SHRINIKA TELU

700741742

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.**

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.**

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.**

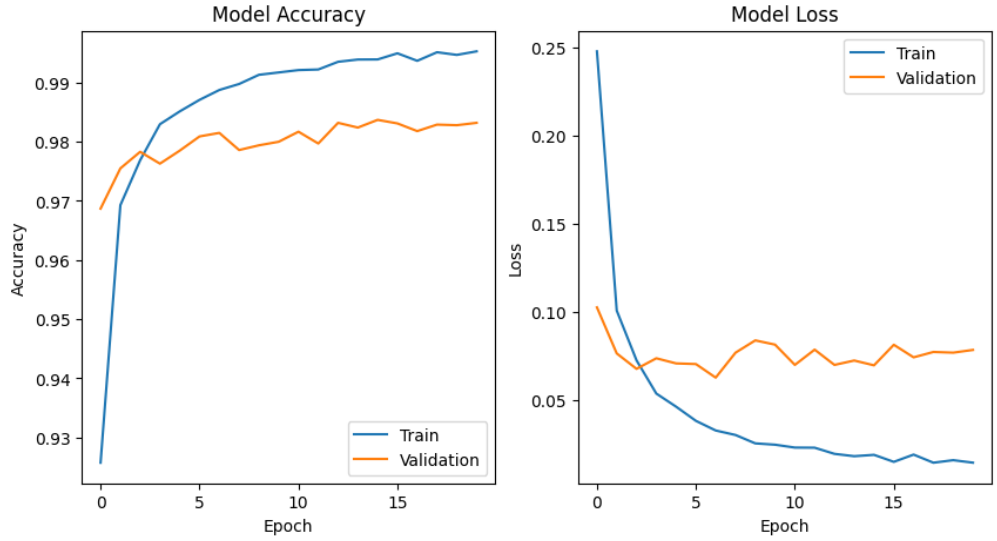
4. **Run the same code without scaling the images and check the performance?**

```python
#2 Use Image Classification on the hand written digits data set (mnist)
from google.colab import drive
drive.mount('/content/gdrive')
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical

(train_images,train_labels),(test_images, test_labels) = mnist.load_data()

print(train_images.shape[1:])
#process the data
#1. convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
print(dimData)
train_data = train_images.reshape(train_images.shape[0],dimData)
test_data = test_images.reshape(test_images.shape[0],dimData)

#convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')
#scale data
train_data /=255.0
test_data /=255.0
#change the labels frominteger to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

#creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

```
235/235 [==============================] - 8s 34ms/step - loss: 0.0401 - accuracy: 0.9856 - val_loss: 0.0702 - val_accuracy: 0.9777
Epoch 5/10
235/235 [==============================] - 6s 25ms/step - loss: 0.0321 - accuracy: 0.9897 - val_loss: 0.0718 - val_accuracy: 0.9796
Epoch 6/10
235/235 [==============================] - 8s 34ms/step - loss: 0.0239 - accuracy: 0.9924 - val_loss: 0.0687 - val_accuracy: 0.9811
Epoch 7/10
235/235 [==============================] - 6s 26ms/step - loss: 0.0172 - accuracy: 0.9945 - val_loss: 0.0752 - val_accuracy: 0.9798
Epoch 8/10
235/235 [==============================] - 8s 34ms/step - loss: 0.0131 - accuracy: 0.9959 - val_loss: 0.0644 - val_accuracy: 0.9838
Epoch 9/10
235/235 [==============================] - 6s 25ms/step - loss: 0.0094 - accuracy: 0.9972 - val_loss: 0.0772 - val_accuracy: 0.9824
Epoch 10/10
235/235 [==============================] - 8s 33ms/step - loss: 0.0080 - accuracy: 0.9972 - val_loss: 0.0658 - val_accuracy: 0.9835
```

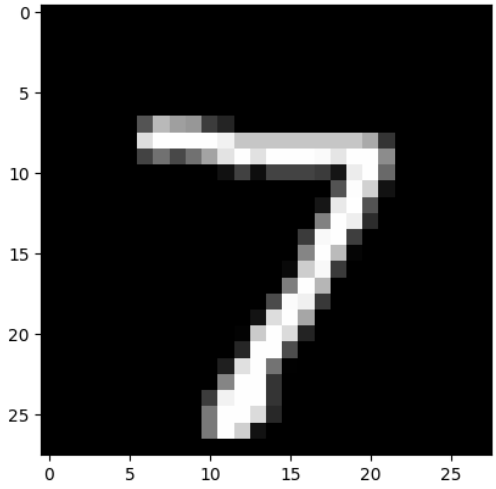
```
[ ] #2(a) Plot the loss and accuracy for both training data and validation data using the history object in the source
    #code
    from google.colab import drive
    drive.mount('/content/gdrive')
    import keras
    from keras.datasets import mnist
    from keras.models import Sequential
    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
```

```
469/469 [==============================] - 11s 24ms/step - loss: 0.0184 - accuracy: 0.9939 - val_loss: 0.0725 - val_accuracy: 0.9824
Epoch 15/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0191 - accuracy: 0.9939 - val_loss: 0.0698 - val_accuracy: 0.9837
Epoch 16/20
469/469 [==============================] - 9s 20ms/step - loss: 0.0152 - accuracy: 0.9949 - val_loss: 0.0815 - val_accuracy: 0.9831
Epoch 17/20
469/469 [==============================] - 11s 25ms/step - loss: 0.0193 - accuracy: 0.9937 - val_loss: 0.0743 - val_accuracy: 0.9818
Epoch 18/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0147 - accuracy: 0.9951 - val_loss: 0.0774 - val_accuracy: 0.9829
Epoch 19/20
469/469 [==============================] - 11s 22ms/step - loss: 0.0161 - accuracy: 0.9947 - val_loss: 0.0770 - val_accuracy: 0.9828
Epoch 20/20
469/469 [==============================] - 10s 20ms/step - loss: 0.0147 - accuracy: 0.9953 - val_loss: 0.0785 - val_accuracy: 0.9832
```



```
[ ]  #2(b)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.

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

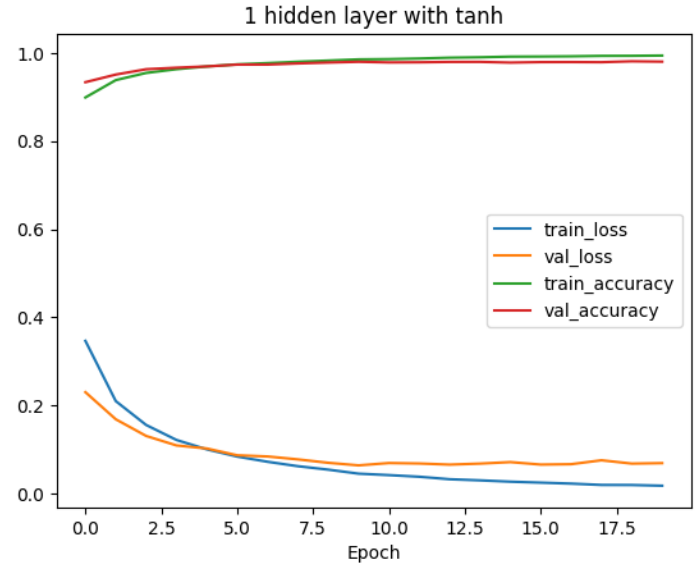
```
Epoch 20/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0184 - accuracy: 0.9939 - val_loss: 0.0717 - val_accuracy: 0.9840
```



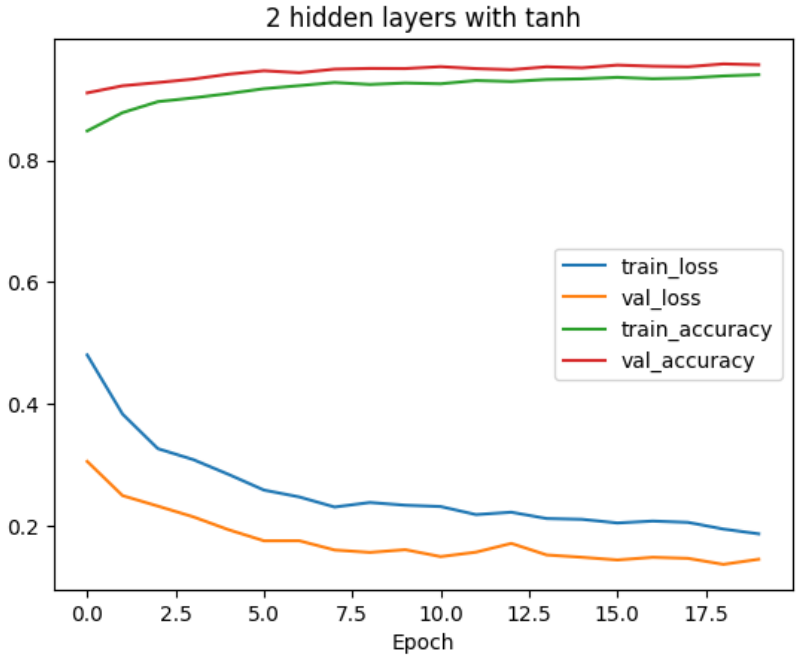

```
1/1 [==============================] - 0s 149ms/step
Model prediction: 7
```

```
[ ] #2(c)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.

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

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

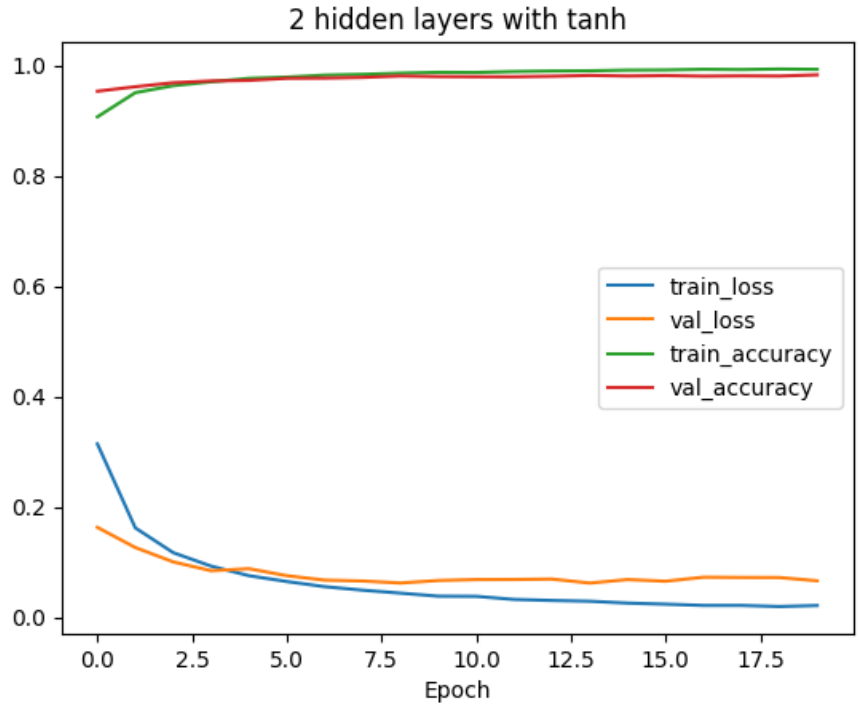



```
1 hidden layer with tanh - Test loss: 0.0689, Test accuracy: 0.9803
```




```
2 hidden layers with tanh - Test loss: 0.1446, Test accuracy: 0.9569
```

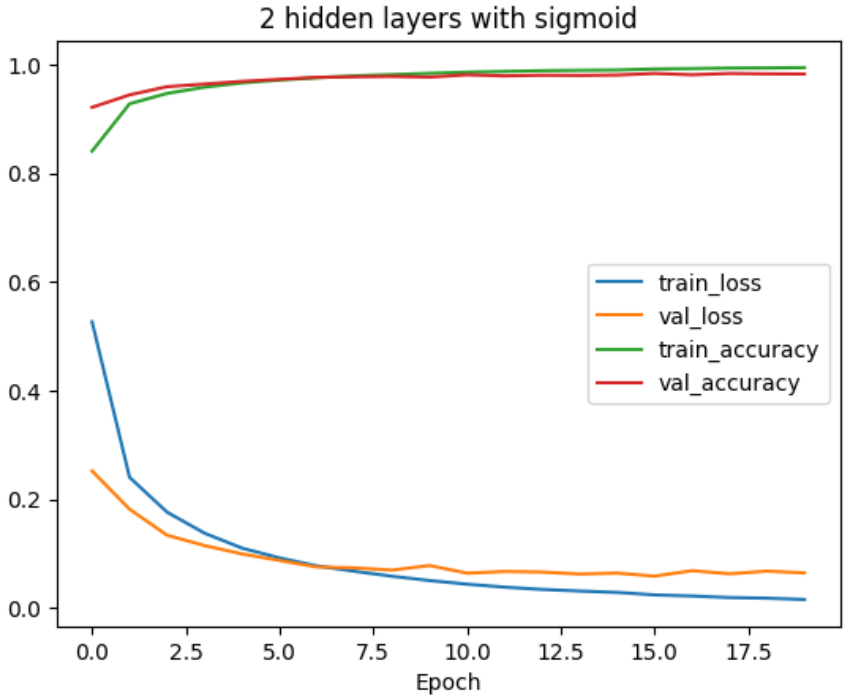
```
1 hidden layer with sigmoid - Test loss: 0.0597, Test accuracy: 0.9818
```




```
2 hidden layers with tanh - Test loss: 0.0663, Test accuracy: 0.9827
```

[4]

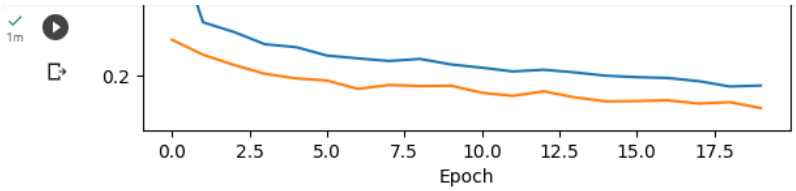
```
2 hidden layers with tanh - Test loss: 0.0663, Test accuracy: 0.9827
```



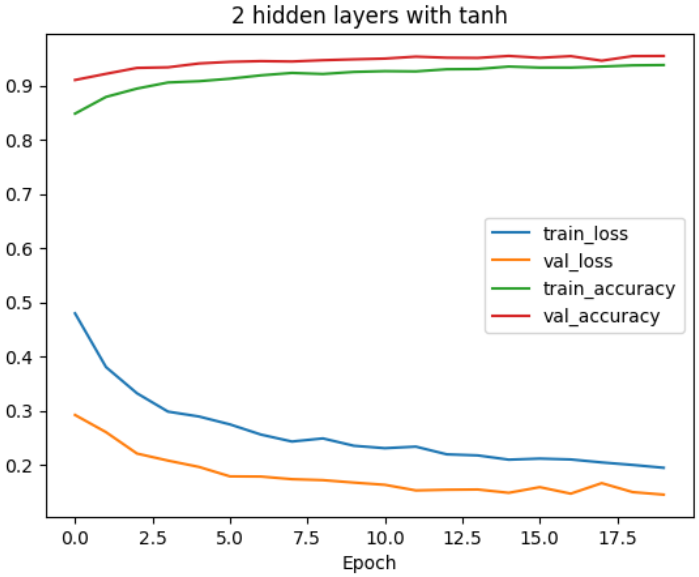

```
2 hidden layers with sigmoid - Test loss: 0.0652, Test accuracy: 0.9826
```

[5]

```python
#2(d)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
```

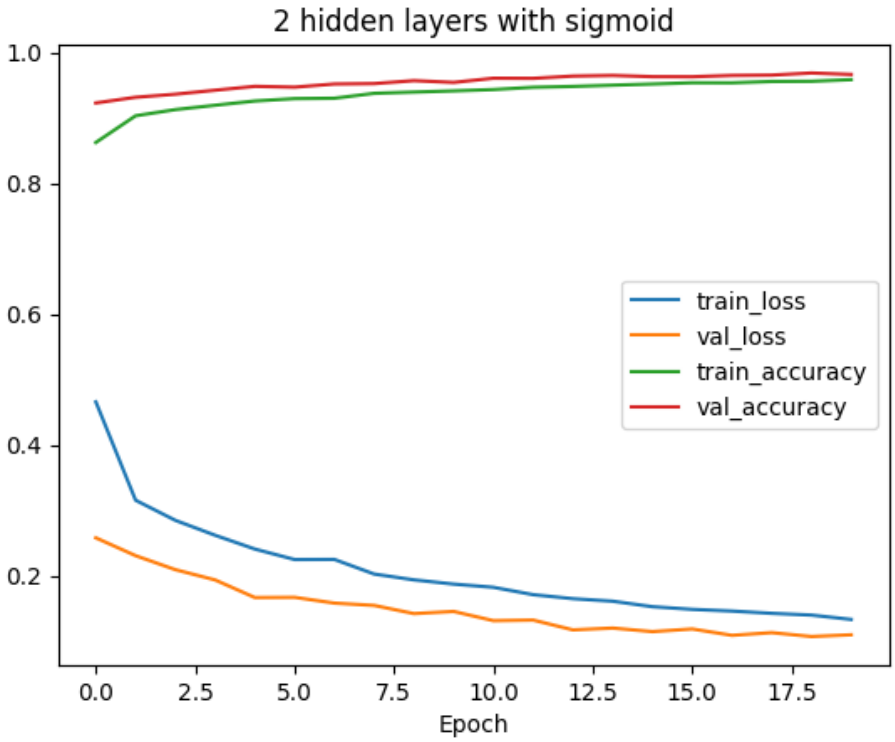



```
1 hidden layer with sigmoid - Test loss: 0.1389, Test accuracy: 0.9610
```




```
2 hidden layers with tanh - Test loss: 0.1455, Test accuracy: 0.9543
```

2 hidden layers with tanh - Test loss: 0.1455, Test accuracy: 0.9543




```
2 hidden layers with sigmoid - Test loss: 0.1097, Test accuracy: 0.9660
```