Neural Networks & Deep Learning: ICP1

Name: Pavani Medavarthi

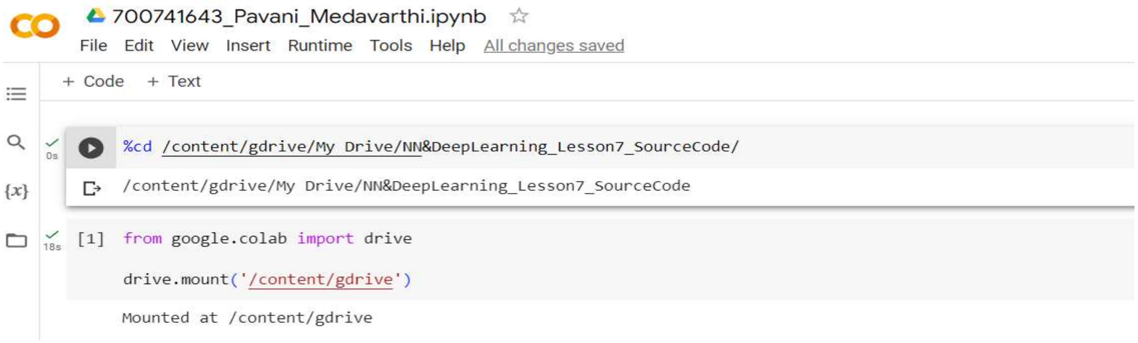
Student ID: 700741643

GitHub Link: https://github.com/Pavanimedavarthi/NN-DL_Summer-2

Video Link: https://drive.google.com/file/d/1gf_FQE4mllCSPQ162J3bhg699C-JSIlK/view?usp=sharing

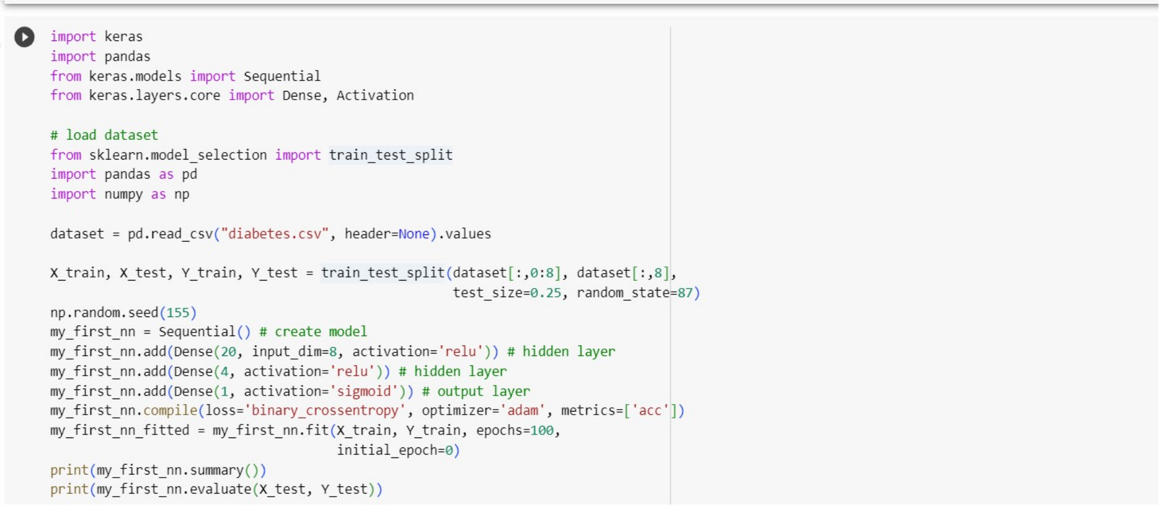
Problem 1

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



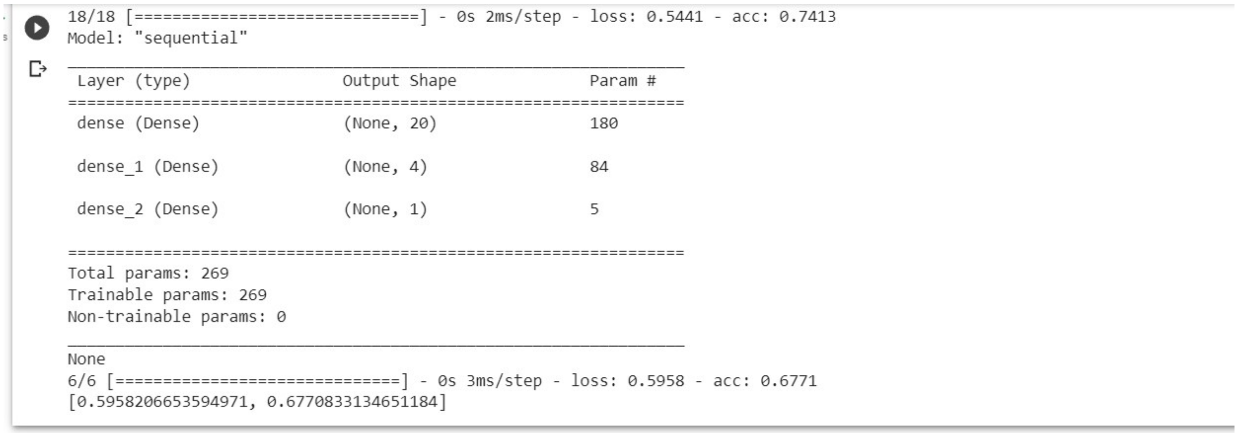
```
%cd /content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/
/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode
```

```
from google.colab import drive
drive.mount('/content/gdrive')
Mounted at /content/gdrive
```



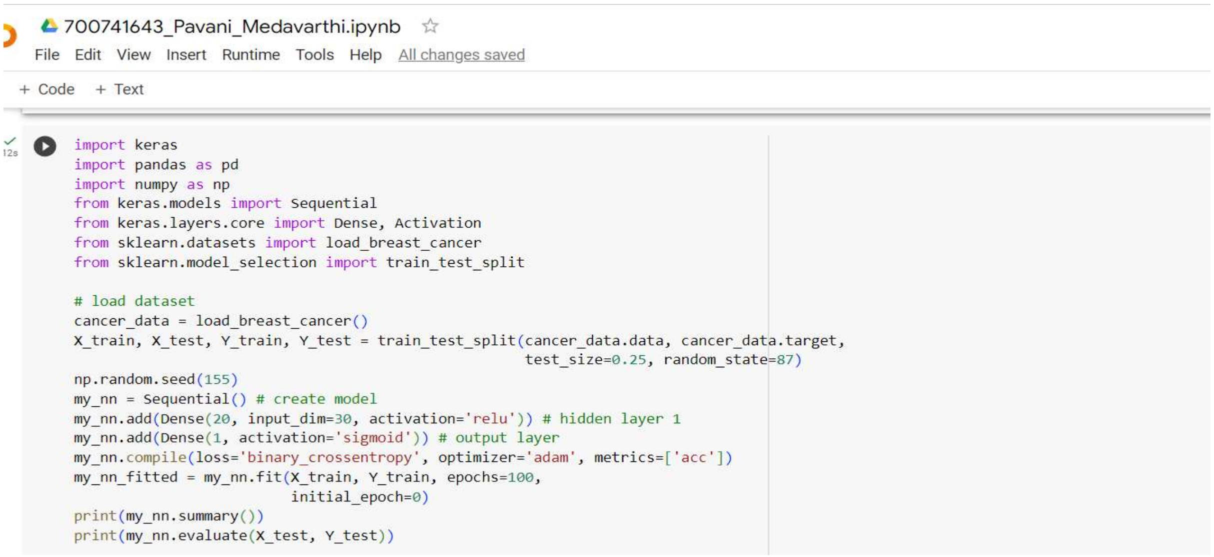
```python
import keras
import pandas
from keras.models import Sequential
from keras.layers.core import Dense, Activation

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

dataset = pd.read_csv("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(4, 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))
```

Executed Result:



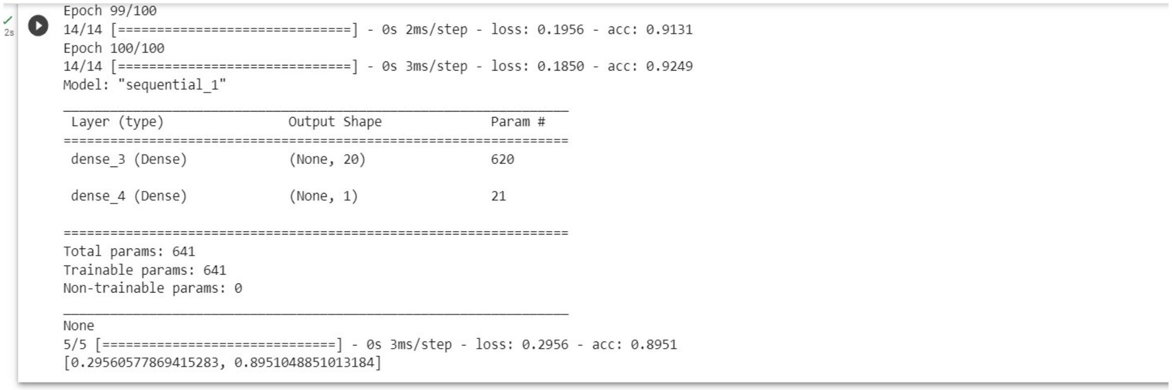
```
18/18 [==============================] - 0s 2ms/step - loss: 0.5441 - acc: 0.7413
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 20)                180

 dense_1 (Dense)             (None, 4)                 84

 dense_2 (Dense)             (None, 1)                 5

=================================================================
Total params: 269
Trainable params: 269
Non-trainable params: 0
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.5958 - acc: 0.6771
[0.5958206653594971, 0.6770833134651184]
```

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


700741643_Pavani_Medavarthi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
12s

```
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

# 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))
```

Executed Result:

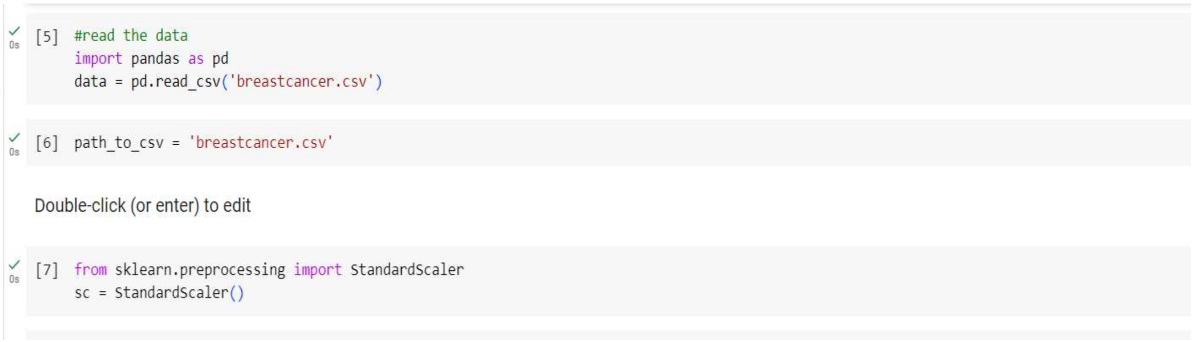


```
Epoch 99/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1956 - acc: 0.9131
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1850 - acc: 0.9249
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 20)                620

 dense_4 (Dense)             (None, 1)                 21

=================================================================
Total params: 641
Trainable params: 641
Non-trainable params: 0
_________________________________________________________________
None
5/5 [==============================] - 0s 3ms/step - loss: 0.2956 - acc: 0.8951
[0.29560577869415283, 0.8951048851013184]
```

3. 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()

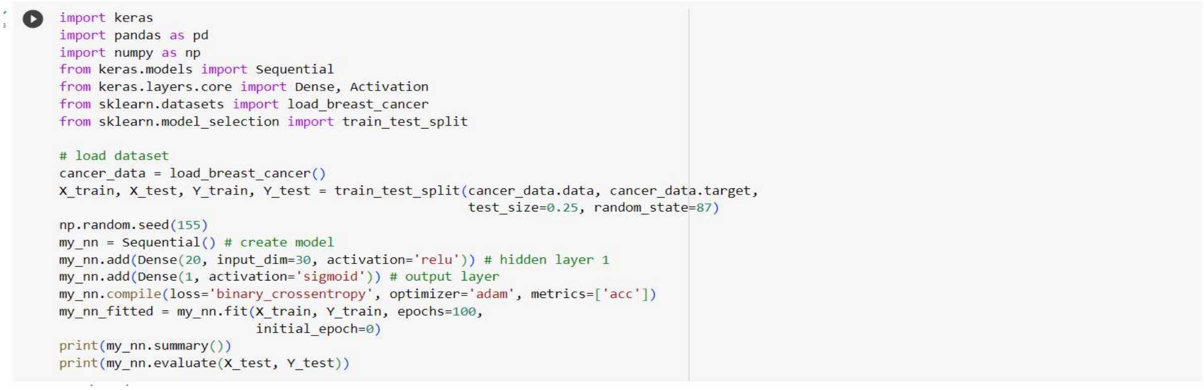


```
[5] #read the data
    import pandas as pd
    data = pd.read_csv('breastcancer.csv')
```

```
[6] path_to_csv = 'breastcancer.csv'
```

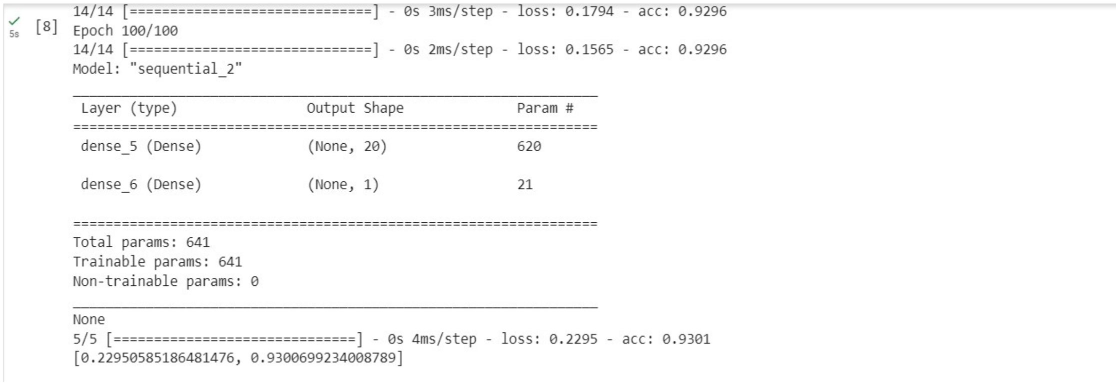
Double-click (or enter) to edit

```
[7] 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

# 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))
```

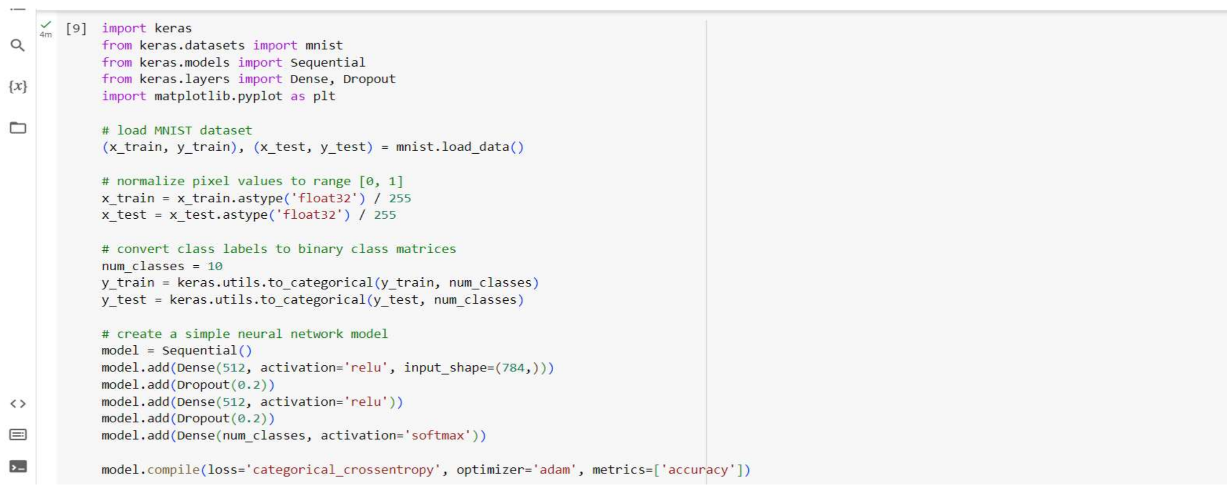
Executed Result:



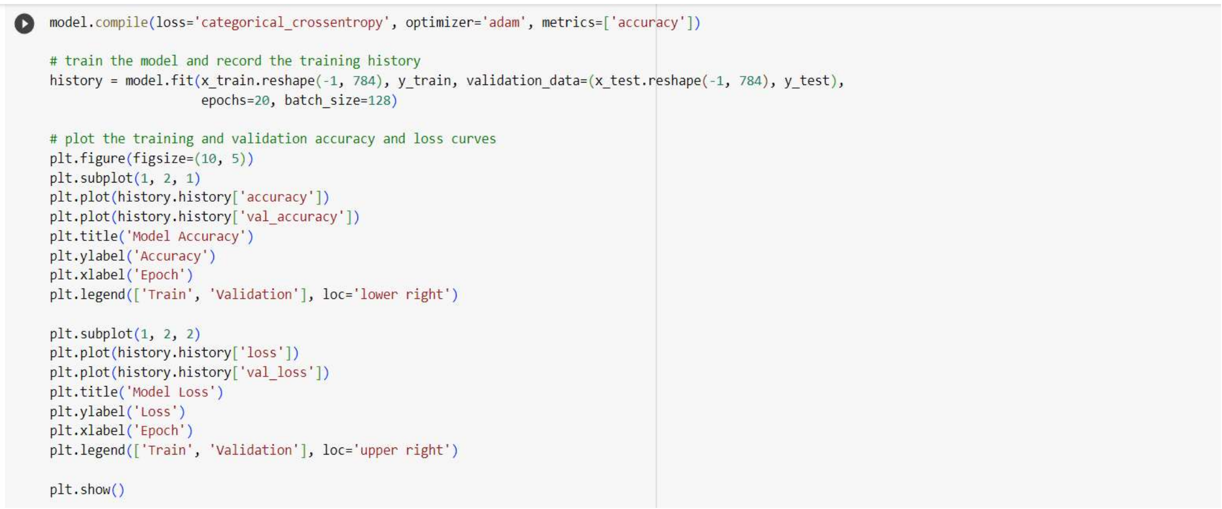
```
[8] 14/14 [==============================] - 0s 3ms/step - loss: 0.1794 - acc: 0.9296
    Epoch 100/100
    14/14 [==============================] - 0s 2ms/step - loss: 0.1565 - acc: 0.9296
    Model: "sequential_2"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     dense_5 (Dense)             (None, 20)                620

     dense_6 (Dense)             (None, 1)                 21

    =================================================================
    Total params: 641
    Trainable params: 641
    Non-trainable params: 0
    _________________________________________________________________
    None
    5/5 [==============================] - 0s 4ms/step - loss: 0.2295 - acc: 0.9301
    [0.22950585186481476, 0.9300699234008789]
```

Problem 2

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

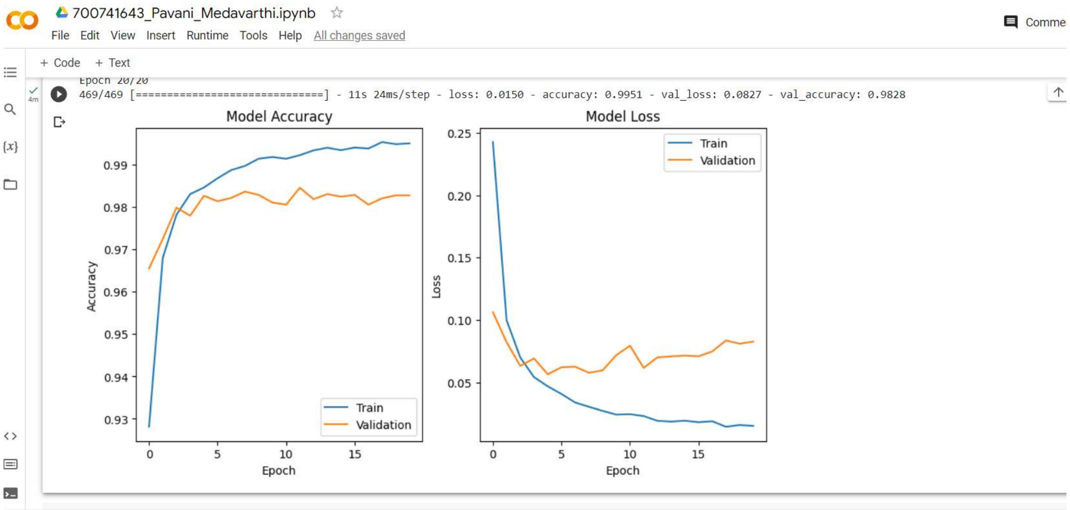


```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt

# 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'])
```



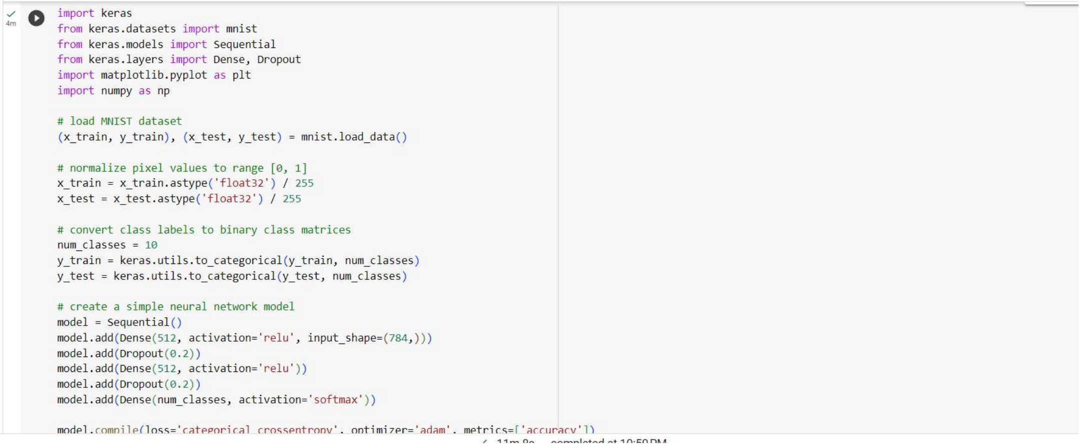
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# 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()
```

Executed Result:

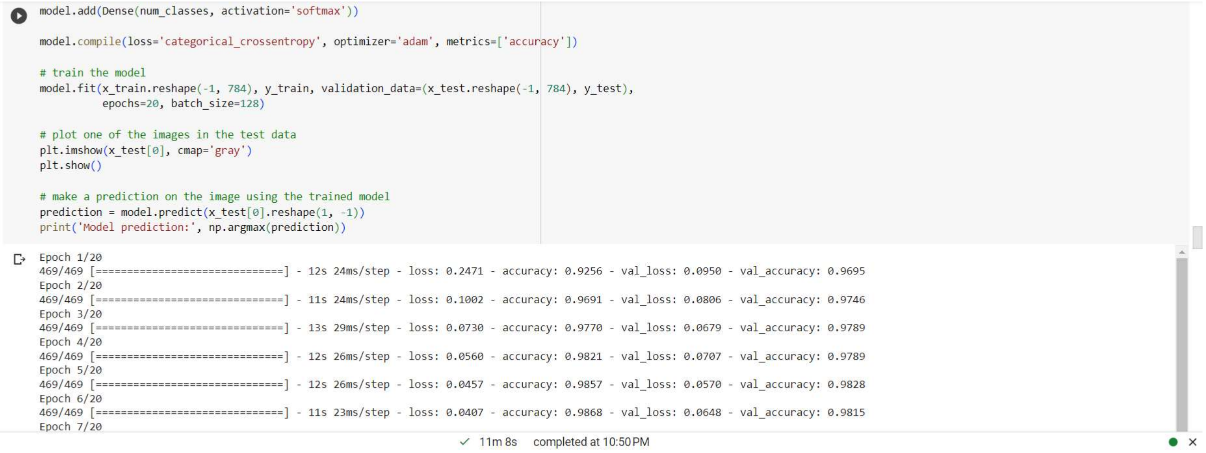


```
Epoch 20/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0150 - accuracy: 0.9951 - val_loss: 0.0827 - val_accuracy: 0.9828
```

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.



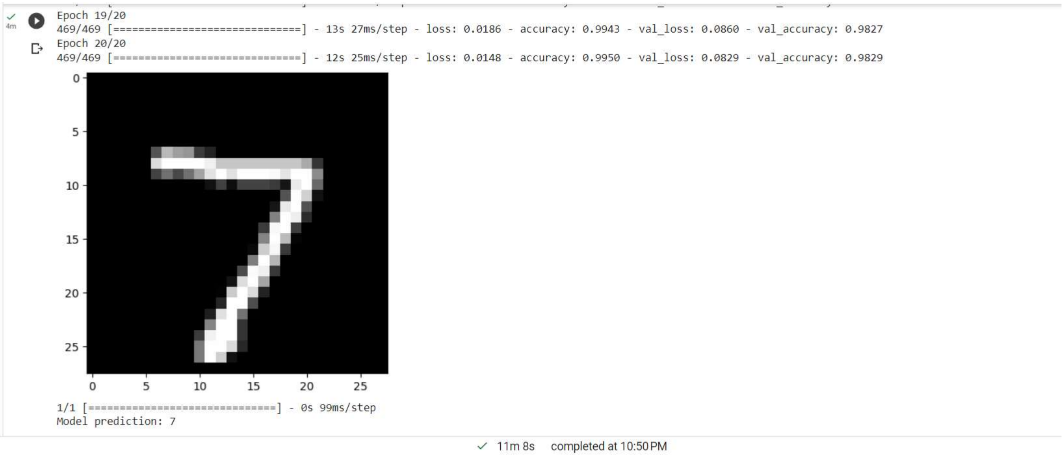
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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'])
```



```python
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 24ms/step - loss: 0.2471 - accuracy: 0.9256 - val_loss: 0.0950 - val_accuracy: 0.9695
Epoch 2/20
469/469 [==============================] - 11s 24ms/step - loss: 0.1002 - accuracy: 0.9691 - val_loss: 0.0806 - val_accuracy: 0.9746
Epoch 3/20
469/469 [==============================] - 13s 29ms/step - loss: 0.0730 - accuracy: 0.9770 - val_loss: 0.0679 - val_accuracy: 0.9789
Epoch 4/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0560 - accuracy: 0.9821 - val_loss: 0.0707 - val_accuracy: 0.9789
Epoch 5/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0457 - accuracy: 0.9857 - val_loss: 0.0570 - val_accuracy: 0.9828
Epoch 6/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0407 - accuracy: 0.9868 - val_loss: 0.0648 - val_accuracy: 0.9815
Epoch 7/20
```

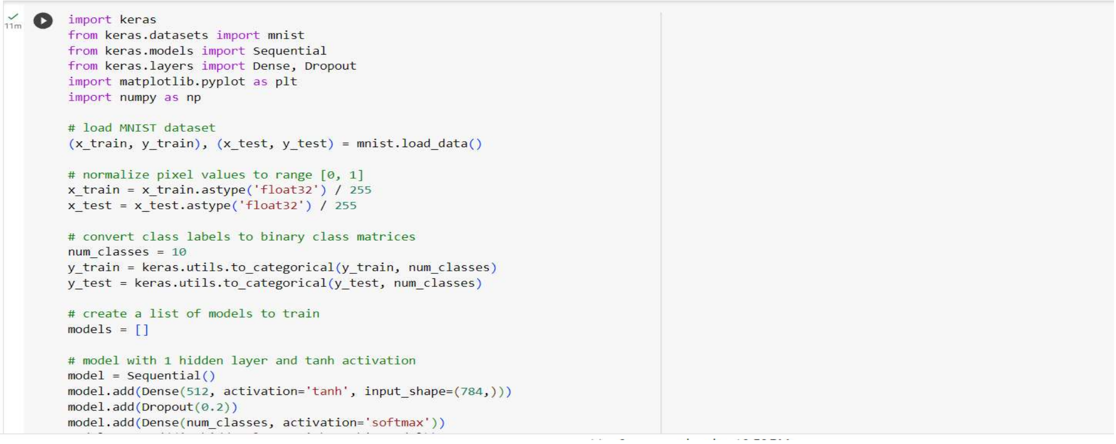
Executed Result:



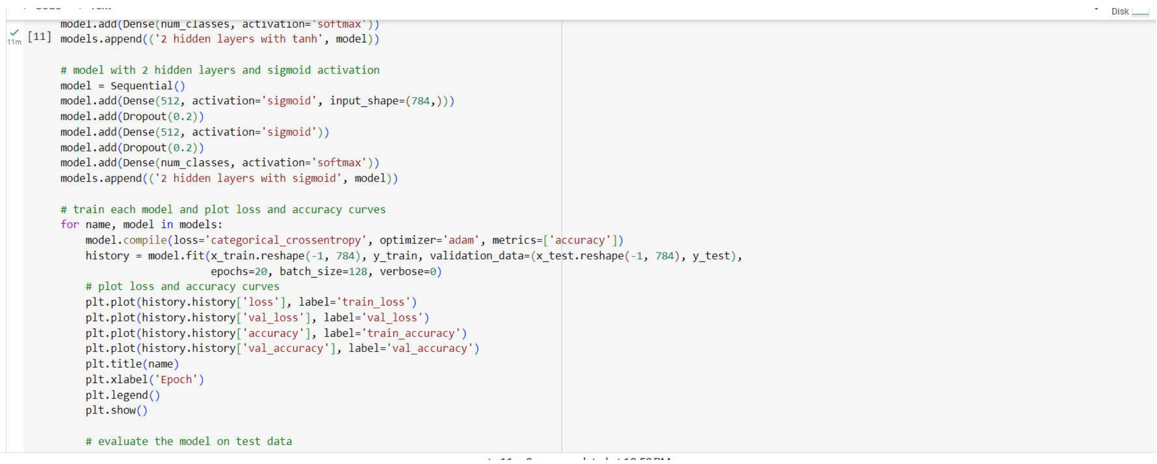
```
Epoch 19/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0186 - accuracy: 0.9943 - val_loss: 0.0860 - val_accuracy: 0.9827
Epoch 20/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0148 - accuracy: 0.9950 - val_loss: 0.0829 - val_accuracy: 0.9829
```

```
1/1 [==============================] - 0s 99ms/step
Model prediction: 7
```

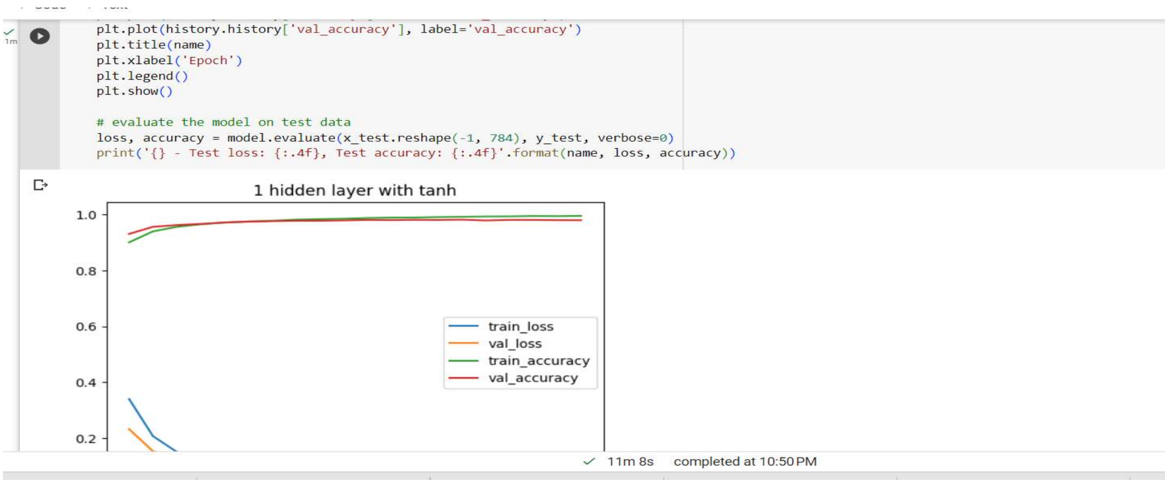
3.We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.



```python
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'))
```



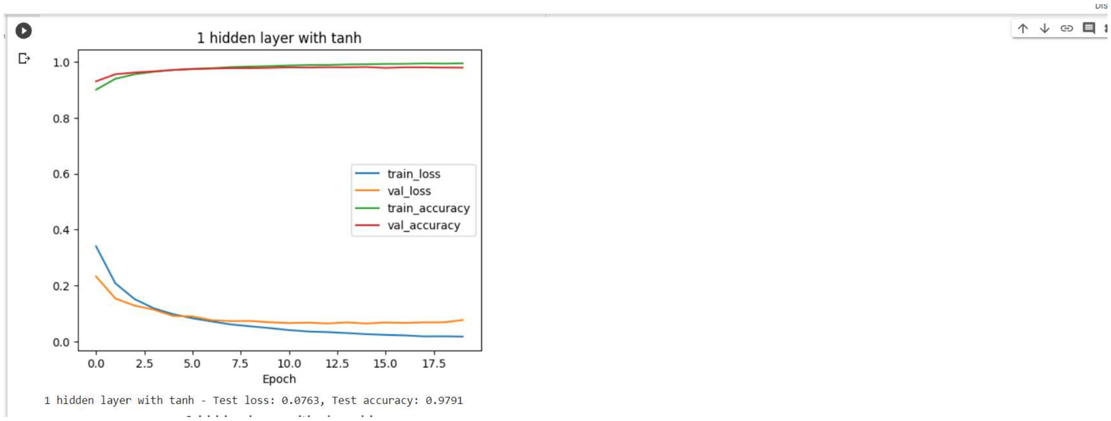
```
[11] model.add(Dense(num_classes, activation='softmax'))
     models.append(('2 hidden layers with tanh', model))

     # 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
```



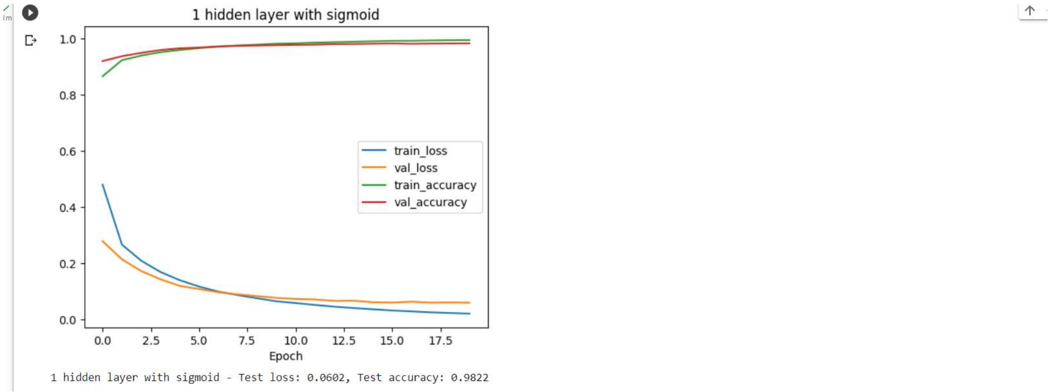
```
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.title(name)
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()

    # 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 8s completed at 10:50 PM

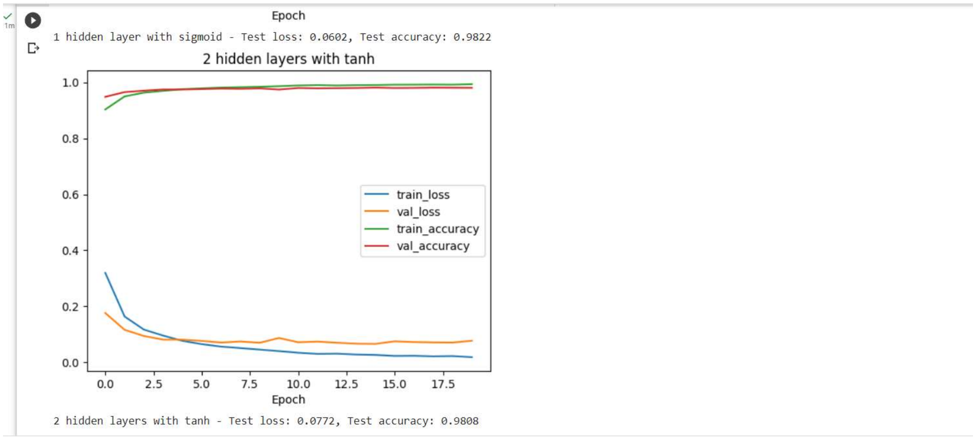
Executed Result:



```
1 hidden layer with tanh - Test loss: 0.0763, Test accuracy: 0.9791
```

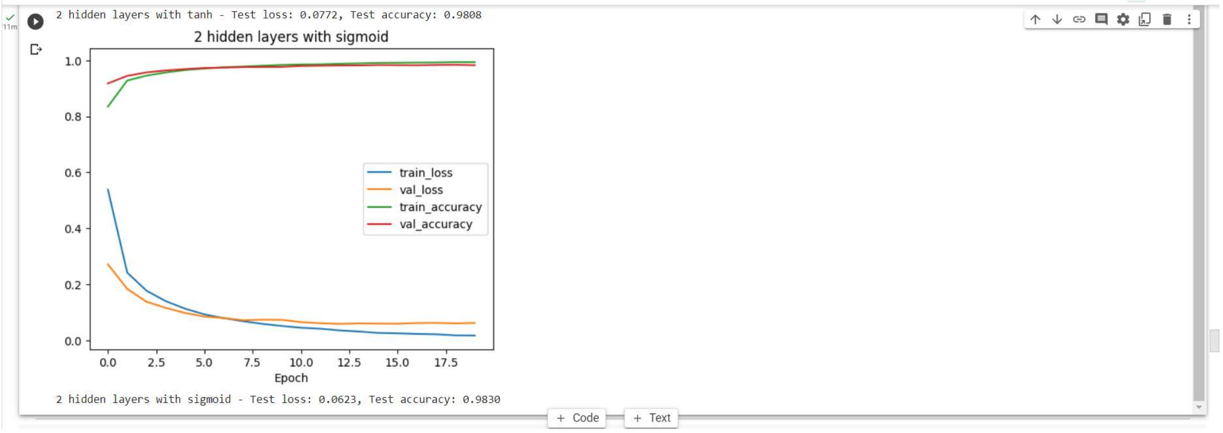


```
1 hidden layer with sigmoid - Test loss: 0.0602, Test accuracy: 0.9822
```



```
1 hidden layer with sigmoid - Test loss: 0.0602, Test accuracy: 0.9822
```

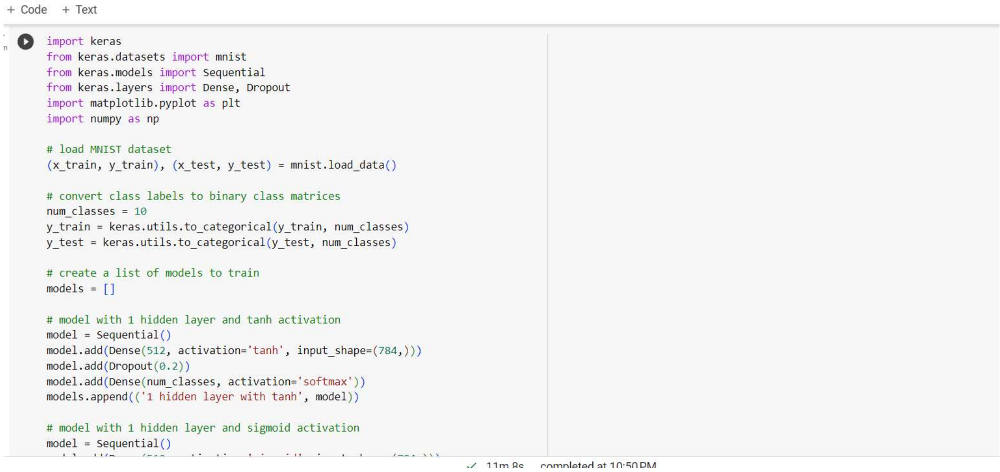
```
2 hidden layers with tanh - Test loss: 0.0772, Test accuracy: 0.9808
```



```
2 hidden layers with tanh - Test loss: 0.0772, Test accuracy: 0.9808
```

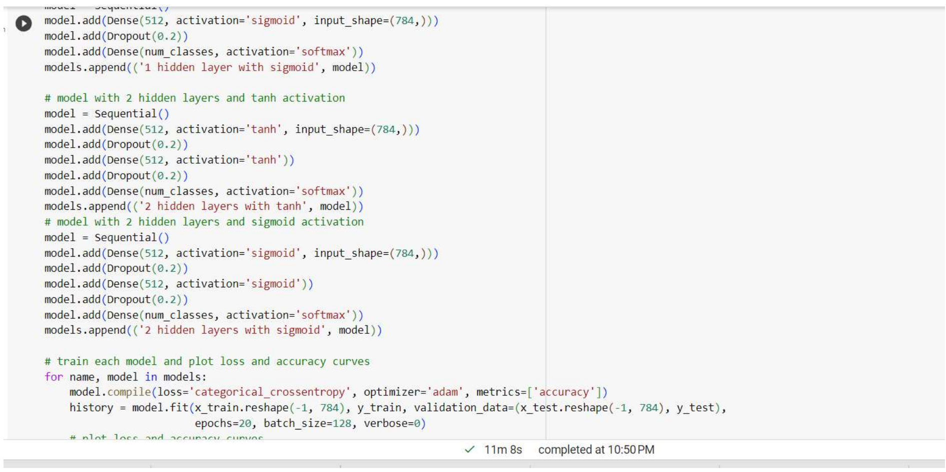
```
2 hidden layers with sigmoid - Test loss: 0.0623, Test accuracy: 0.9830
```

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



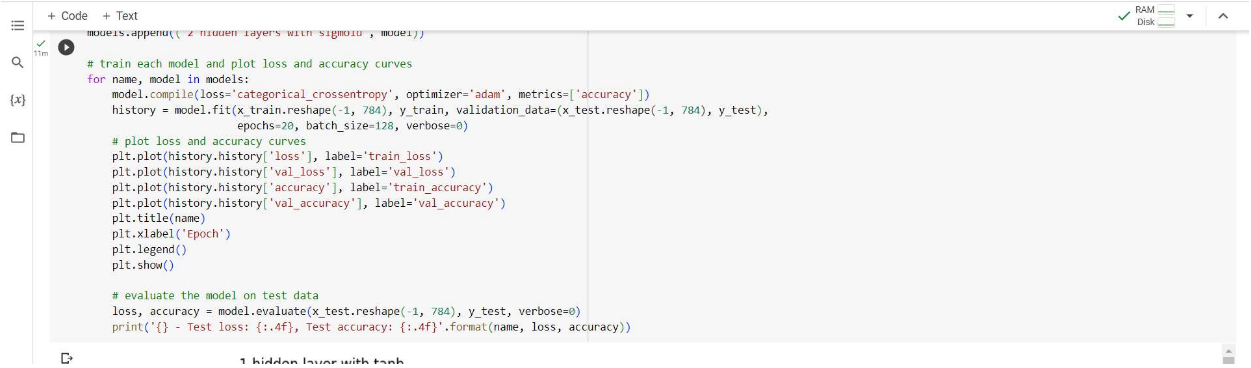
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

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

# 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,)))
```

✓ 11m 8s completed at 10:50 PM



```
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
```

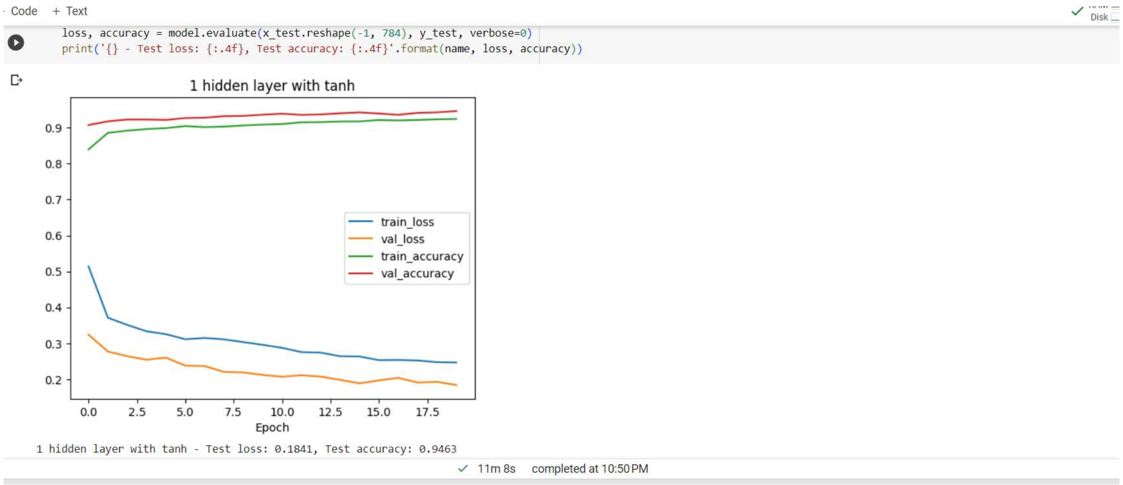
✓ 11m 8s completed at 10:50 PM



```
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))
```

1 hidden layer with tanh

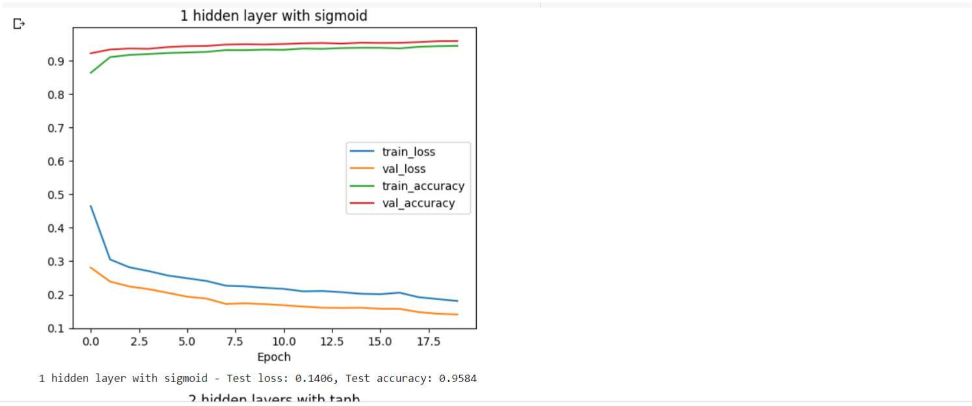
Executed Result:



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

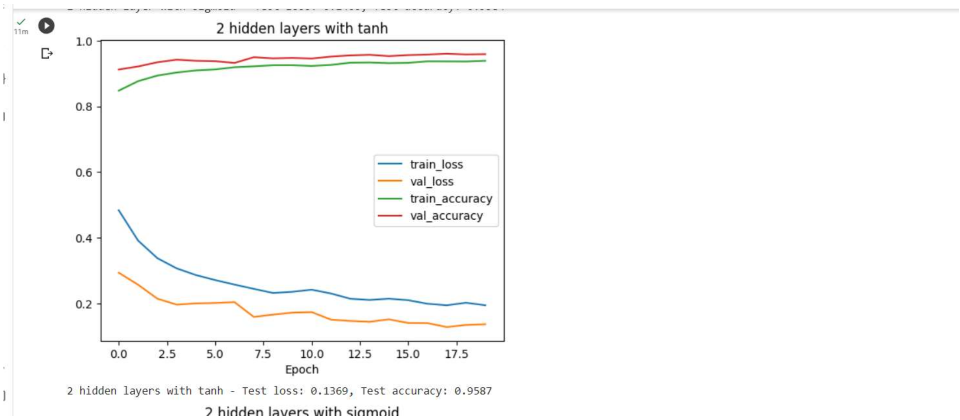
```
1 hidden layer with tanh - Test loss: 0.1841, Test accuracy: 0.9463
```

✓ 11m 8s completed at 10:50 PM

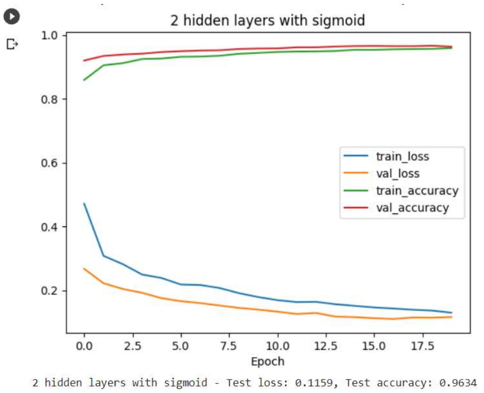


```
1 hidden layer with sigmoid - Test loss: 0.1406, Test accuracy: 0.9584
```

2 hidden layers with tanh



```
2 hidden layers with tanh - Test loss: 0.1369, Test accuracy: 0.9587
```



```
2 hidden layers with sigmoid - Test loss: 0.1159, Test accuracy: 0.9634
```