_SourceCode\NN&DeepLearnin
         # 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(155)
         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())
         # getting the loss value & metrics values for the model in test mode
         print(my_nn.evaluate(X_test, Y_test))
```

```
Epoch 1/100
14/14 [==============================] - 1s 4ms/step - loss: 165.1170 - acc: 0.6197
Epoch 2/100
14/14 [==============================] - 0s 3ms/step - loss: 93.2458 - acc: 0.6197
Epoch 3/100
14/14 [==============================] - 0s 3ms/step - loss: 26.9188 - acc: 0.5469
Epoch 4/100
14/14 [==============================] - 0s 3ms/step - loss: 11.9702 - acc: 0.3779
Epoch 5/100
14/14 [==============================] - 0s 4ms/step - loss: 4.6633 - acc: 0.5188
Epoch 6/100
14/14 [==============================] - 0s 4ms/step - loss: 2.8587 - acc: 0.4366
Epoch 7/100
14/14 [==============================] - 0s 4ms/step - loss: 2.2183 - acc: 0.4883
Epoch 8/100
14/14 [==============================] - 0s 4ms/step - loss: 1.7242 - acc: 0.6620
Epoch 9/100
14/14 [==============================] - 0s 3ms/step - loss: 1.4117 - acc: 0.7183
Epoch 10/100
```

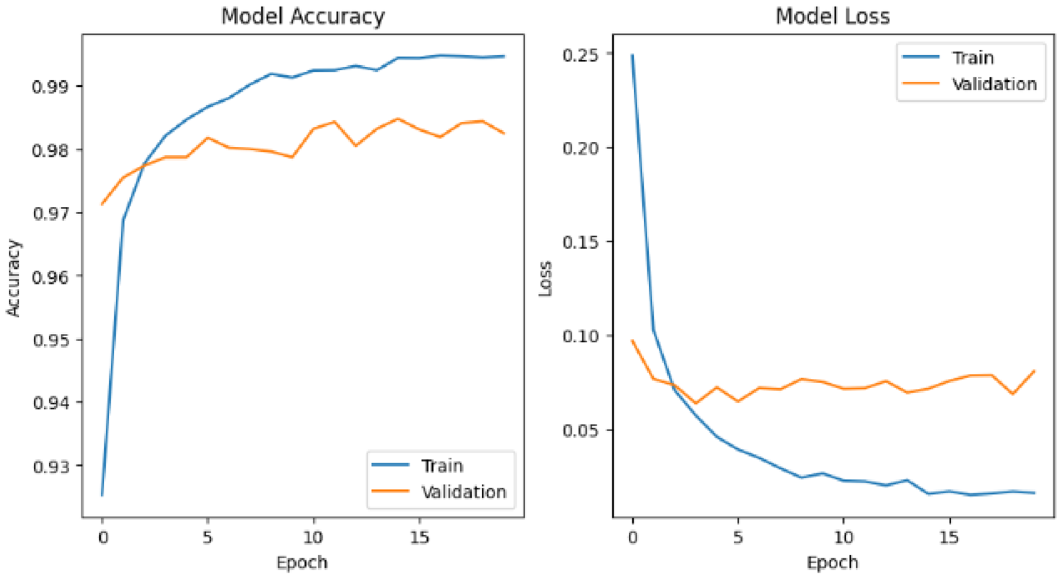
In [22]:

```
#2.1
#. 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 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
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()
```

```
Epoch 1/20
469/469 [==============================] - 14s 26ms/step - loss: 0.2488 - accuracy: 0.9253 - val_loss: 0.0971 - val_accuracy:
0.9713
Epoch 2/20
469/469 [==============================] - 12s 26ms/step - loss: 0.1029 - accuracy: 0.9688 - val_loss: 0.0769 - val_accuracy:
0.9755
Epoch 3/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0709 - accuracy: 0.9777 - val_loss: 0.0735 - val_accuracy:
0.9774
Epoch 4/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0574 - accuracy: 0.9821 - val_loss: 0.0638 - val_accuracy:
0.9787
```



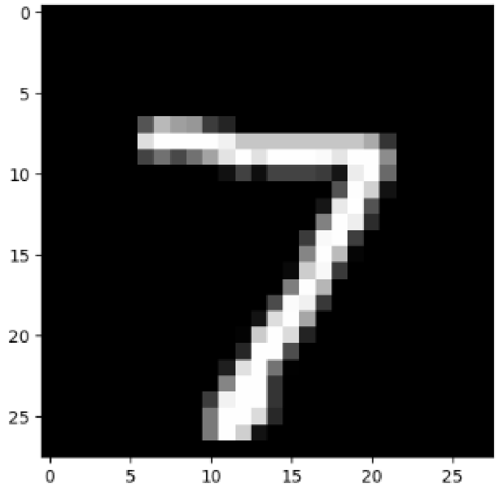
```
0.9848
Epoch 16/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0170 - accuracy: 0.9944 - val_loss: 0.0758 - val_accuracy:
0.9831
Epoch 17/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0152 - accuracy: 0.9948 - val_loss: 0.0787 - val_accuracy:
0.9819
Epoch 18/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0159 - accuracy: 0.9947 - val_loss: 0.0789 - val_accuracy:
0.9841
Epoch 19/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0170 - accuracy: 0.9945 - val_loss: 0.0688 - val_accuracy:
0.9844
Epoch 20/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0162 - accuracy: 0.9947 - val_loss: 0.0810 - val_accuracy:
0.9825
```



```python
#2.2 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(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 [==============================] - 13s 25ms/step - loss: 0.2530 - accuracy: 0.9243 - val_loss: 0.1092 - val_accuracy:
0.9668
Epoch 2/20
469/469 [==============================] - 11s 24ms/step - loss: 0.1002 - accuracy: 0.9701 - val_loss: 0.0798 - val_accuracy:
0.9739
Epoch 3/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0723 - accuracy: 0.9772 - val_loss: 0.0714 - val_accuracy:
0.9776
Epoch 4/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0555 - accuracy: 0.9825 - val_loss: 0.0668 - val_accuracy:
0.9796
Epoch 5/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0454 - accuracy: 0.9854 - val_loss: 0.0673 - val_accuracy:
```

```
469/469 [==============================] - 12s 25ms/step - loss: 0.0287 - accuracy: 0.9905 - val_loss: 0.0721 - val_accuracy:
0.9813
Epoch 10/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0255 - accuracy: 0.9915 - val_loss: 0.0661 - val_accuracy:
0.9830
Epoch 11/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0234 - accuracy: 0.9926 - val_loss: 0.0652 - val_accuracy:
0.9821
Epoch 12/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0215 - accuracy: 0.9928 - val_loss: 0.0660 - val_accuracy:
0.9833
Epoch 13/20
469/469 [==============================] - 16s 35ms/step - loss: 0.0189 - accuracy: 0.9936 - val_loss: 0.0753 - val_accuracy:
0.9837
Epoch 14/20
469/469 [==============================] - 18s 38ms/step - loss: 0.0201 - accuracy: 0.9934 - val_loss: 0.0814 - val_accuracy:
0.9819
Epoch 15/20
469/469 [==============================] - 14s 31ms/step - loss: 0.0199 - accuracy: 0.9933 - val_loss: 0.0713 - val_accuracy:
0.9835
Epoch 16/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0170 - accuracy: 0.9942 - val_loss: 0.0793 - val_accuracy:
0.9819
Epoch 17/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0180 - accuracy: 0.9941 - val_loss: 0.0825 - val_accuracy:
0.9826
Epoch 18/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0159 - accuracy: 0.9949 - val_loss: 0.0727 - val_accuracy:
0.9839
Epoch 19/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0148 - accuracy: 0.9949 - val_loss: 0.0787 - val_accuracy:
0.9827
Epoch 20/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0158 - accuracy: 0.9949 - val_loss: 0.0819 - val_accuracy:
0.9833
```



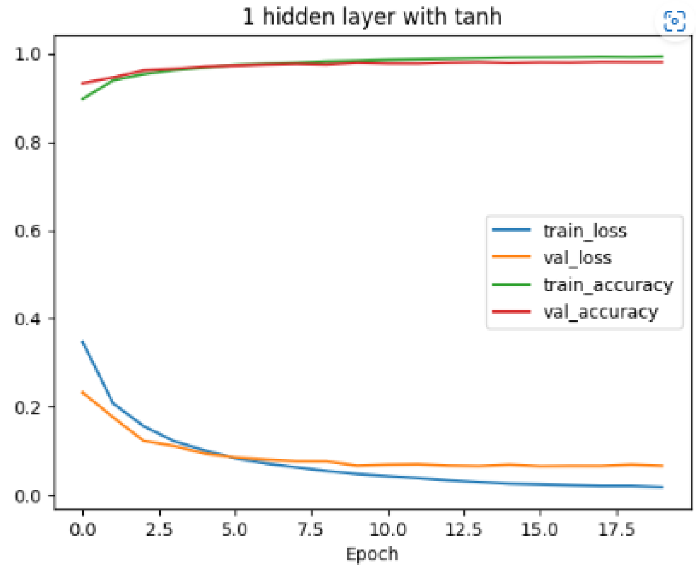
```
1/1 [==============================] - 0s 174ms/step
Model prediction: 7
```

]:

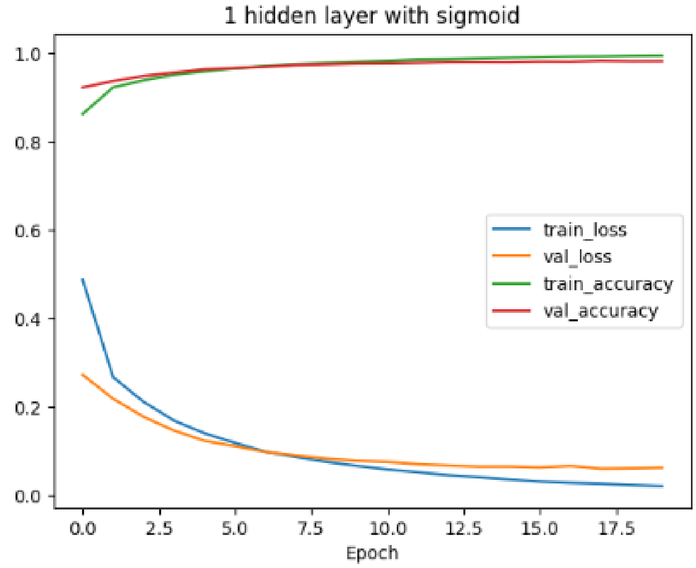
```
# 2.3We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the
#activation to tanh or sigmoid and see what happens
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot 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'])
    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()
```

In [2]:

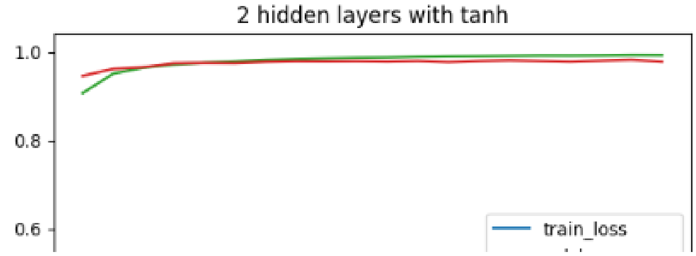
```
#2.4  Run the same code without scaling the images and check the performance?

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()

# 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'])
    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()
```



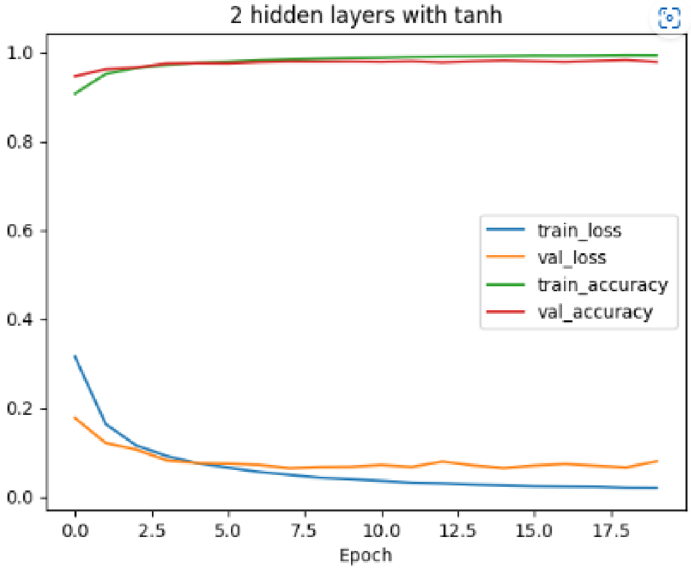
```
1 hidden layer with tanh - Test loss: 0.0665, Test accuracy: 0.9811
```



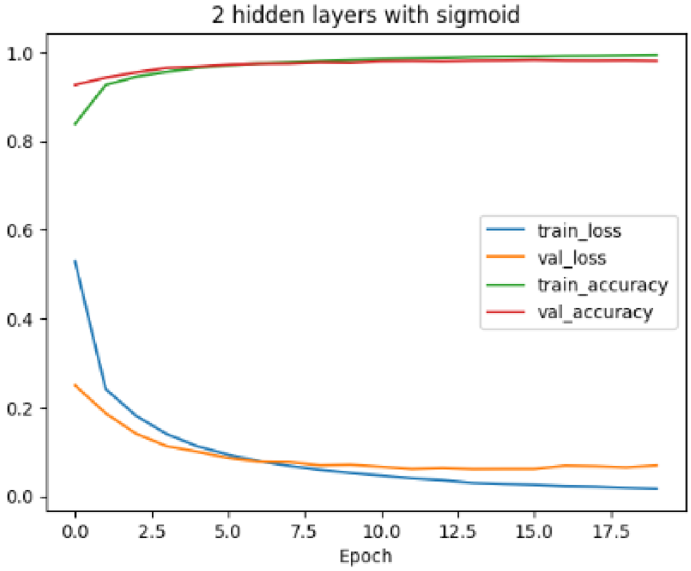
```
1 hidden layer with sigmoid - Test loss: 0.0633, Test accuracy: 0.9812
```



```
1 hidden layer with sigmoid - Test loss: 0.0633, Test accuracy: 0.9812
```



```
2 hidden layers with tanh - Test loss: 0.0796, Test accuracy: 0.9779
```



```
2 hidden layers with sigmoid - Test loss: 0.0692, Test accuracy: 0.9815
```