Assignment – 2

Name: Krishna Vamsi Koppula
Student ID: 700745021

Video Link: https://drive.google.com/file/d/1LiVOneeHzeQ8Ta49MdXqNnQ0h7gWQmFF/view?usp=sharing

GitHub Link: https://github.com/Krishnavamsikoppula/Assignment_2_NNDL_Summer

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

```
[31] #read the data
     data = pd.read_csv('sample_data/A2_sample_data/diabetes.csv')
```

```
[ ]
```

```
[32] path_to_csv = 'sample_data/A2_sample_data/diabetes.csv'
```

```
     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))
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_53 (Dense)            (None, 20)                180

 dense_54 (Dense)            (None, 4)                 84

 dense_55 (Dense)            (None, 1)                 5

=================================================================
Total params: 269
Trainable params: 269
Non-trainable params: 0

_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.6667 - acc: 0.6198
[0.6667177677154541, 0.6197916865348816]
```

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

```
[34] #read the data
     data = pd.read_csv('sample_data/A2_sample_data/breastcancer.csv')
```

```
[35] path_to_csv = 'sample_data/A2_sample_data/breastcancer.csv'
```

```
[16] 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))
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_13 (Dense)            (None, 20)                620

 dense_14 (Dense)            (None, 1)                 21

=================================================================
Total params: 641
Trainable params: 641
Non-trainable params: 0
_________________________________________________________________
None
5/5 [==============================] - 0s 8ms/step - loss: 0.3764 - acc: 0.8741
[0.37635719776153564, 0.8741258978843689]
```

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()

```
[36] #read the data
     data = pd.read_csv('sample_data/A2_sample_data/breastcancer.csv')
```

```
[21] path_to_csv = 'sample_data/A2_sample_data/breastcancer.csv'
```

```
[22] from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
```

```
[23] 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))
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 20)                620

 dense_16 (Dense)            (None, 1)                 21

=================================================================
Total params: 641
Trainable params: 641
Non-trainable params: 0
_________________________________________________________________
None
5/5 [==============================] - 0s 4ms/step - loss: 0.2609 - acc: 0.9161
[0.26089033484458923, 0.9160839319229126]
```

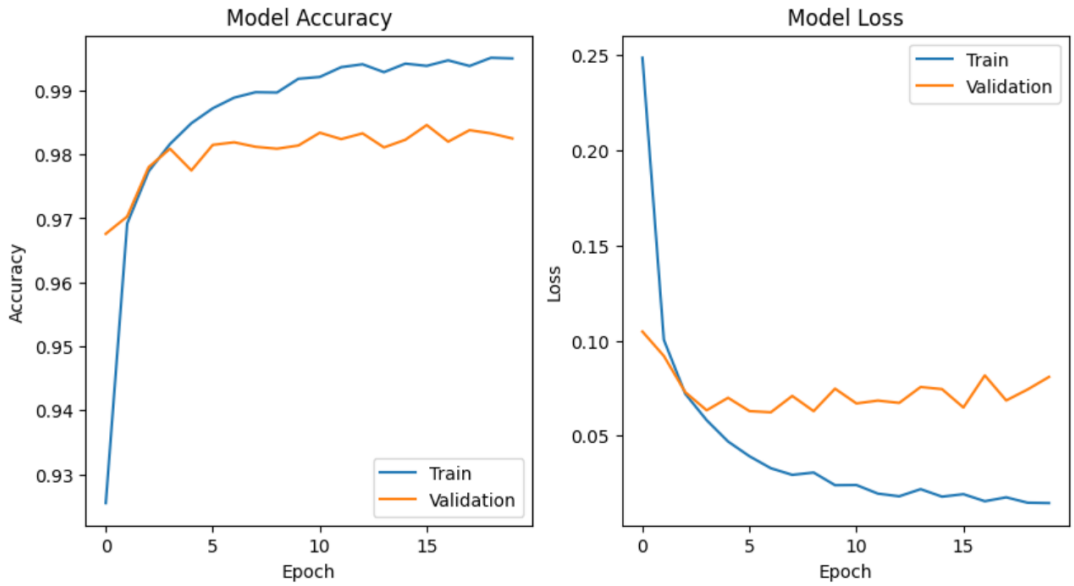
Problem2

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

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()
```

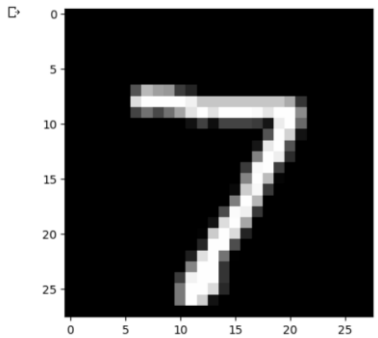
Output


Model Accuracy
Accuracy
Epoch
Train
Validation
Model Loss
Loss
Epoch
Train
Validation

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))
```

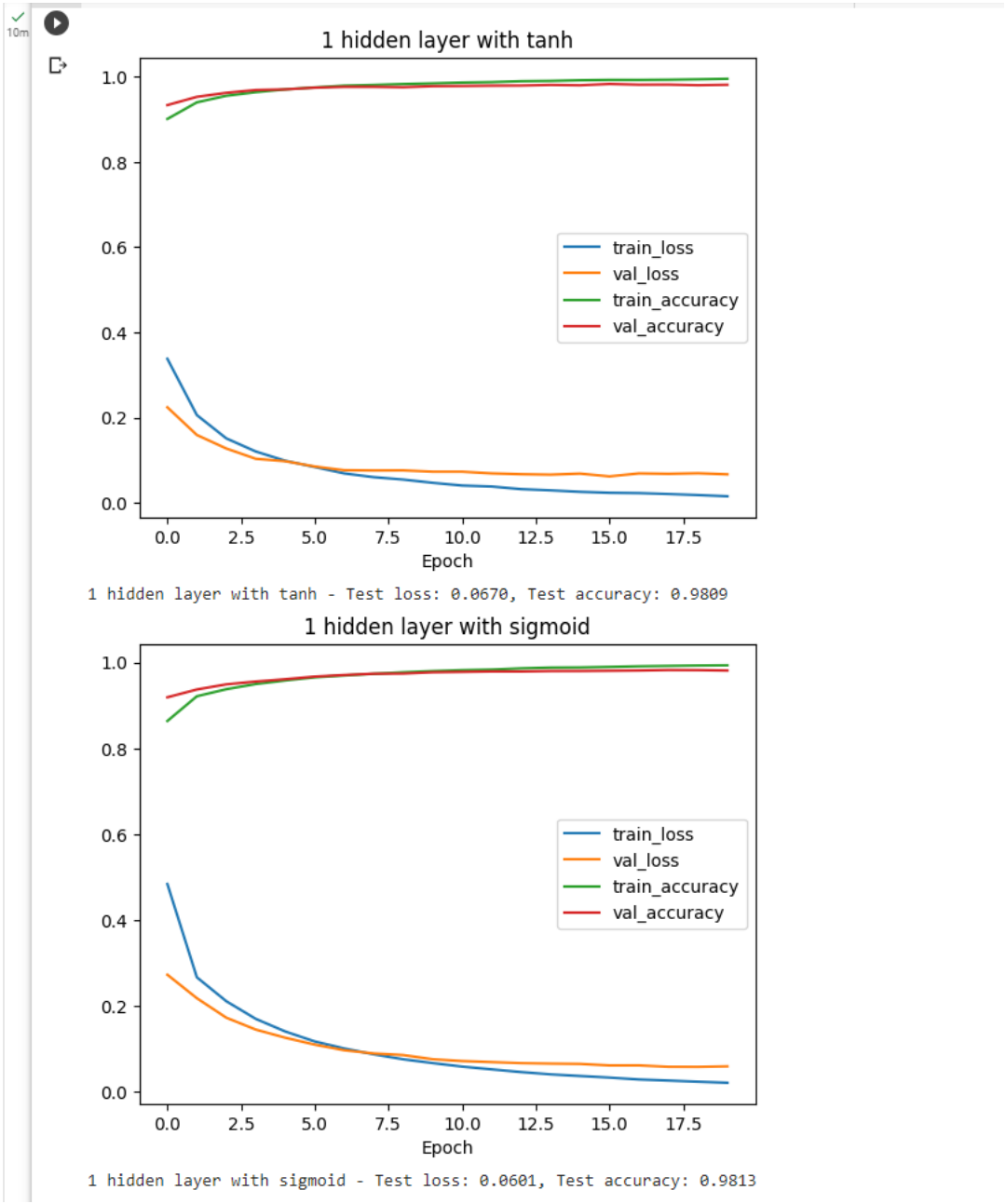
Output

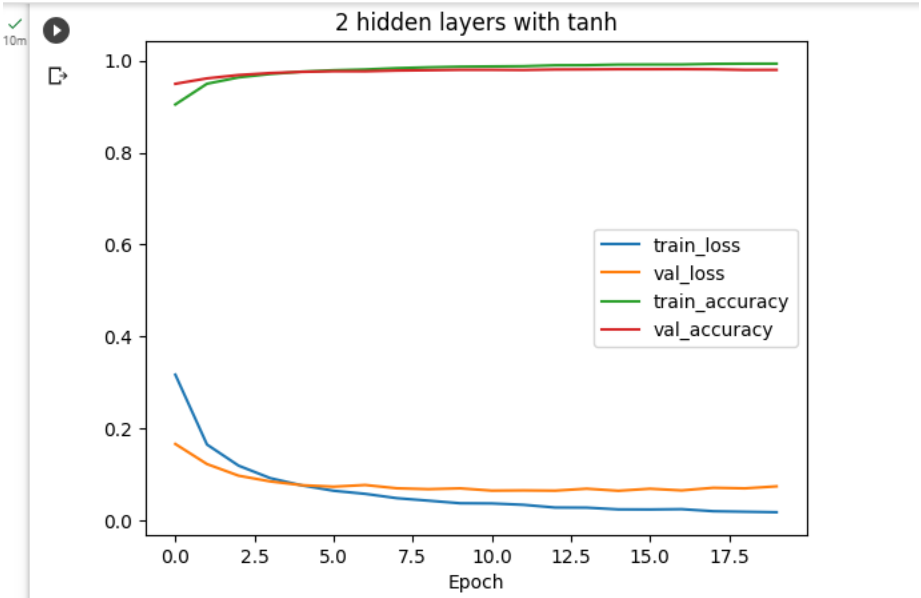


3. We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.

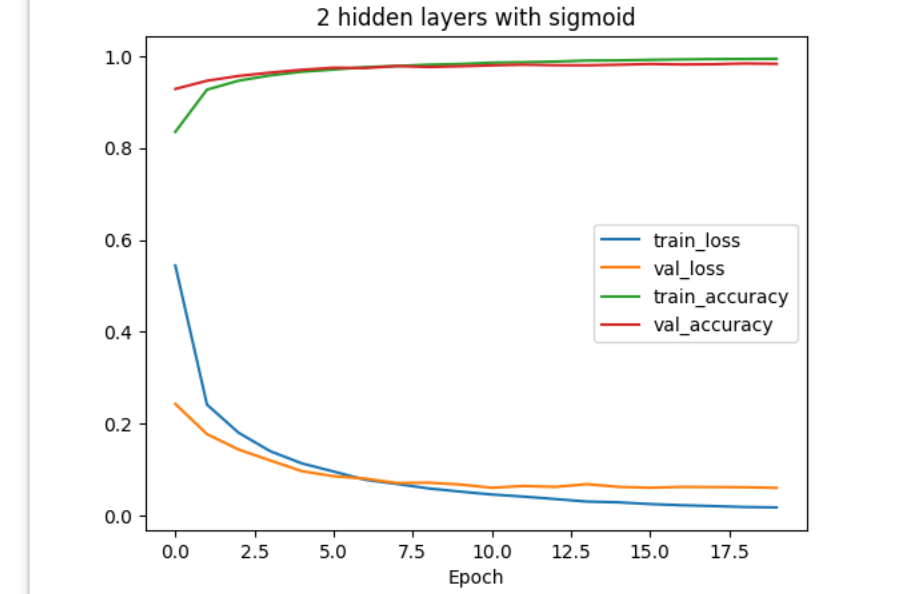
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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()

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

Output


1 hidden layer with tanh
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
train_loss
val_loss
train_accuracy
val_accuracy
1 hidden layer with tanh - Test loss: 0.0670, Test accuracy: 0.9809
1 hidden layer with sigmoid
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
train_loss
val_loss
train_accuracy
val_accuracy
1 hidden layer with sigmoid - Test loss: 0.0601, Test accuracy: 0.9813




```
2 hidden layers with tanh - Test loss: 0.0744, Test accuracy: 0.9800
```

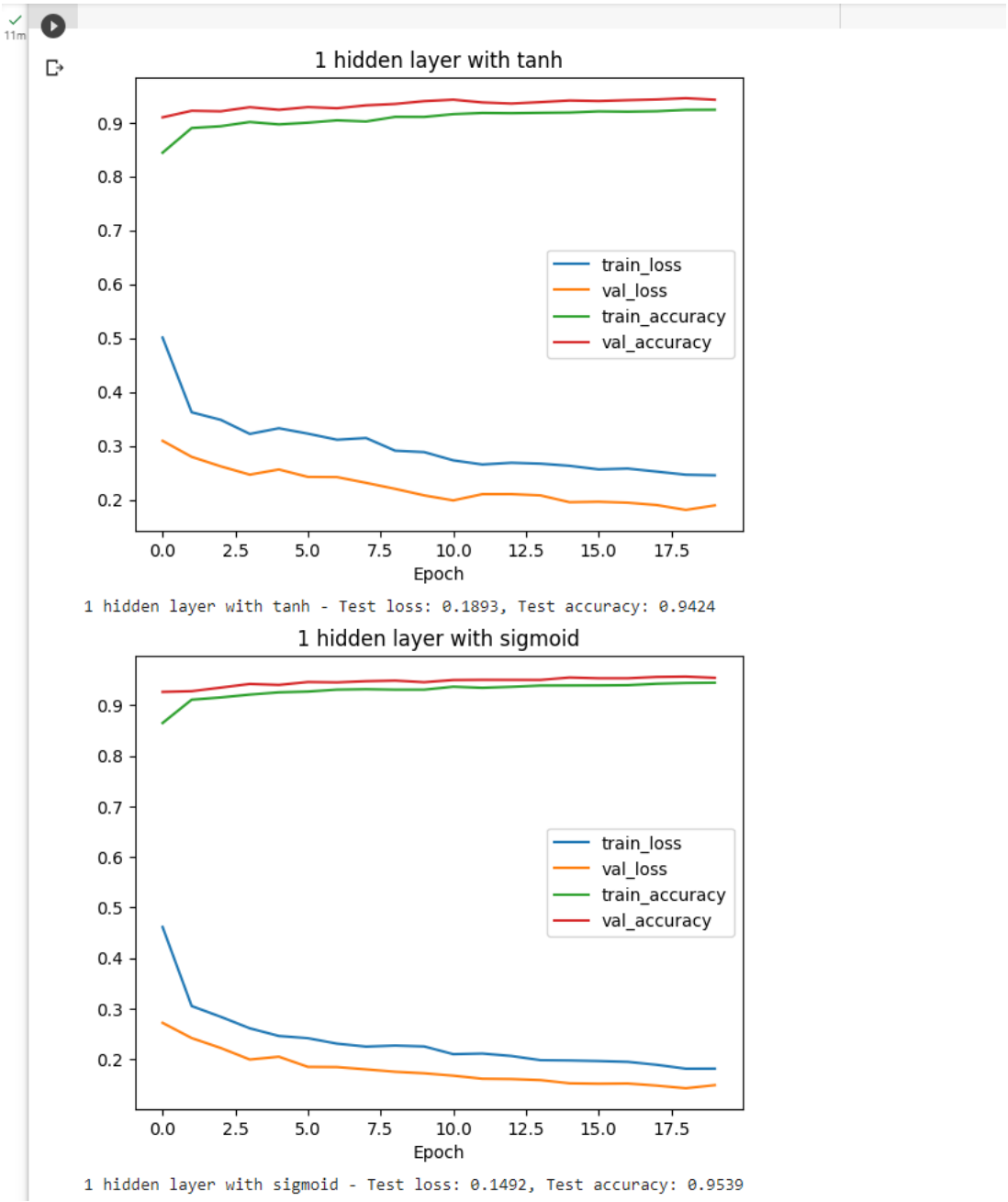


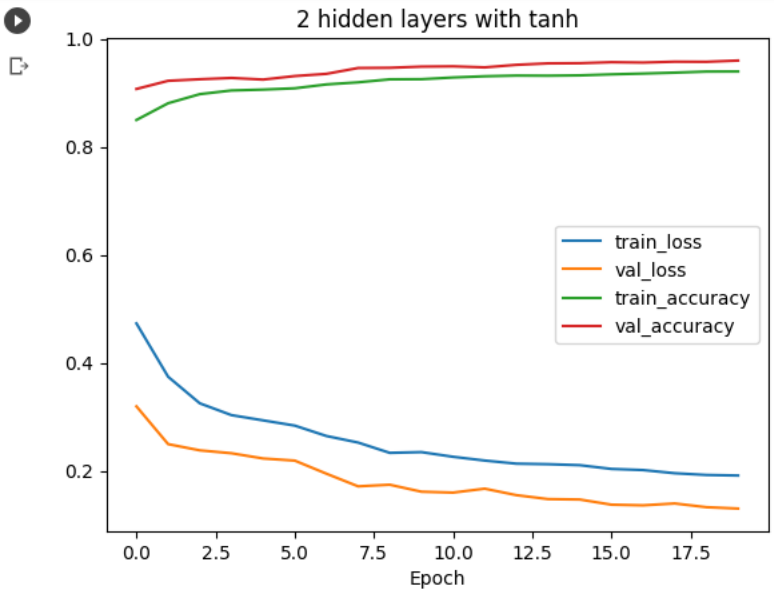

```
2 hidden layers with sigmoid - Test loss: 0.0606, Test accuracy: 0.9830
```

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

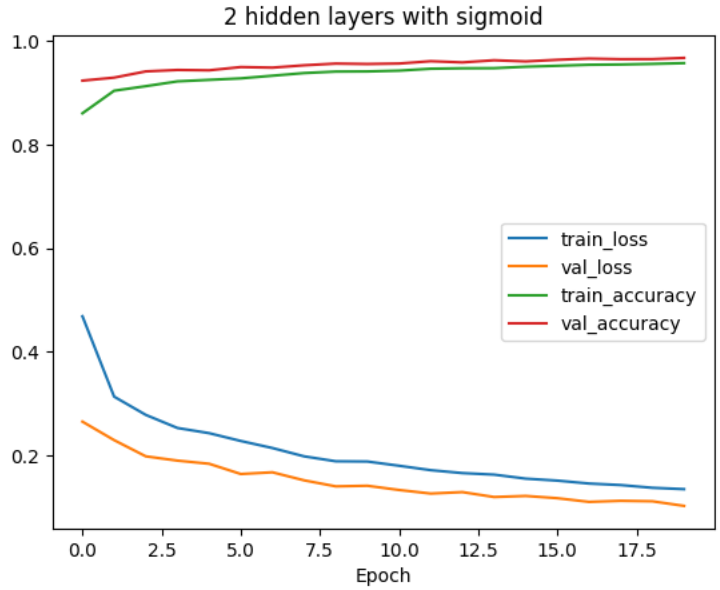
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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,)))
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))
```

Output


11m
1 hidden layer with tanh
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
train_loss
val_loss
train_accuracy
val_accuracy
1 hidden layer with tanh - Test loss: 0.1893, Test accuracy: 0.9424
1 hidden layer with sigmoid
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
train_loss
val_loss
train_accuracy
val_accuracy
1 hidden layer with sigmoid - Test loss: 0.1492, Test accuracy: 0.9539




```
2 hidden layers with tanh - Test loss: 0.1303, Test accuracy: 0.9600
```




```
2 hidden layers with sigmoid - Test loss: 0.1022, Test accuracy: 0.9674
```