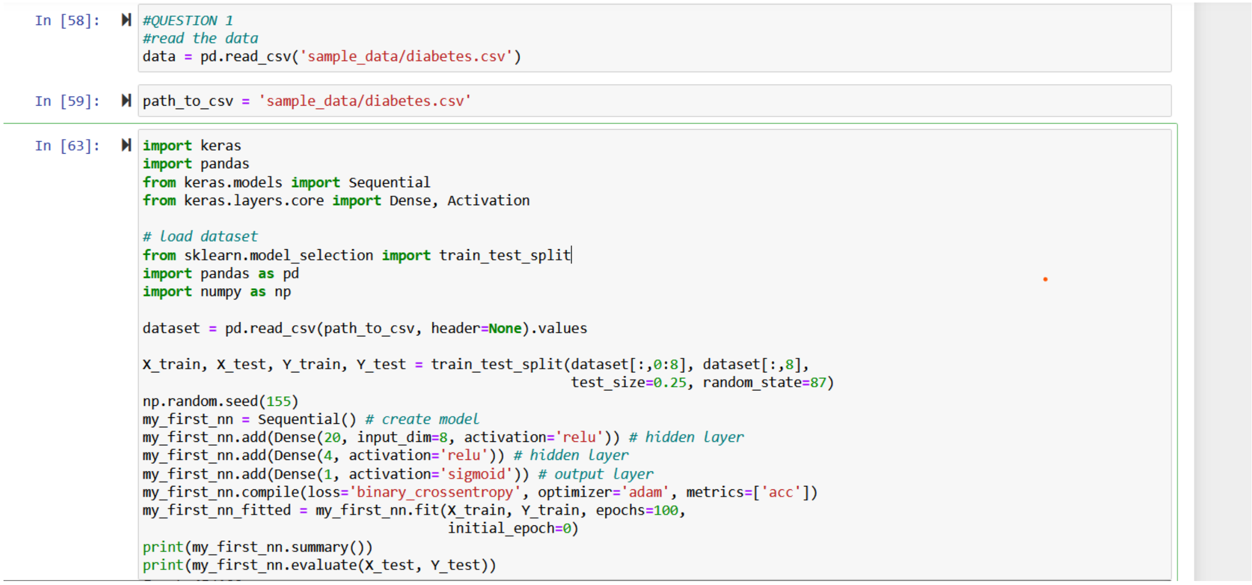
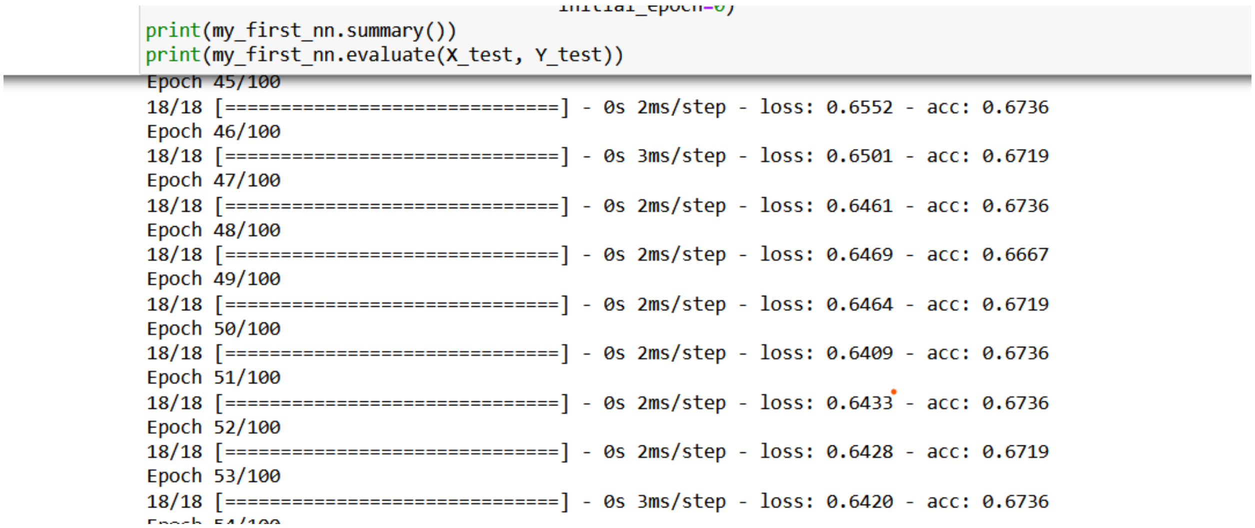
PRIYANKA GUNIGANTI 700747212

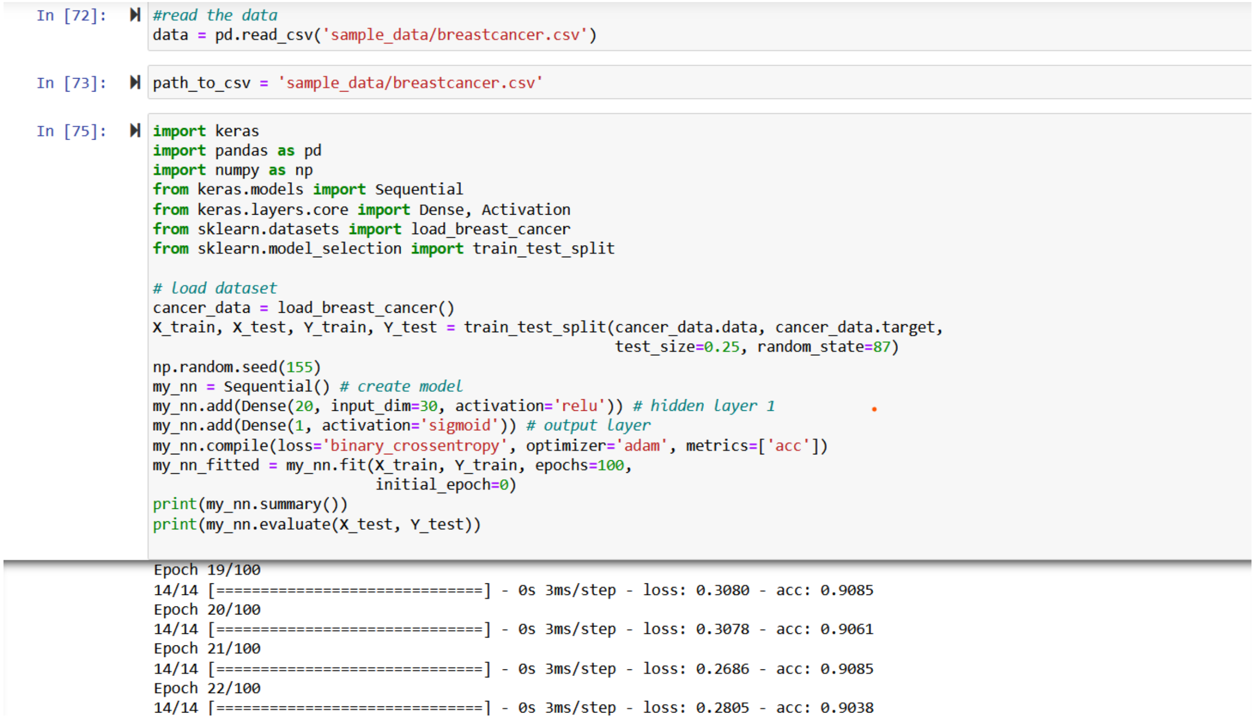


```
In [58]:  #QUESTION 1
          #read the data
          data = pd.read_csv('sample_data/diabetes.csv')

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))
```

I have plotted the loss and accuracy for both training data and validation data using the history object in the source code.

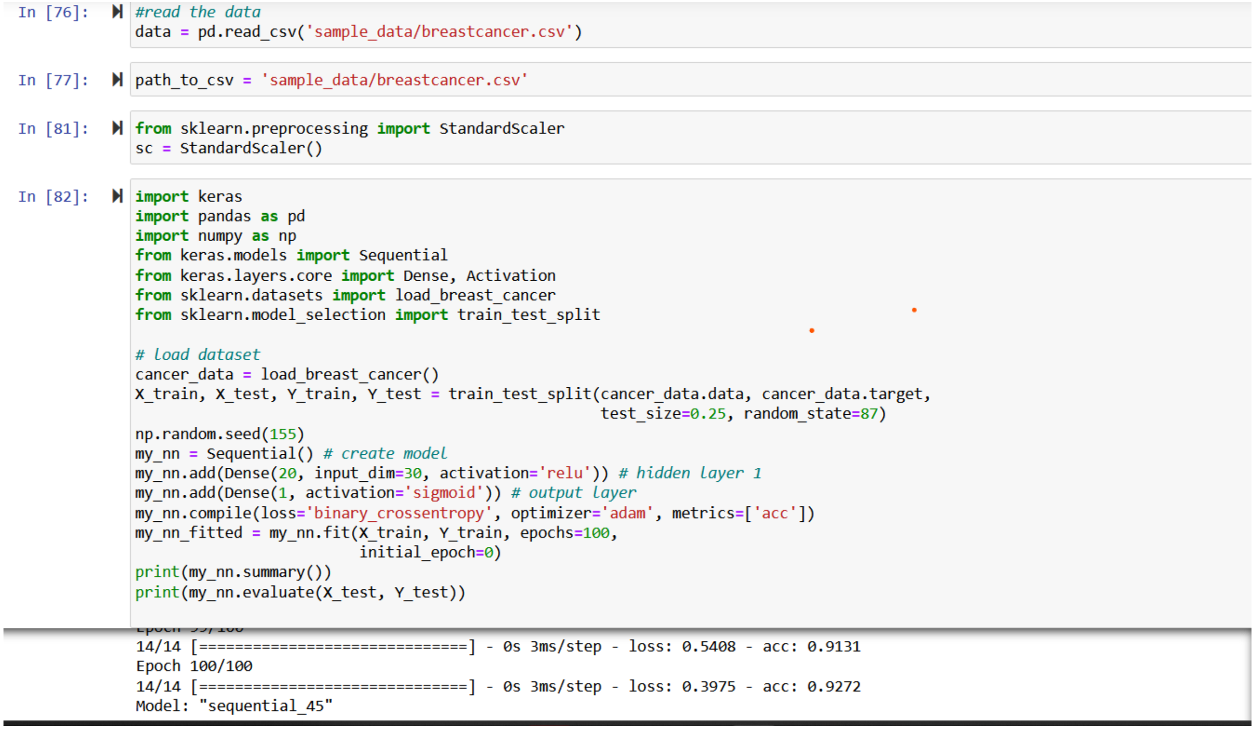


```
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test))
Epoch 45/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6552 - acc: 0.6736
Epoch 46/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6501 - acc: 0.6719
Epoch 47/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6461 - acc: 0.6736
Epoch 48/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6469 - acc: 0.6667
Epoch 49/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6464 - acc: 0.6719
Epoch 50/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6409 - acc: 0.6736
Epoch 51/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6433 - acc: 0.6736
Epoch 52/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6428 - acc: 0.6719
Epoch 53/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6420 - acc: 0.6736
Epoch 54/100
```



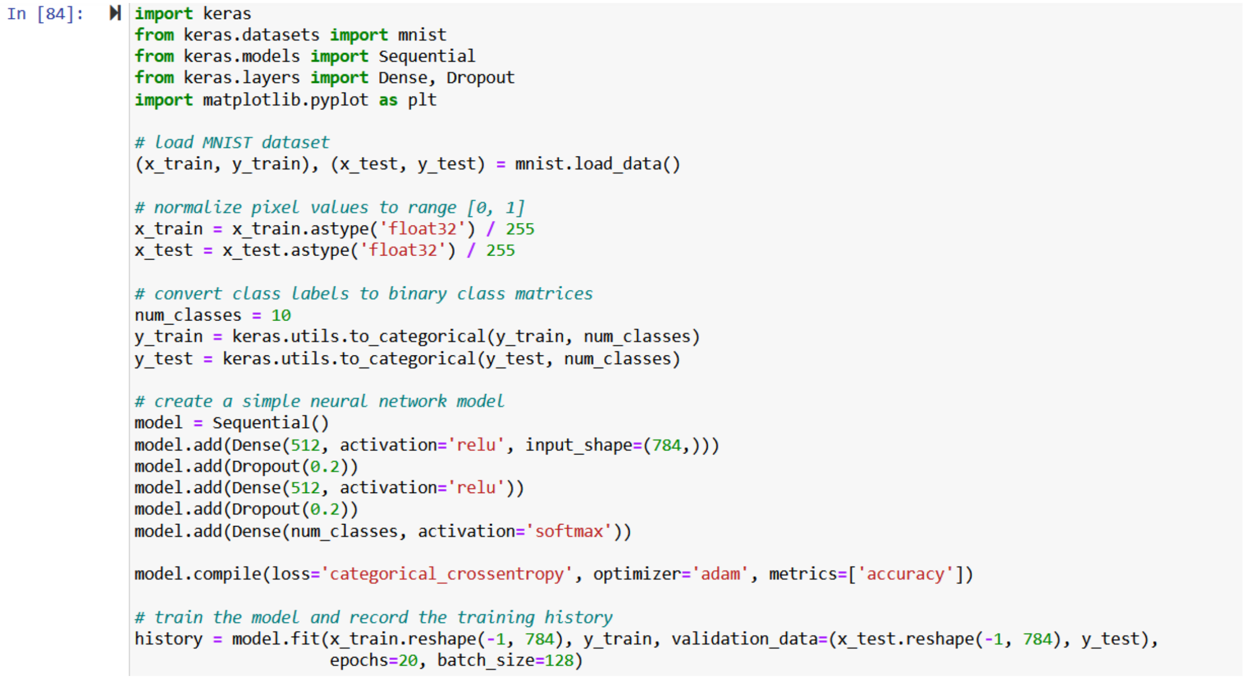
```
In [72]:  #read the data
          data = pd.read_csv('sample_data/breastcancer.csv')

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.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 3ms/step - loss: 0.3080 - acc: 0.9085
Epoch 20/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3078 - acc: 0.9061
Epoch 21/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2686 - acc: 0.9085
Epoch 22/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2805 - acc: 0.9038
```

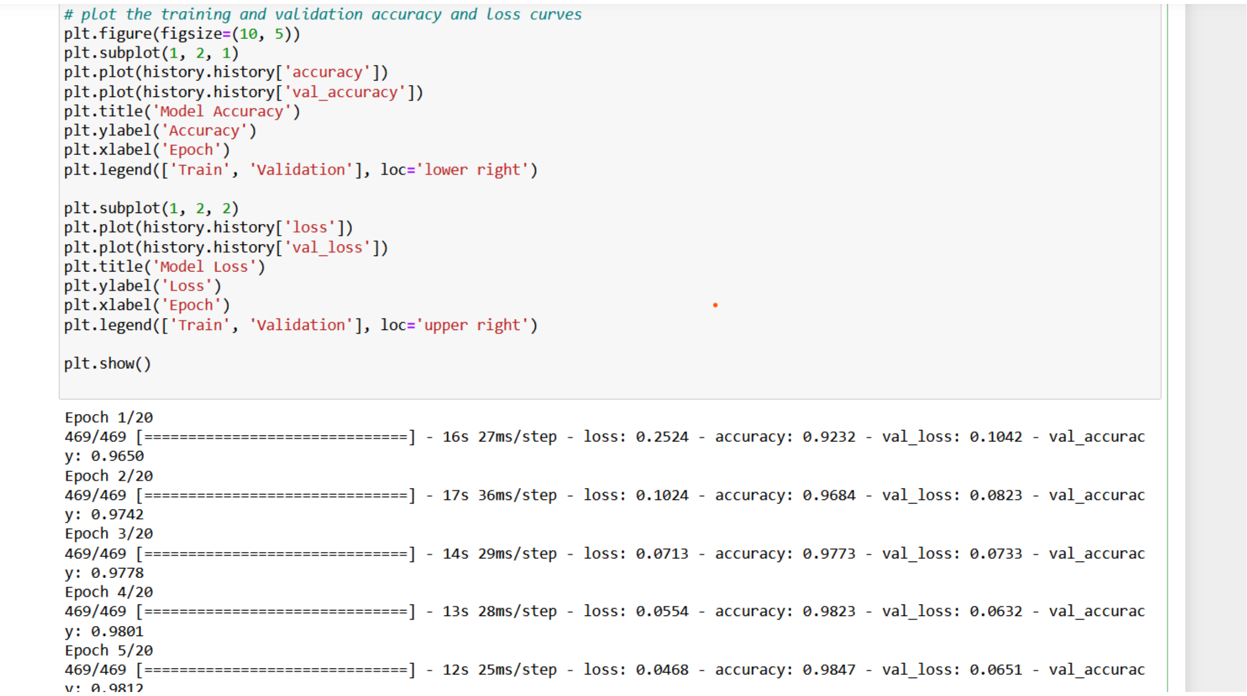


```
In [76]:  #read the data
          data = pd.read_csv('sample_data/breastcancer.csv')

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.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 99/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5408 - acc: 0.9131
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3975 - acc: 0.9272
Model: "sequential_45"
```



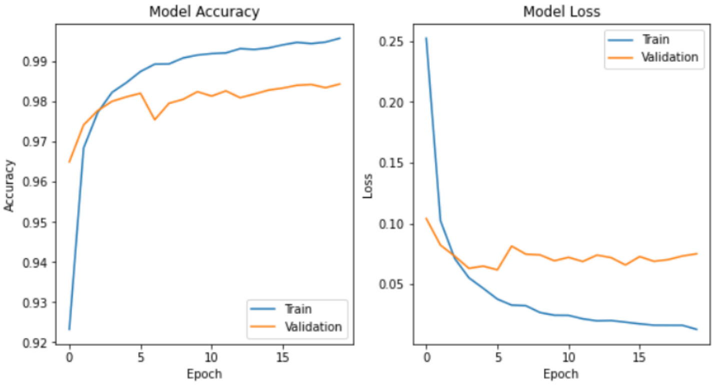
```
In [84]:  import keras
          from keras.datasets import mnist
          from keras.models import Sequential
          from keras.layers import Dense, Dropout
          import matplotlib.pyplot as plt

          # 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
          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()
```

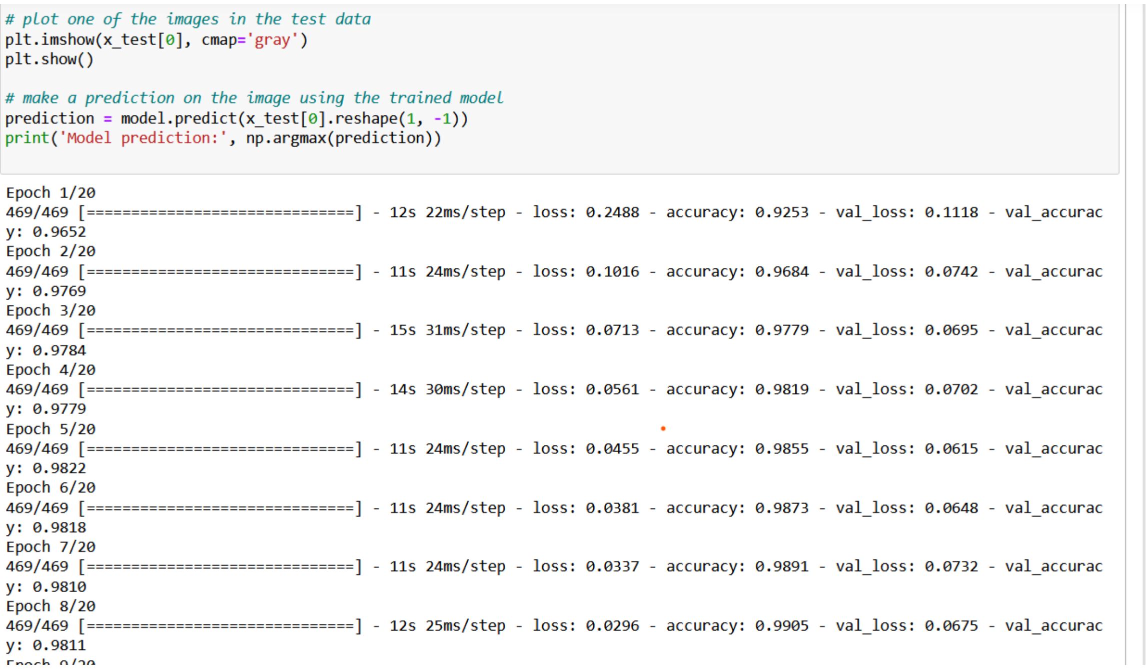
```
Epoch 1/20
469/469 [==============================] - 16s 27ms/step - loss: 0.2524 - accuracy: 0.9232 - val_loss: 0.1042 - val_accurac
y: 0.9650
Epoch 2/20
469/469 [==============================] - 17s 36ms/step - loss: 0.1024 - accuracy: 0.9684 - val_loss: 0.0823 - val_accurac
y: 0.9742
Epoch 3/20
469/469 [==============================] - 14s 29ms/step - loss: 0.0713 - accuracy: 0.9773 - val_loss: 0.0733 - val_accurac
y: 0.9778
Epoch 4/20
469/469 [==============================] - 13s 28ms/step - loss: 0.0554 - accuracy: 0.9823 - val_loss: 0.0632 - val_accurac
y: 0.9801
Epoch 5/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0468 - accuracy: 0.9847 - val_loss: 0.0651 - val_accurac
y: 0.9812
```

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



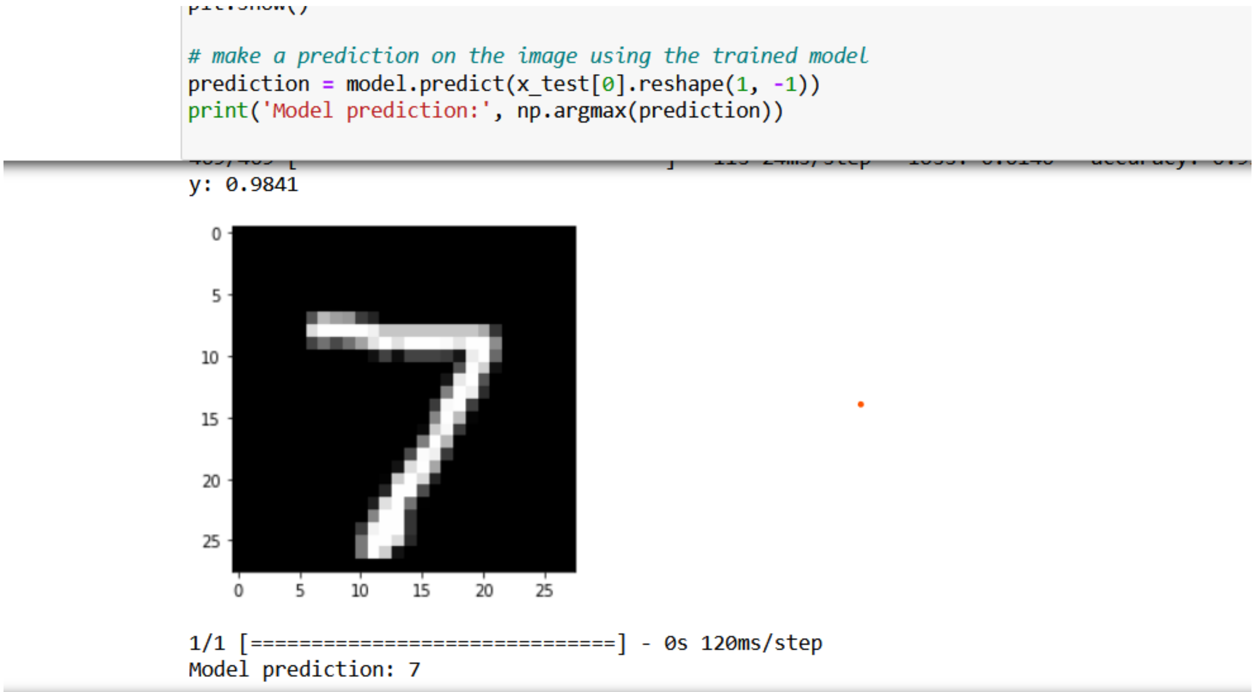
In [85]:

```python
#QUESQION 2
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))
```

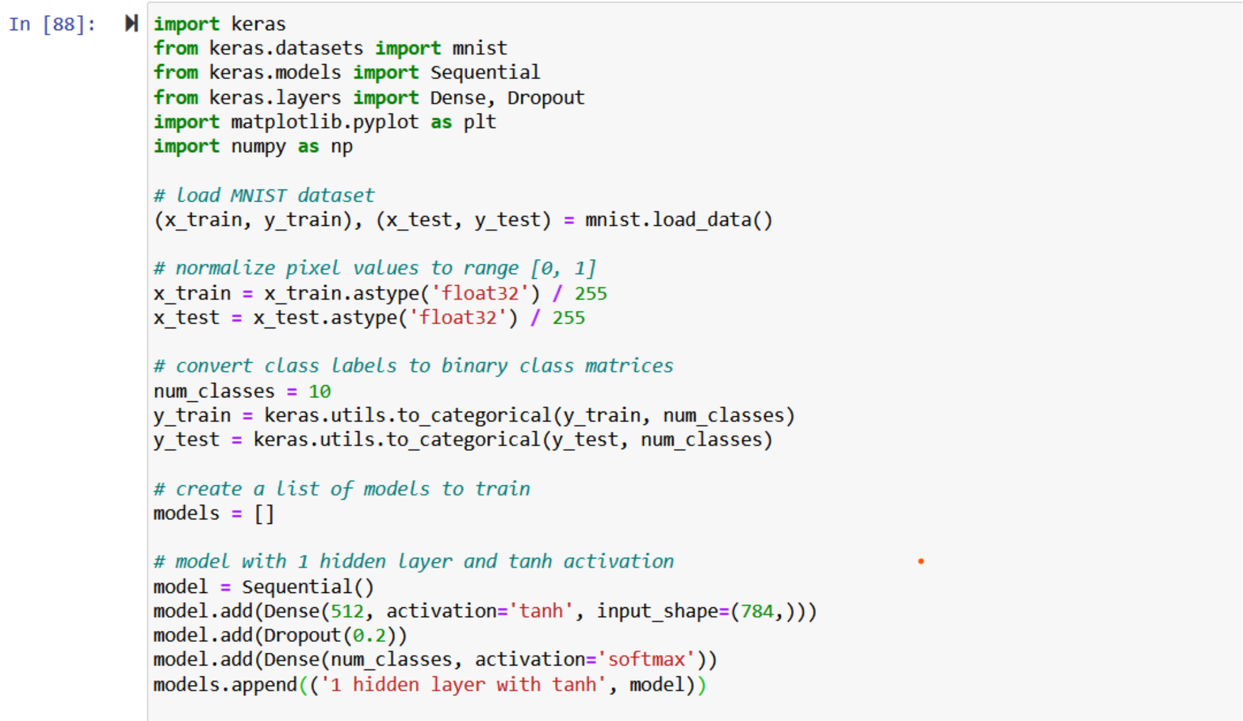
```
Epoch 1/20
469/469 [==============================] - 12s 22ms/step - loss: 0.2488 - accuracy: 0.9253 - val_loss: 0.1118 - val_accurac
y: 0.9652
Epoch 2/20
469/469 [==============================] - 11s 24ms/step - loss: 0.1016 - accuracy: 0.9684 - val_loss: 0.0742 - val_accurac
y: 0.9769
Epoch 3/20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



```
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))
```

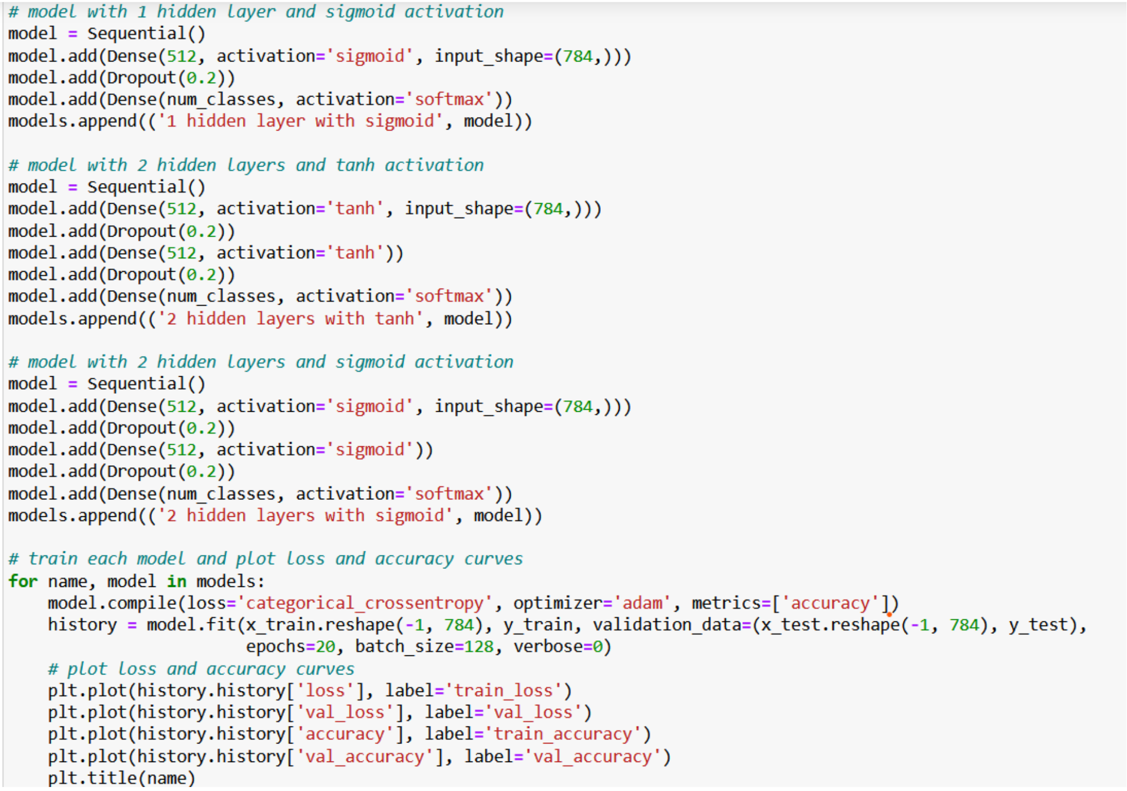
```
y: 0.9841
```

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



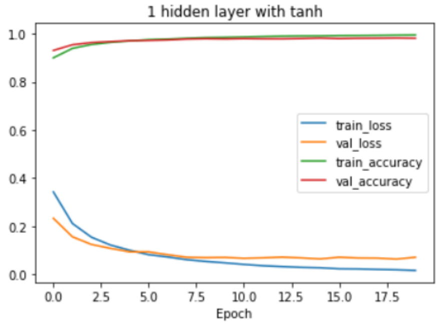
In [88]:

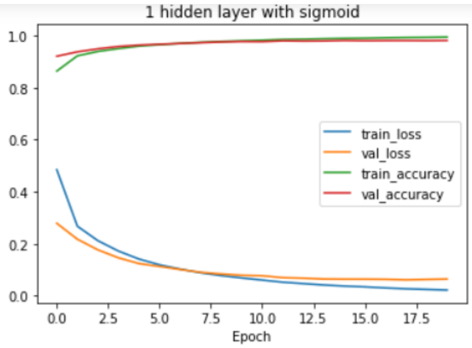
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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))
```



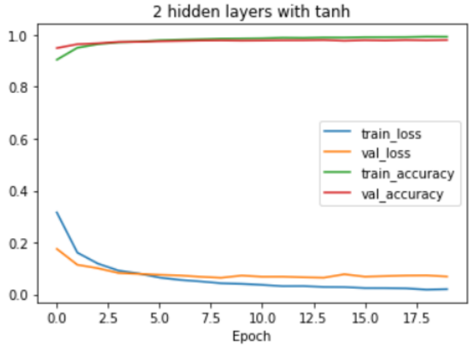
```
# model with 1 hidden layer and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))

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

```python
# 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))
```

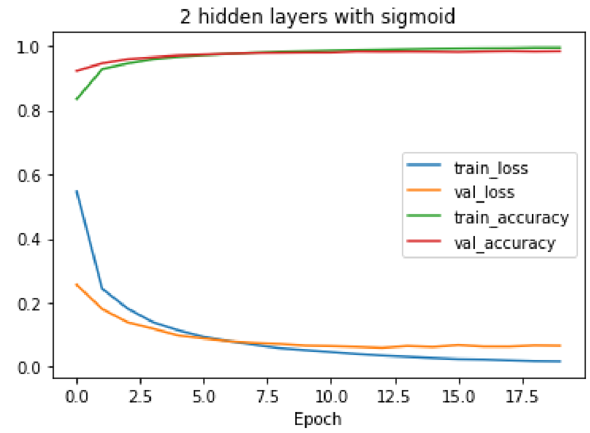




```
1 hidden layer with sigmoid - Test loss: 0.0642, Test accuracy: 0.9809
```



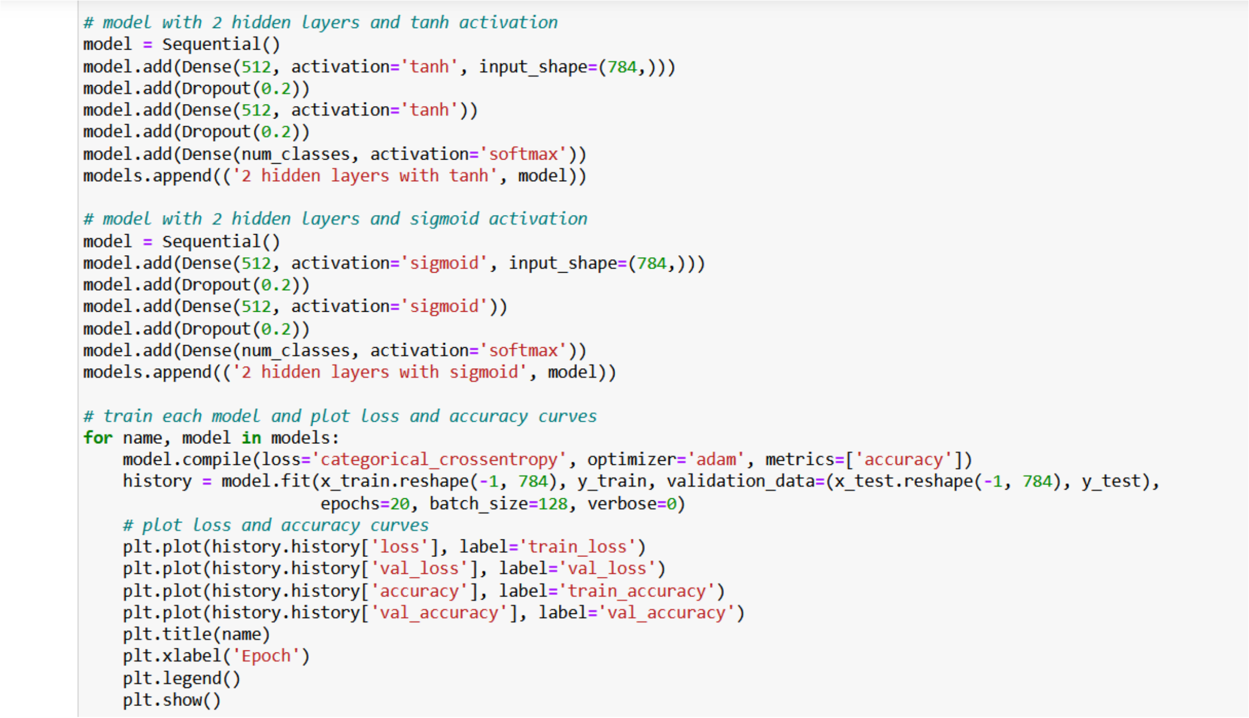
```
2 hidden lavers with tanh - Test loss: 0.0686. Test accuracv: 0.9808
```



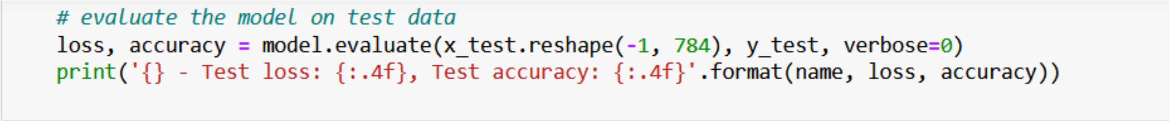
```
2 hidden layers with sigmoid - Test loss: 0.0663, Test accuracy: 0.9830
```

In [89]:

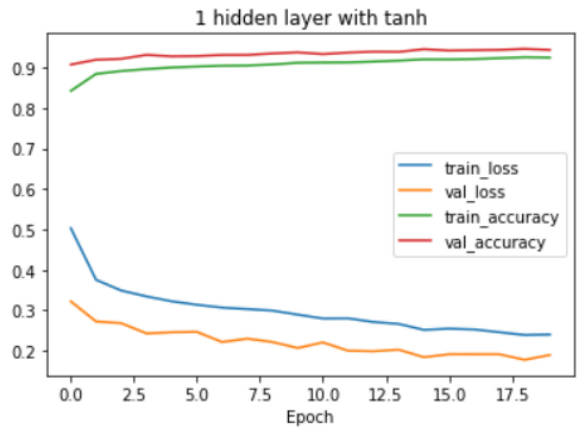
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

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

# 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()
```

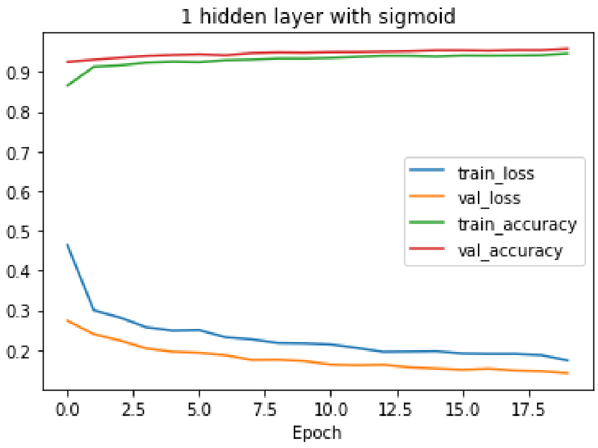


```
    # evaluate the model on test data
    loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```

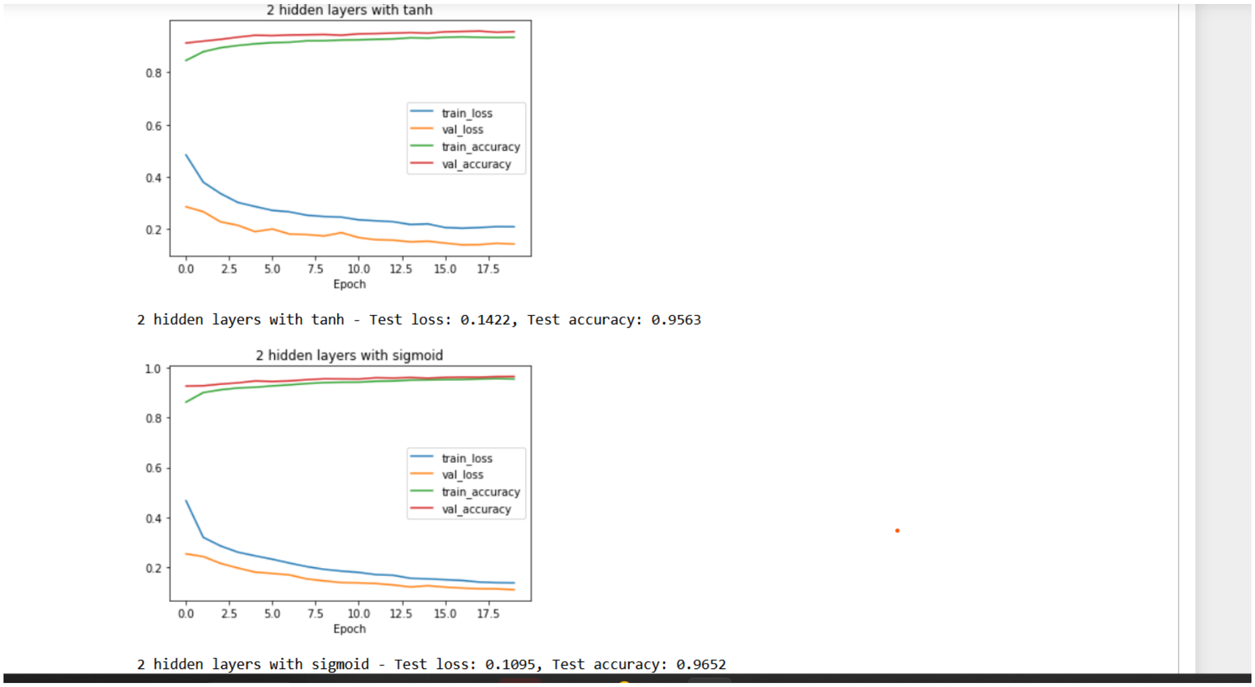


1 hidden layer with tanh

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

When we check the performance, it shows better accuracy when there is scaling than when scaling is removed.

My github link:

https://github.com/priyanka-minni/NNDL_assignment2