ASSIGNMENT-2

NEURAL NETWORKS & DEEP LEARNING

NAME:GAYATHRI KESHAMONI
ID:700742488

Github link:

https://github.com/GayathriKeshamoni/ASSIGNMENT-2-NEURAL-NETWORKS-DEEP-LEARNING/upload/main

Video Link: https://youtu.be/wAPfldFs4LQ

Description: Predictingthediabetesdisease.

Programmingelements: KerasBasics.

Inclassprogramming:

1. Usetheusecaseintheclass:a.AddmoreDenselayerstotheexistingcodeandcheckhowthe accuracy changes.

2. ChangethedatasourcetoBreastCancerdataset*availableinthesourcecodefolder andmakerequiredchanges.Report accuracyofthemodel.

3. Normalizethedatabeforefeedingthedatatothemodelandcheckhowthenormalizationchangeyouraccuracy(codegivenbelow).

fromsklearn.preprocessing
importStandardScalersc=StandardScaler().

BreastCancerdatasetisdesignatedtopredictifapatienthasMalignant(M)orBenign=B cancer.

Inclassprogramming:

UseImageClassificationonthehand writtendigitsdataset(mnist).

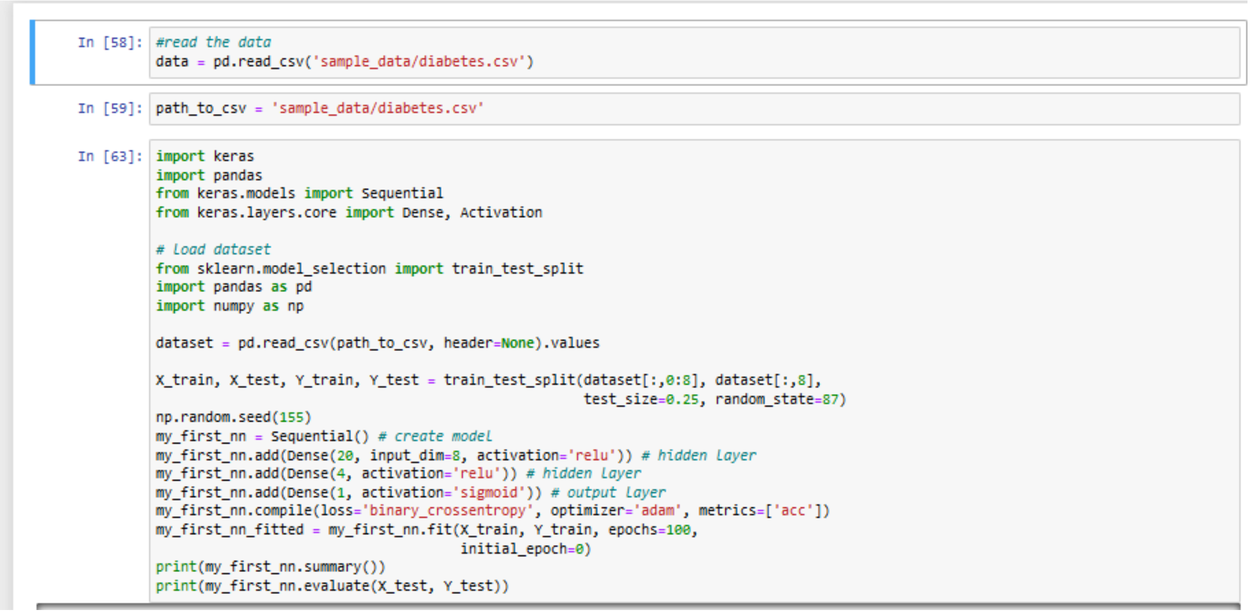
1. Plotthelossandaccuracyforbothtrainingdataandvalidationdatausingthehistoryobjectin thesourcecode.

2. Plotone of the imagesinthe testdata,andthendoinferencingtocheckwhatisthepredictionofthemodel on

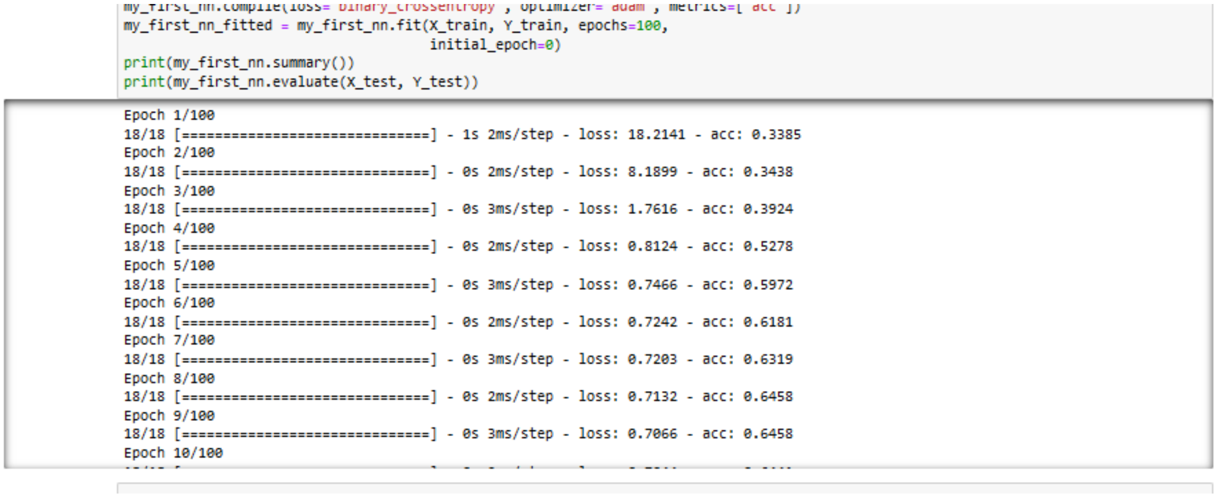
thatsingleimage.

3. Wehadused2hiddenlayersandReluactivation.Trytochangethenumberofhiddenlayer and theactivation totanh or sigmoid andseewhathappens.

4. Runthesamecode withoutscaling theimagesand checktheperformance?

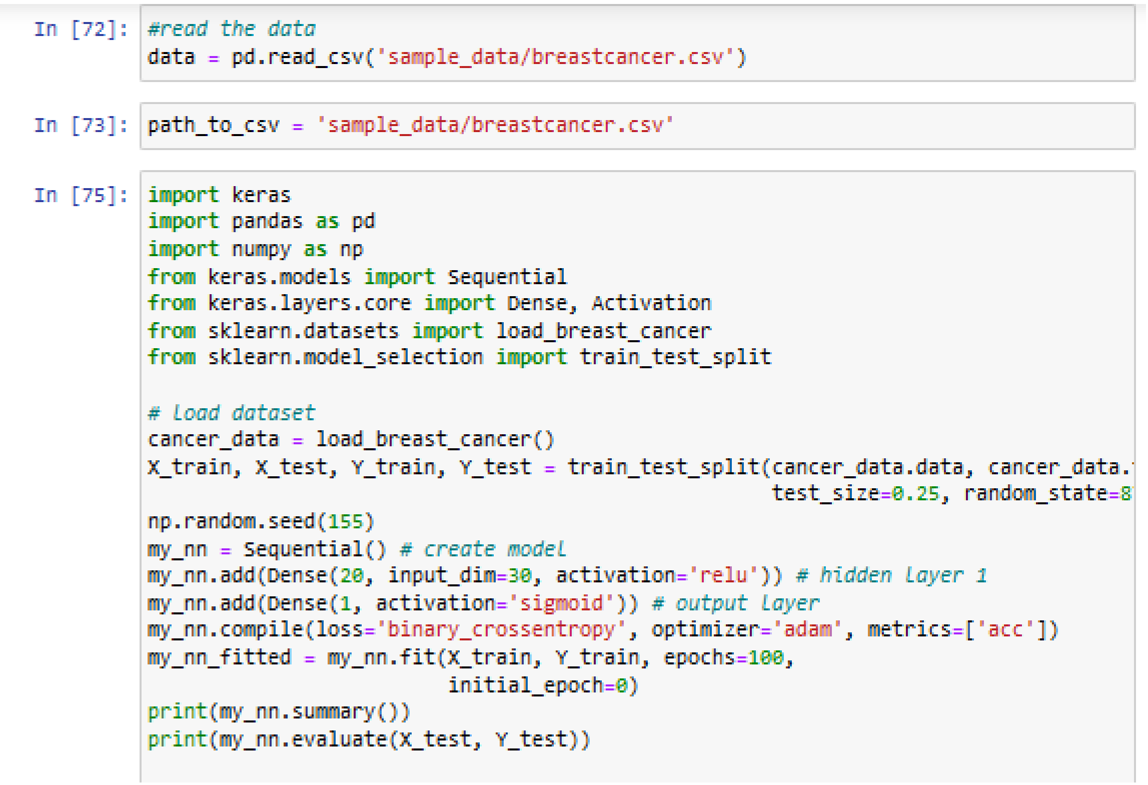


```
In [58]: #read the data
         data = pd.read_csv('sample_data/diabetes.csv')

In [59]: path_to_csv = 'sample_data/diabetes.csv'

In [63]: 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(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))
```

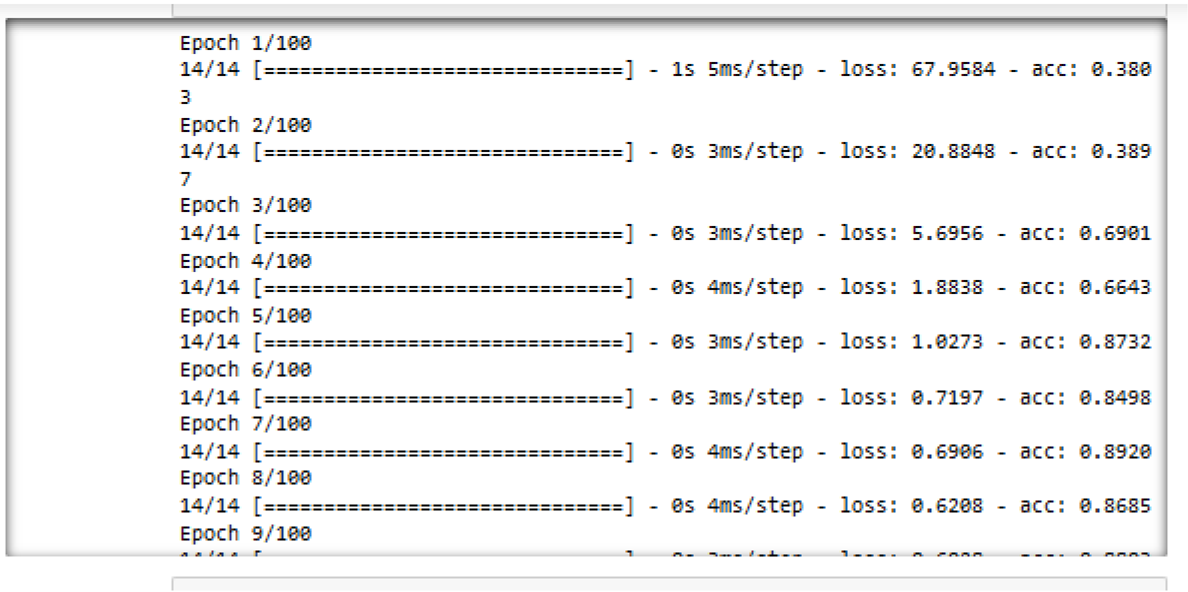


```
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test))
```

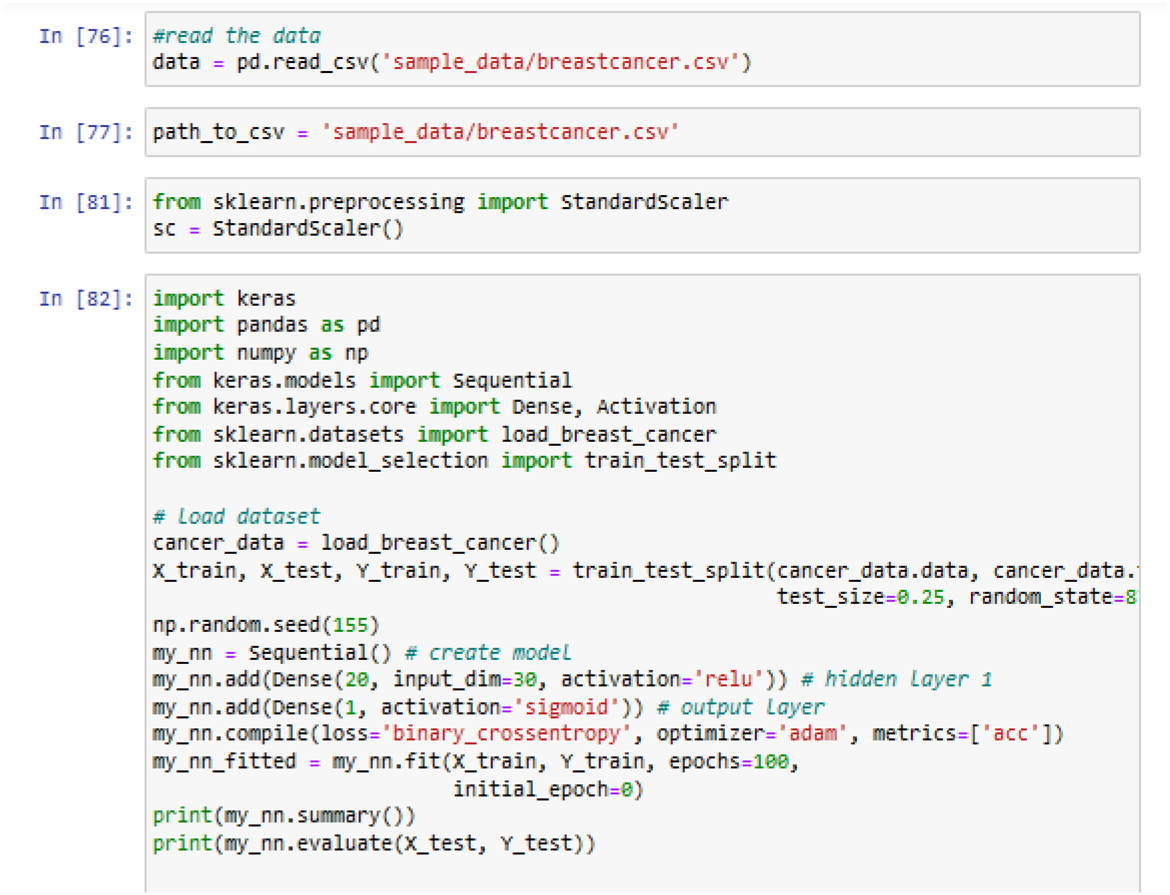
```
Epoch 1/100
18/18 [==============================] - 1s 2ms/step - loss: 18.2141 - acc: 0.3385
Epoch 2/100
18/18 [==============================] - 0s 2ms/step - loss: 8.1899 - acc: 0.3438
Epoch 3/100
18/18 [==============================] - 0s 3ms/step - loss: 1.7616 - acc: 0.3924
Epoch 4/100
18/18 [==============================] - 0s 2ms/step - loss: 0.8124 - acc: 0.5278
Epoch 5/100
18/18 [==============================] - 0s 3ms/step - loss: 0.7466 - acc: 0.5972
Epoch 6/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7242 - acc: 0.6181
Epoch 7/100
18/18 [==============================] - 0s 3ms/step - loss: 0.7203 - acc: 0.6319
Epoch 8/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7132 - acc: 0.6458
Epoch 9/100
18/18 [==============================] - 0s 3ms/step - loss: 0.7066 - acc: 0.6458
Epoch 10/100
```



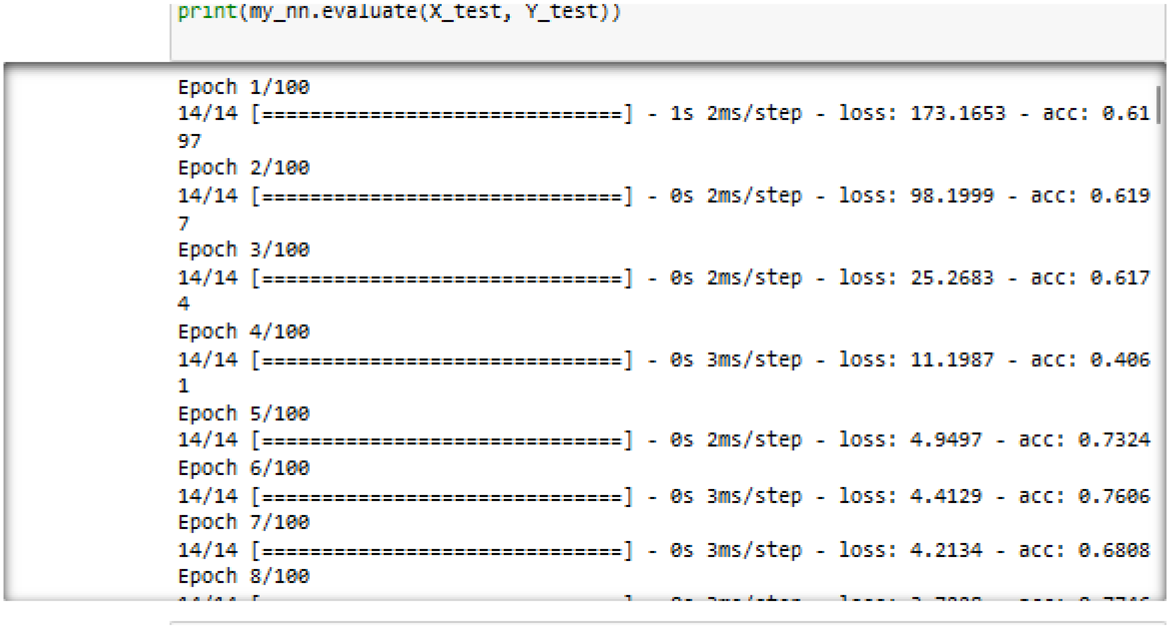
```
In [72]: #read the data
         data = pd.read_csv('sample_data/breastcancer.csv')

In [73]: path_to_csv = 'sample_data/breastcancer.csv'

In [75]: 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.
                                                             test_size=0.25, random_state=8
         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 1/100
14/14 [==============================] - 1s 5ms/step - loss: 67.9584 - acc: 0.380
3
Epoch 2/100
14/14 [==============================] - 0s 3ms/step - loss: 20.8848 - acc: 0.389
7
Epoch 3/100
14/14 [==============================] - 0s 3ms/step - loss: 5.6956 - acc: 0.6901
Epoch 4/100
14/14 [==============================] - 0s 4ms/step - loss: 1.8838 - acc: 0.6643
Epoch 5/100
14/14 [==============================] - 0s 3ms/step - loss: 1.0273 - acc: 0.8732
Epoch 6/100
14/14 [==============================] - 0s 3ms/step - loss: 0.7197 - acc: 0.8498
Epoch 7/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6906 - acc: 0.8920
Epoch 8/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6208 - acc: 0.8685
Epoch 9/100
```

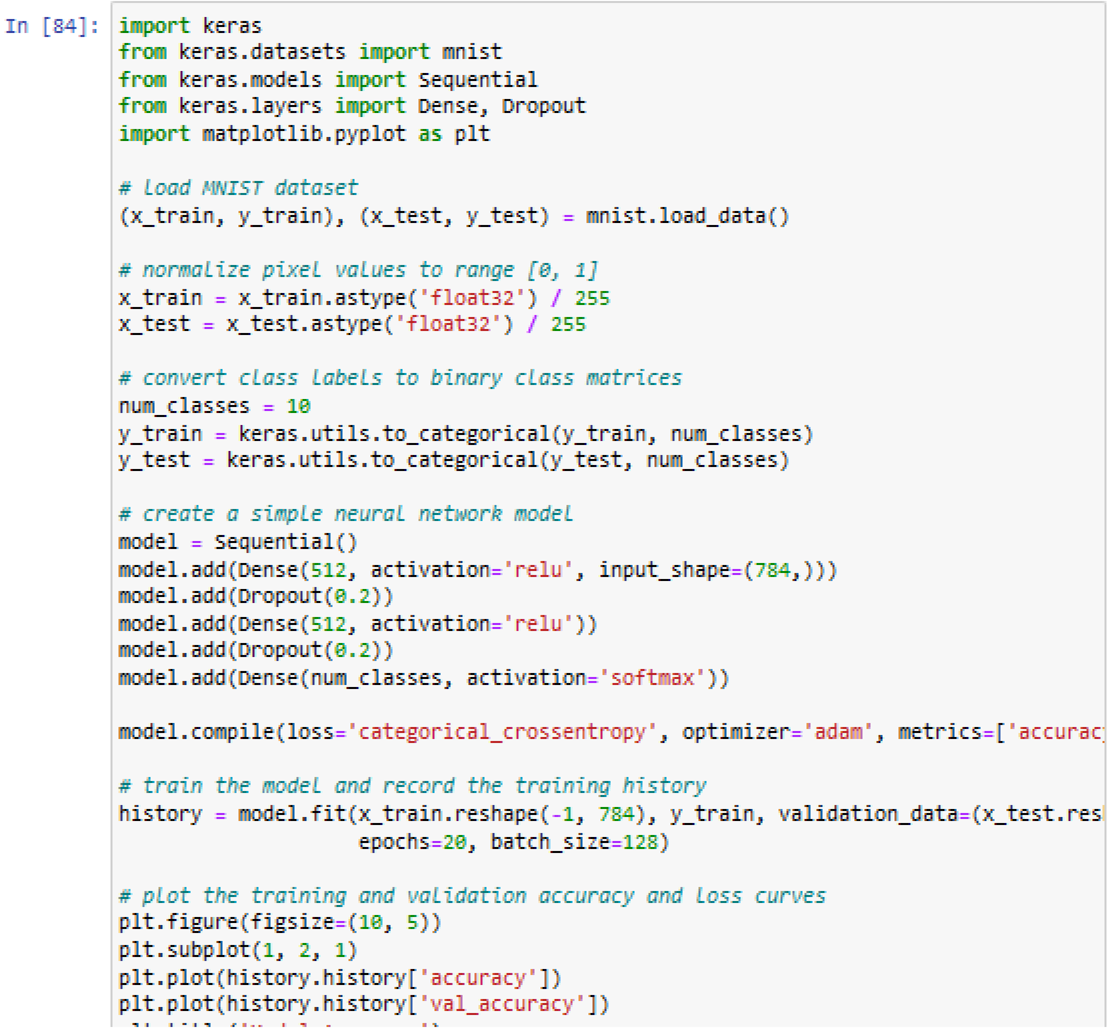


```
In [76]: #read the data
         data = pd.read_csv('sample_data/breastcancer.csv')

In [77]: path_to_csv = 'sample_data/breastcancer.csv'

In [81]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()

In [82]: 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.
                                                             test_size=0.25, random_state=8
         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))
```



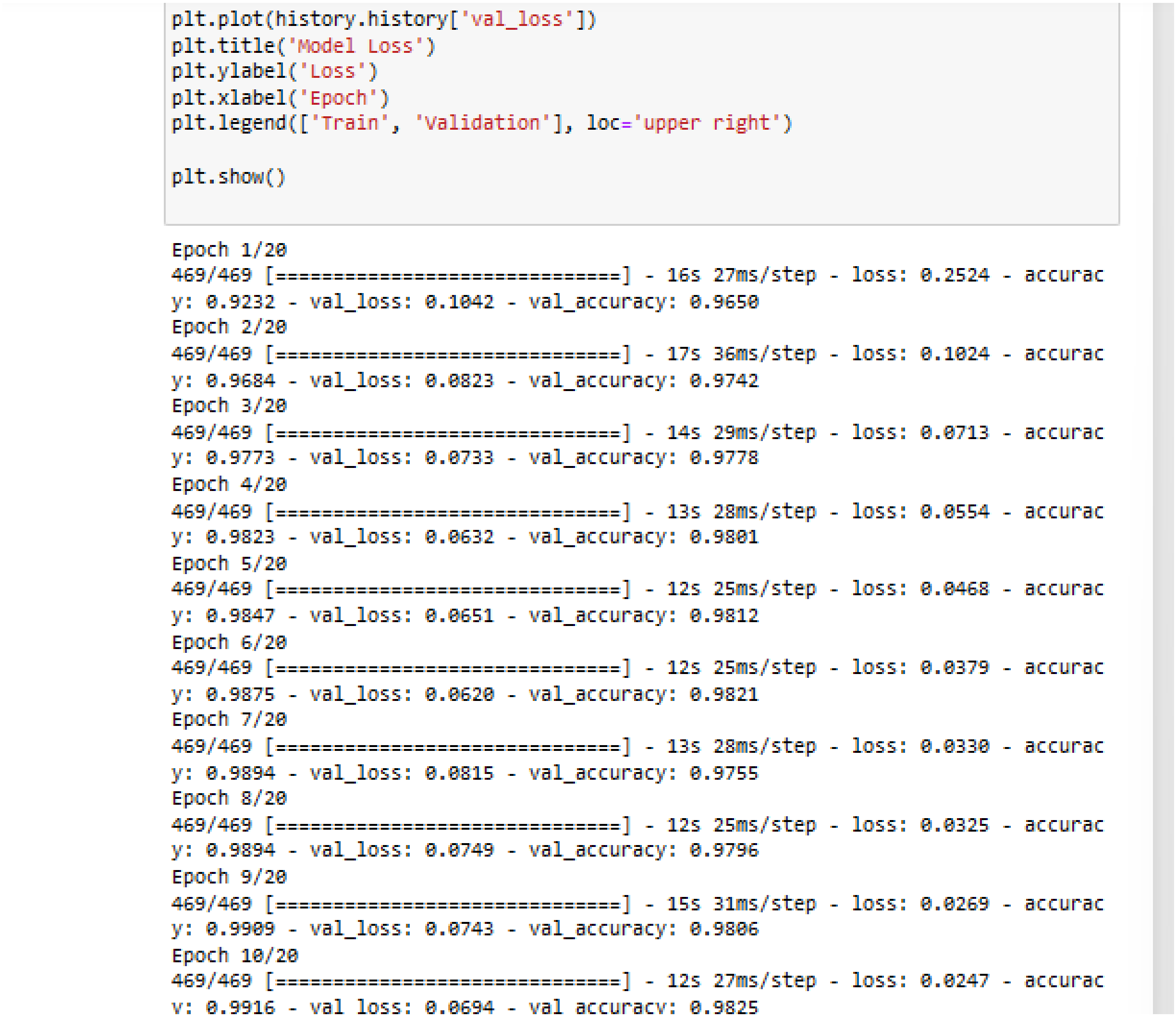
```
print(my_nn.evaluate(X_test, Y_test))
```

```
Epoch 1/100
14/14 [==============================] - 1s 2ms/step - loss: 173.1653 - acc: 0.61
97
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: 98.1999 - acc: 0.619
7
Epoch 3/100
14/14 [==============================] - 0s 2ms/step - loss: 25.2683 - acc: 0.617
4
Epoch 4/100
14/14 [==============================] - 0s 3ms/step - loss: 11.1987 - acc: 0.406
1
Epoch 5/100
14/14 [==============================] - 0s 2ms/step - loss: 4.9497 - acc: 0.7324
Epoch 6/100
14/14 [==============================] - 0s 3ms/step - loss: 4.4129 - acc: 0.7606
Epoch 7/100
14/14 [==============================] - 0s 3ms/step - loss: 4.2134 - acc: 0.6808
Epoch 8/100
```



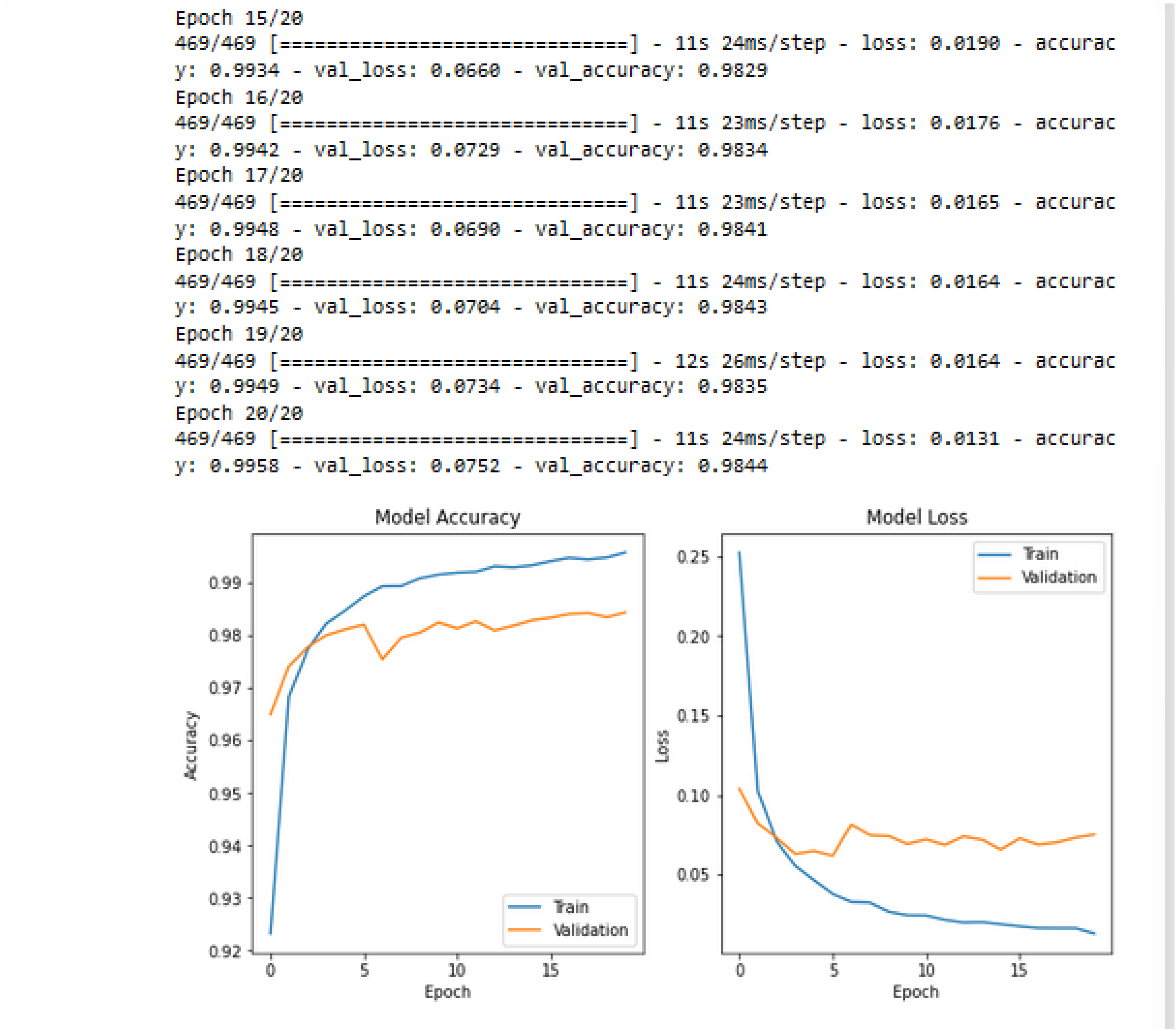
In [84]:

```
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=['accurac

# train the model and record the training history
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.res
                    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'])
```

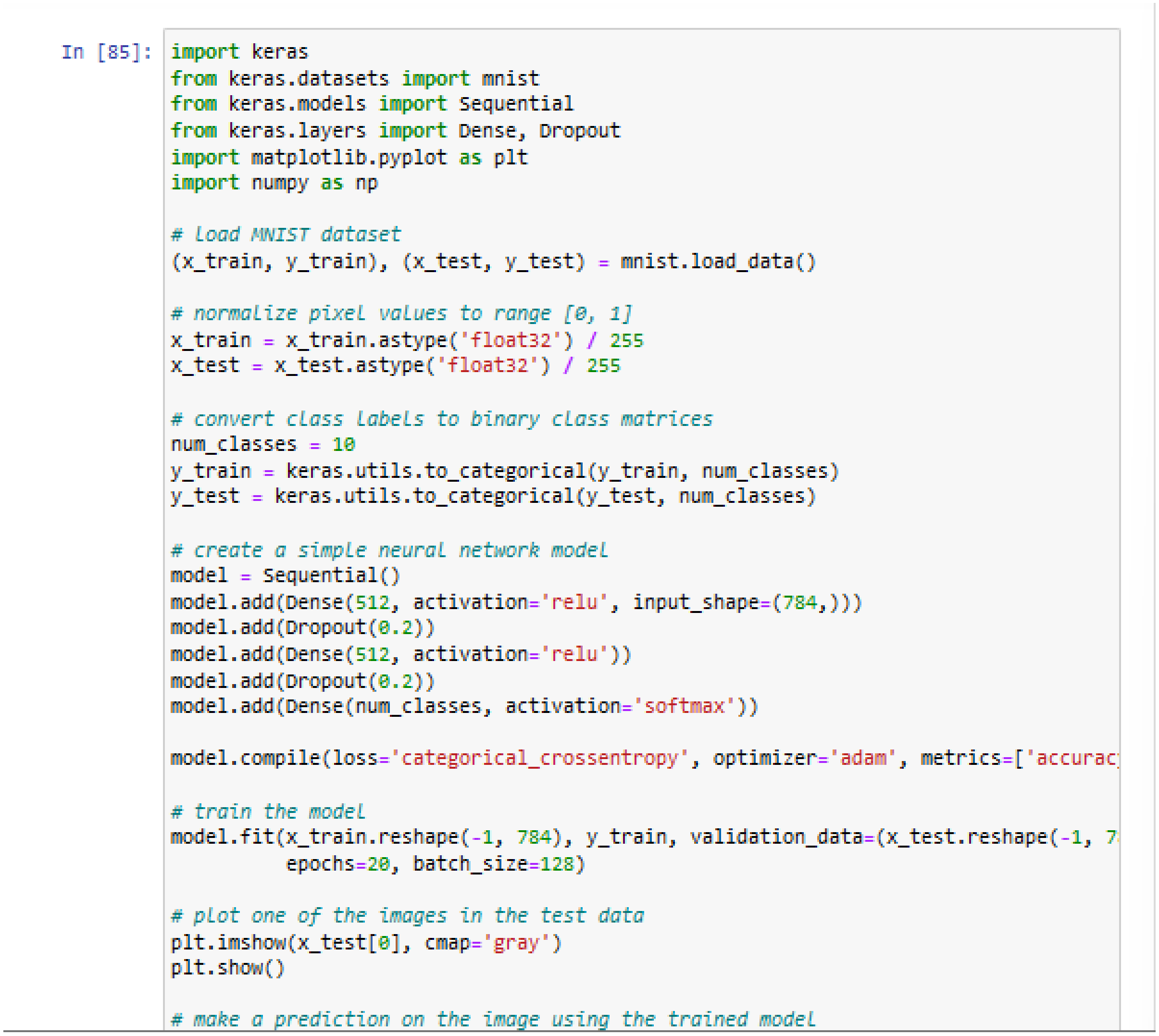


```
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.show()
```

```
Epoch 1/20
469/469 [==============================] - 16s 27ms/step - loss: 0.2524 - accurac
y: 0.9232 - val_loss: 0.1042 - val_accuracy: 0.9650
Epoch 2/20
469/469 [==============================] - 17s 36ms/step - loss: 0.1024 - accurac
y: 0.9684 - val_loss: 0.0823 - val_accuracy: 0.9742
Epoch 3/20
469/469 [==============================] - 14s 29ms/step - loss: 0.0713 - accurac
y: 0.9773 - val_loss: 0.0733 - val_accuracy: 0.9778
Epoch 4/20
469/469 [==============================] - 13s 28ms/step - loss: 0.0554 - accurac
y: 0.9823 - val_loss: 0.0632 - val_accuracy: 0.9801
Epoch 5/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0468 - accurac
y: 0.9847 - val_loss: 0.0651 - val_accuracy: 0.9812
Epoch 6/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0379 - accurac
y: 0.9875 - val_loss: 0.0620 - val_accuracy: 0.9821
Epoch 7/20
469/469 [==============================] - 13s 28ms/step - loss: 0.0330 - accurac
y: 0.9894 - val_loss: 0.0815 - val_accuracy: 0.9755
Epoch 8/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0325 - accurac
y: 0.9894 - val_loss: 0.0749 - val_accuracy: 0.9796
Epoch 9/20
469/469 [==============================] - 15s 31ms/step - loss: 0.0269 - accurac
y: 0.9909 - val_loss: 0.0743 - val_accuracy: 0.9806
Epoch 10/20
469/469 [==============================] - 12s 27ms/step - loss: 0.0247 - accurac
y: 0.9916 - val_loss: 0.0694 - val_accuracy: 0.9825
```

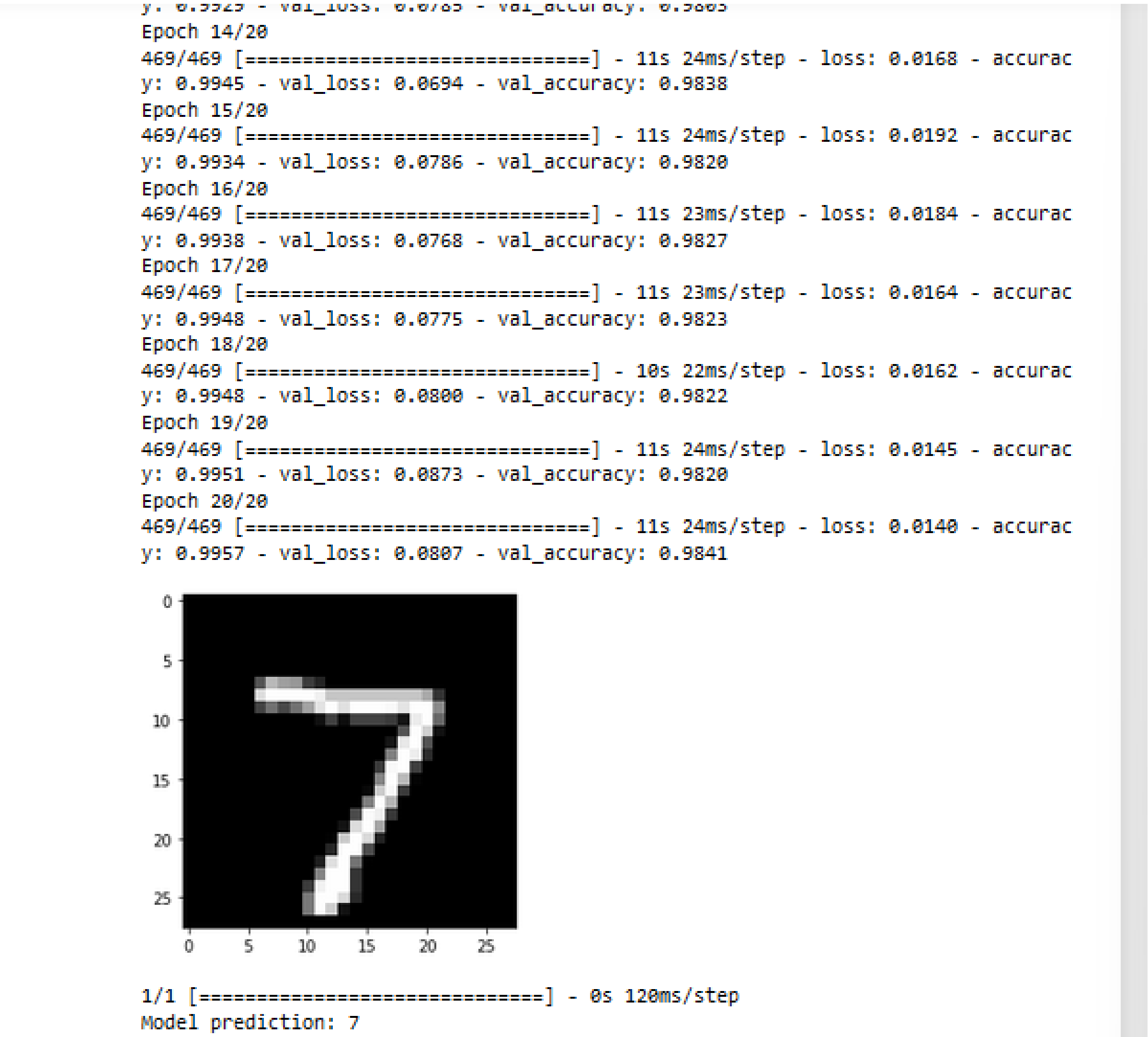


```
Epoch 15/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0190 - accurac
y: 0.9934 - val_loss: 0.0660 - val_accuracy: 0.9829
Epoch 16/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0176 - accurac
y: 0.9942 - val_loss: 0.0729 - val_accuracy: 0.9834
Epoch 17/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0165 - accurac
y: 0.9948 - val_loss: 0.0690 - val_accuracy: 0.9841
Epoch 18/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0164 - accurac
y: 0.9945 - val_loss: 0.0704 - val_accuracy: 0.9843
Epoch 19/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0164 - accurac
y: 0.9949 - val_loss: 0.0734 - val_accuracy: 0.9835
Epoch 20/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0131 - accurac
y: 0.9958 - val_loss: 0.0752 - val_accuracy: 0.9844
```



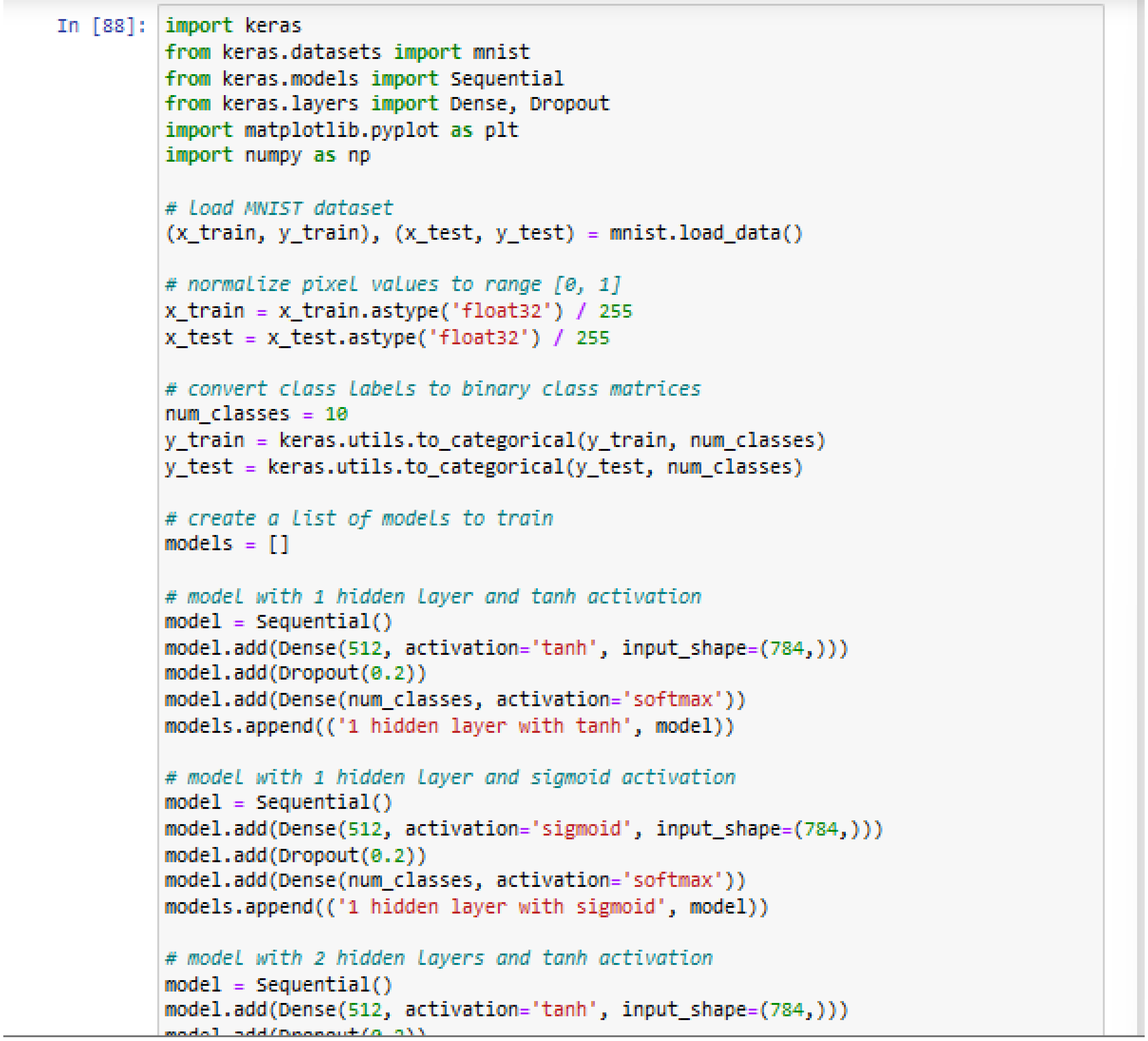
In [85]:

```
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=['accurac

# train the model
model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 7
          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
```



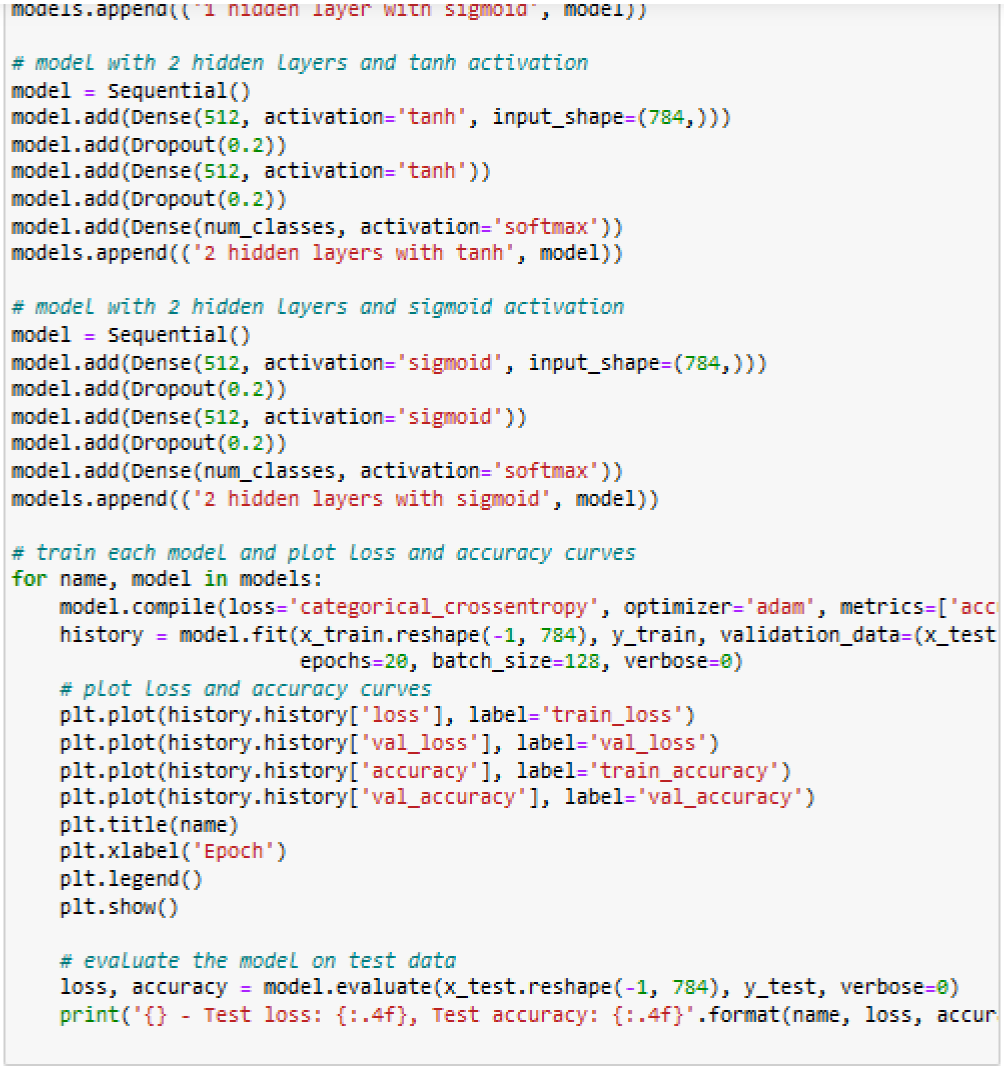
```
y: 0.9929 - val_loss: 0.0785 - val_accuracy: 0.9805
Epoch 14/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0168 - accurac
y: 0.9945 - val_loss: 0.0694 - val_accuracy: 0.9838
Epoch 15/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0192 - accurac
y: 0.9934 - val_loss: 0.0786 - val_accuracy: 0.9820
Epoch 16/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0184 - accurac
y: 0.9938 - val_loss: 0.0768 - val_accuracy: 0.9827
Epoch 17/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0164 - accurac
y: 0.9948 - val_loss: 0.0775 - val_accuracy: 0.9823
Epoch 18/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0162 - accurac
y: 0.9948 - val_loss: 0.0800 - val_accuracy: 0.9822
Epoch 19/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0145 - accurac
y: 0.9951 - val_loss: 0.0873 - val_accuracy: 0.9820
Epoch 20/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0140 - accurac
y: 0.9957 - val_loss: 0.0807 - val_accuracy: 0.9841
```

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

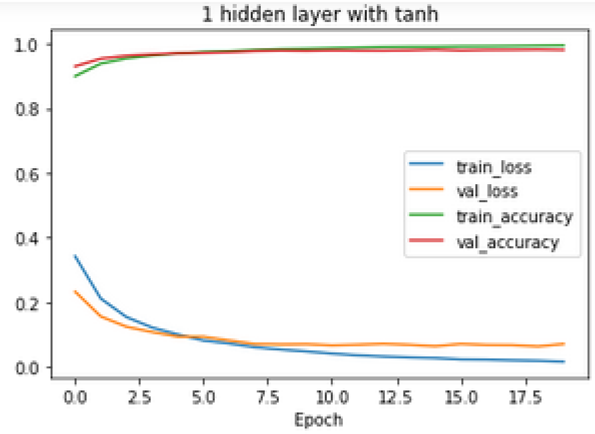


In [88]:

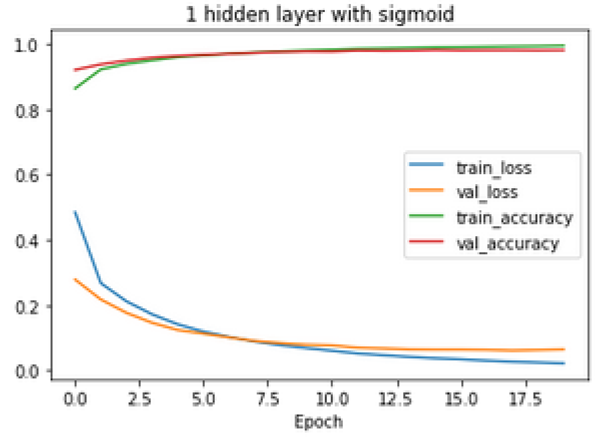
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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))
```



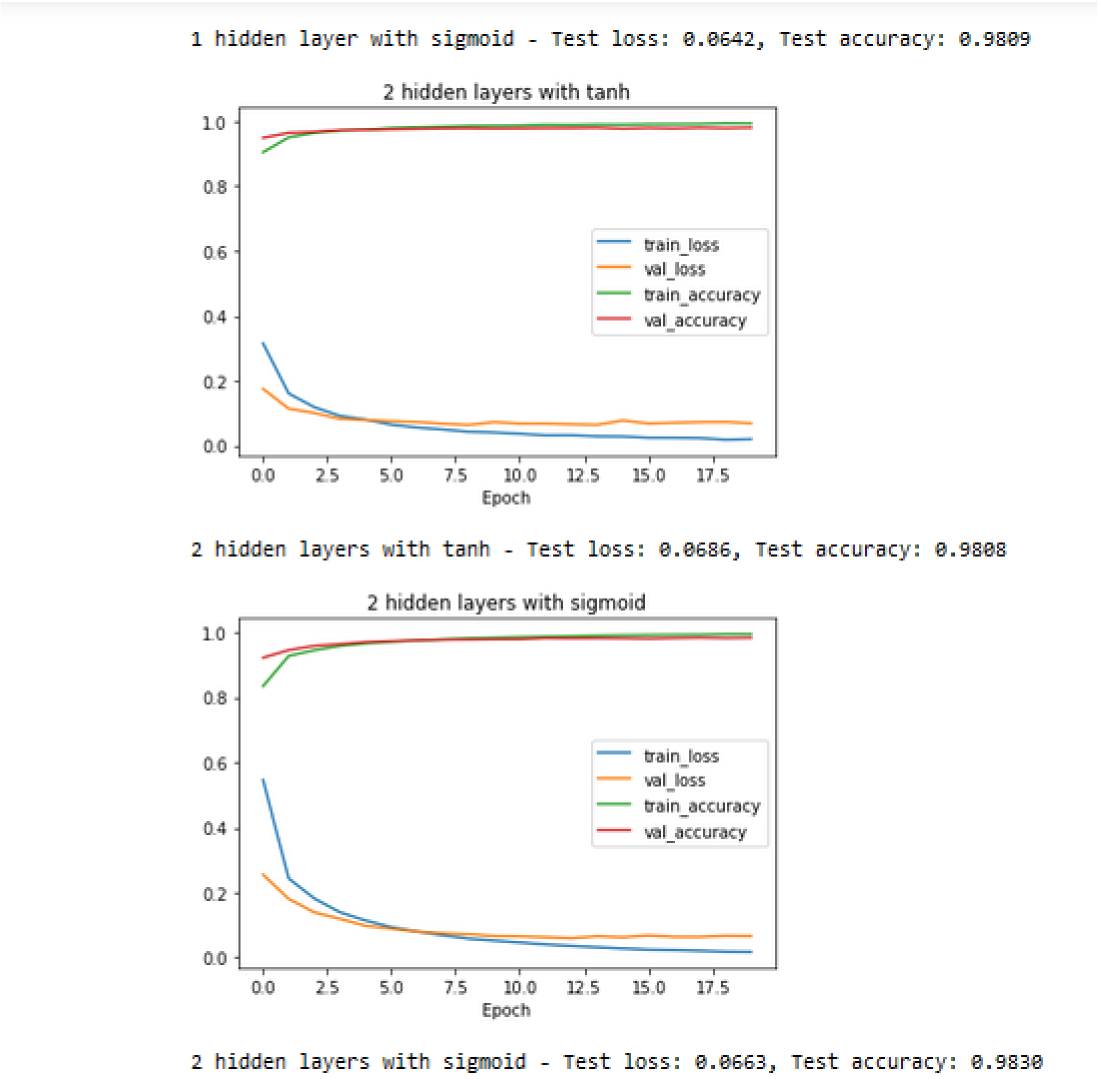
```
models.append(('1 hidden layer with sigmoid', model))

# 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=['acc
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_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, accur
```



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



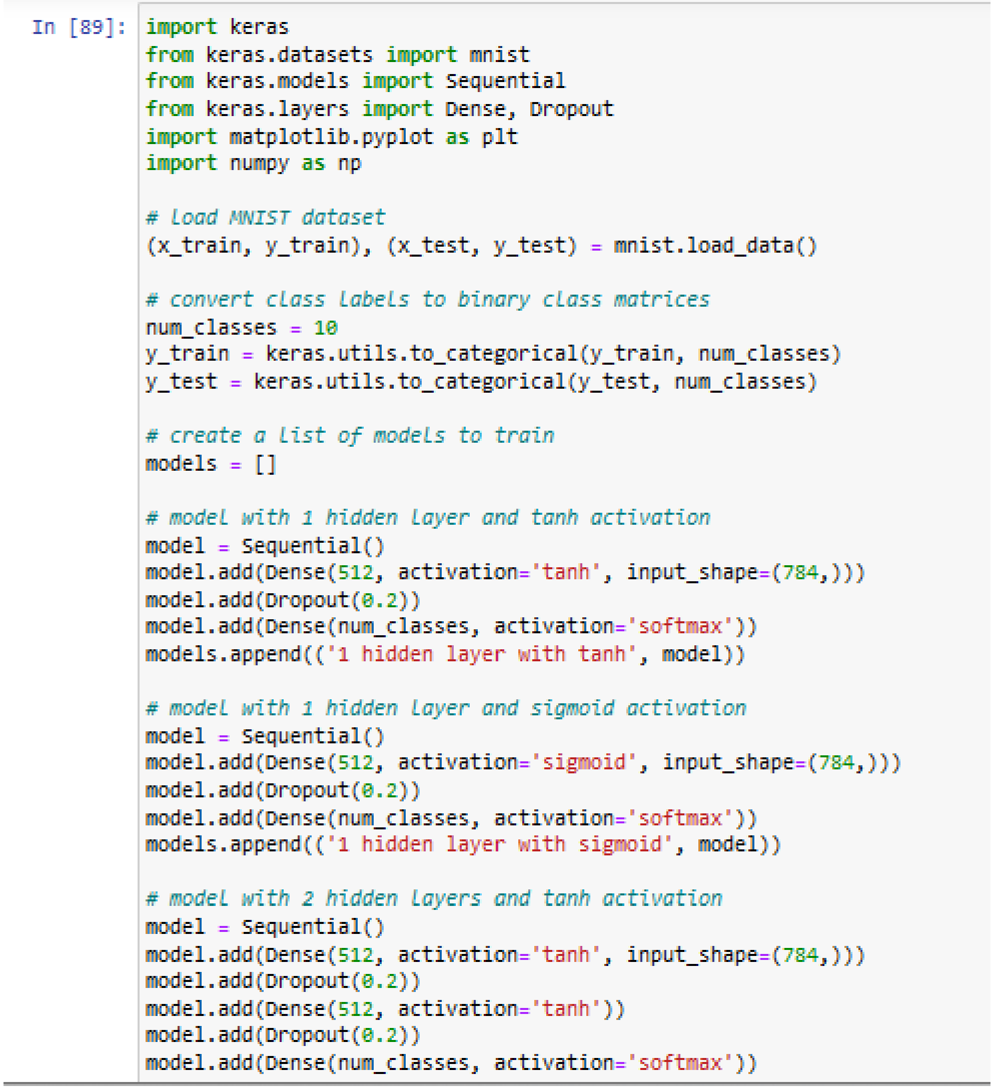
```
1 hidden layer with sigmoid - Test loss: 0.0642, Test accuracy: 0.9809
```



```
1 hidden layer with sigmoid - Test loss: 0.0642, Test accuracy: 0.9809
```

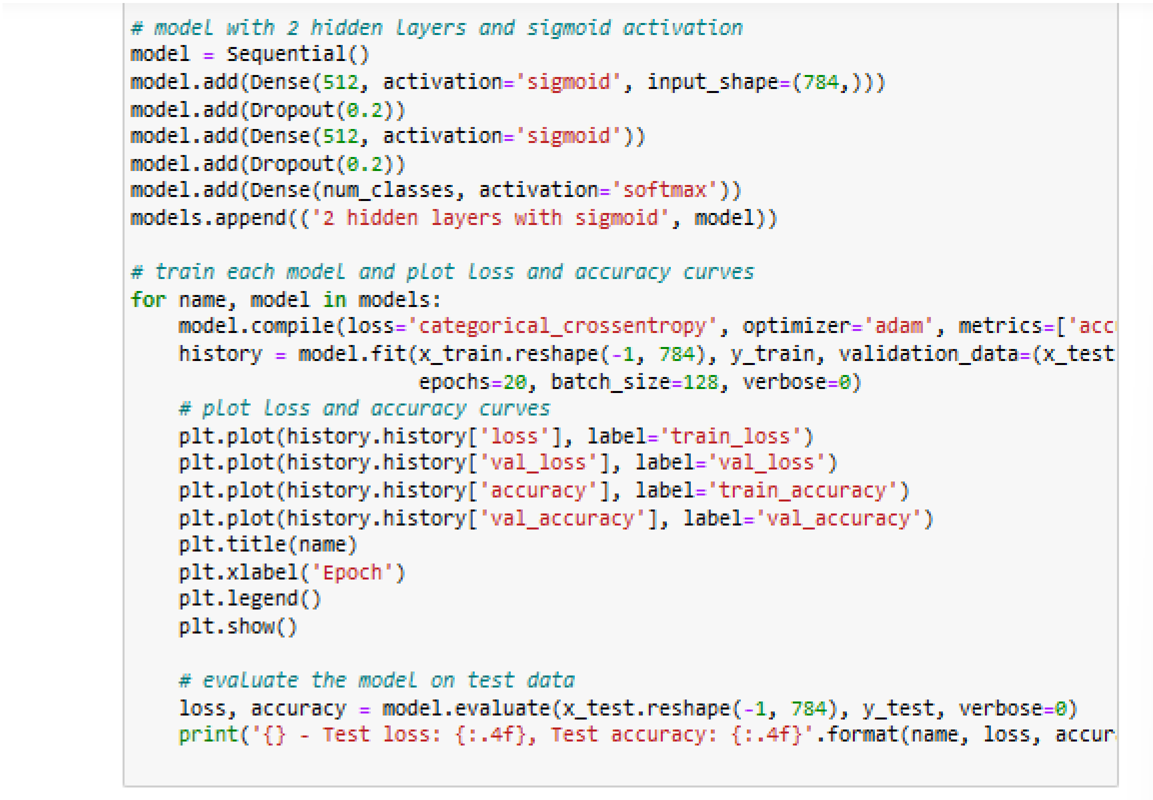
```
2 hidden layers with tanh - Test loss: 0.0686, Test accuracy: 0.9808
```

```
2 hidden layers with sigmoid - Test loss: 0.0663, Test accuracy: 0.9830
```

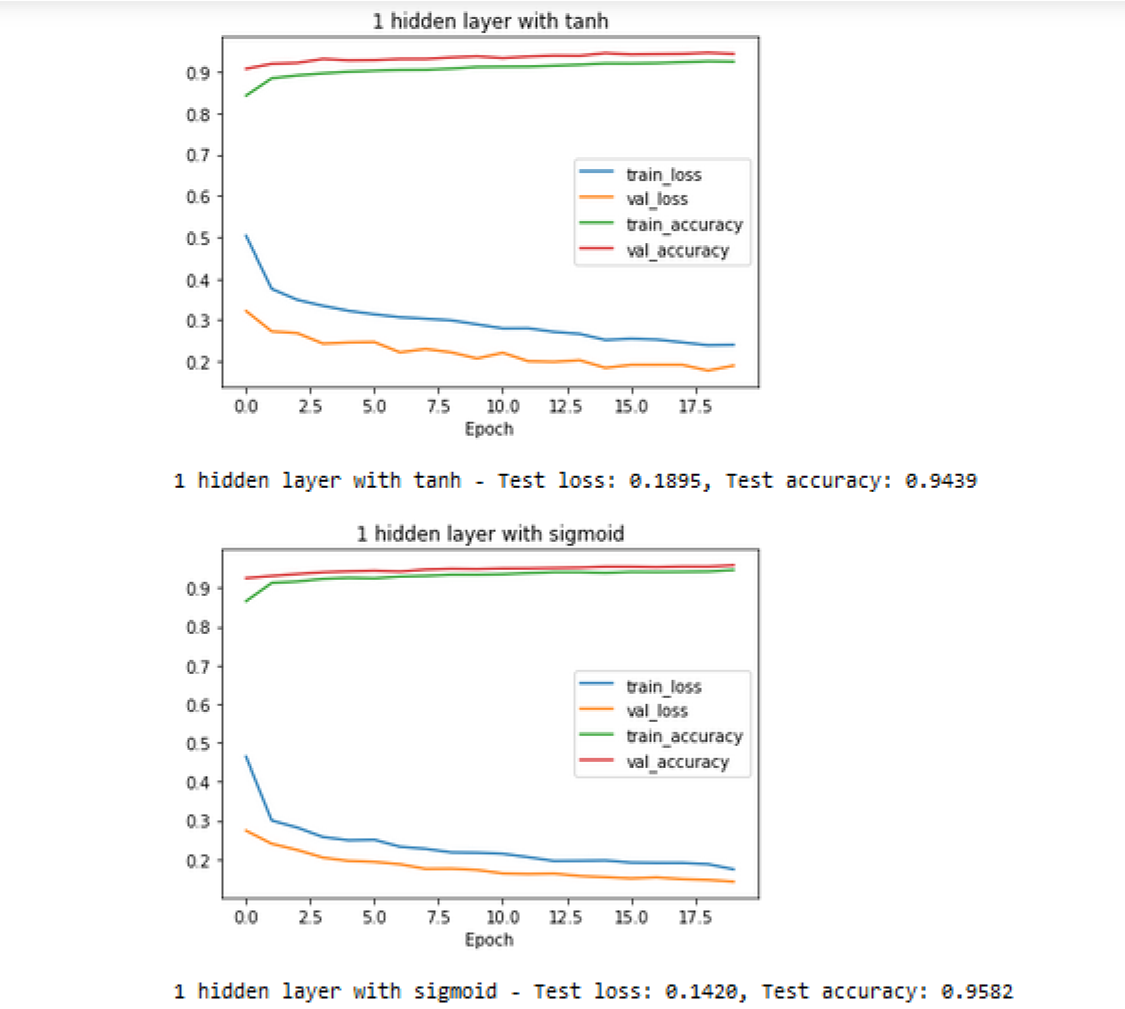


In [89]:

```
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))

# 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'))
```

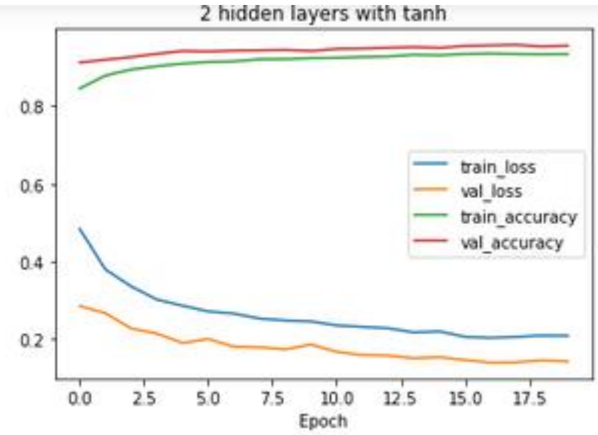


```
# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 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=['acc
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_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, accur
```

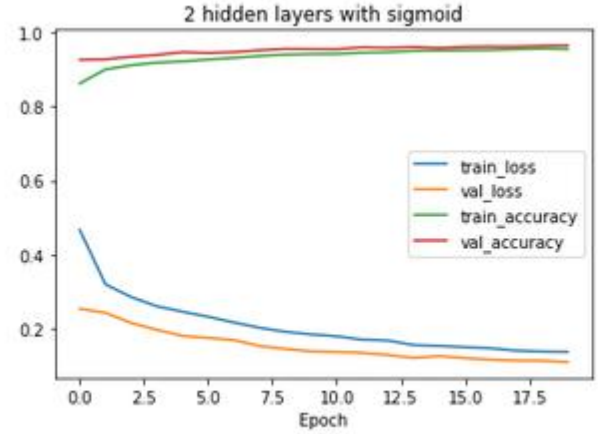


```
1 hidden layer with tanh - Test loss: 0.1895, Test accuracy: 0.9439
```

```
1 hidden layer with sigmoid - Test loss: 0.1420, Test accuracy: 0.9582
```



```
2 hidden layers with tanh - Test loss: 0.1422, Test accuracy: 0.9563
```



```
2 hidden layers with sigmoid - Test loss: 0.1095, Test accuracy: 0.9652
```