Git Hub Link: https://github.com/srija1609/NNDL_ICP-6

Spring 2024: CS5720

Neural Networks & Deep Learning - ICP-6

Name: Baby Srija Bitra

#700: 700755908

1. Use the use case in the class:

a. Add more Dense layers to the existing code and check how the accuracy changes.

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

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()

Breast Cancer dataset is designated to predict if a patient has Malignant (M) or Benign = B cancer

Use Image Classification on the hand written 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
#read the data
import pandas as pd
data = pd.read_csv('/content/diabetes.csv')

path_to_csv = '/content/diabetes.csv'

import keras
import pandas
from keras.models import Sequential
from keras.layers 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))
```

```
Epoch 100/100
18/18 [==============================] - 0s 3ms/step - loss: 0.5426 - acc: 0.7326
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 (1.05 KB)
Trainable params: 269 (1.05 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.6027 - acc: 0.7135
[0.60271817445755, 0.7135416865348816]
```

```
[3] #read the data
    data = pd.read_csv('/content/breastcancer.csv')

    path_to_csv = '/content/breastcancer.csv'

    import keras
    import pandas as pd
    import numpy as np
    from keras.models import Sequential
    from keras.layers 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))
```

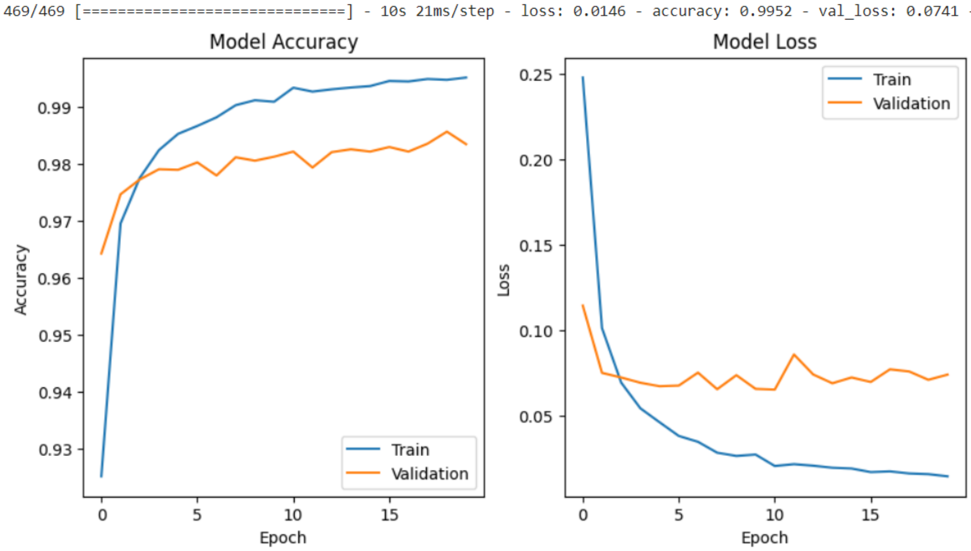
```
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1366 - acc: 0.9413
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 20)                620

 dense_4 (Dense)             (None, 1)                 21

=================================================================
Total params: 641 (2.50 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
5/5 [==============================] - 0s 4ms/step - loss: 0.2581 - acc: 0.9231
[0.2581396996974945, 0.9230769276618958]
```

```
[4] #read the data
    data = pd.read_csv('/content/breastcancer.csv')

    path_to_csv = '/content/breastcancer.csv'

    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 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 100/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1304 - acc: 0.9437
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 20)                620

 dense_6 (Dense)             (None, 1)                 21

=================================================================
Total params: 641 (2.50 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
5/5 [==============================] - 0s 3ms/step - loss: 0.3480 - acc: 0.9231
[0.3479941785335541, 0.9230769276618958]
```

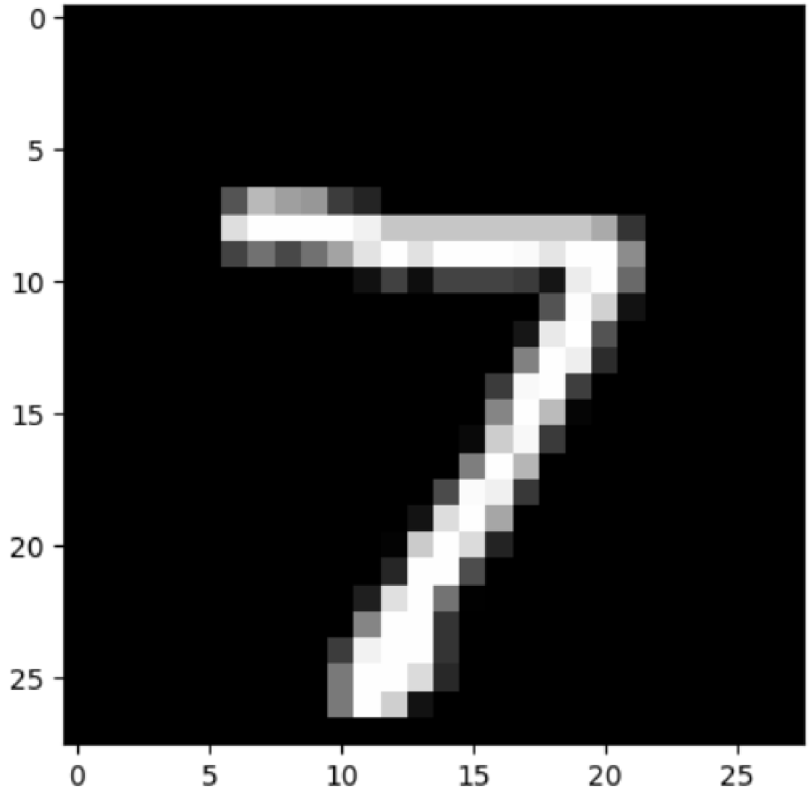
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt

# 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))
```

```
469/469 [==============================] - 10s 21ms/step - loss: 0.0146 - accuracy: 0.9952 - val_loss: 0.0741 - val_accuracy: 0.9835
```

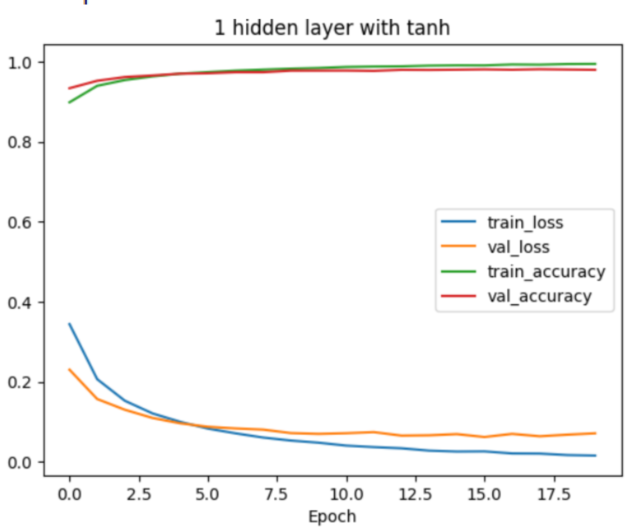



```
Epoch 1/20
469/469 [==============================] - 11s 22ms/step - loss: 0.2501 - accuracy: 0.9239 - val_loss: 0.1029 - val_accuracy: 0.9674
```

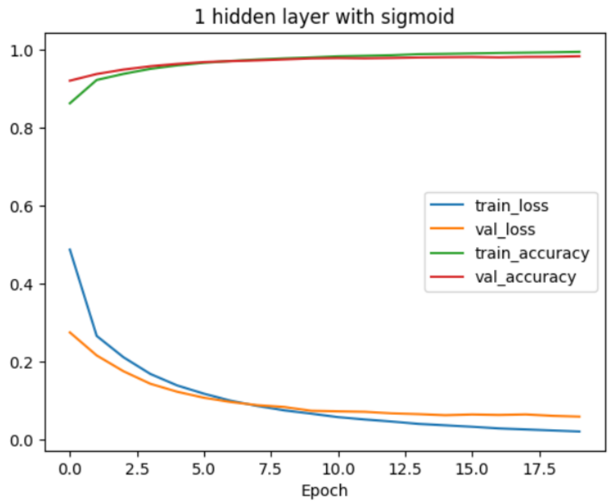
```
Epoch 20/20
469/469 [==============================] - 10s 21ms/step - loss: 0
```



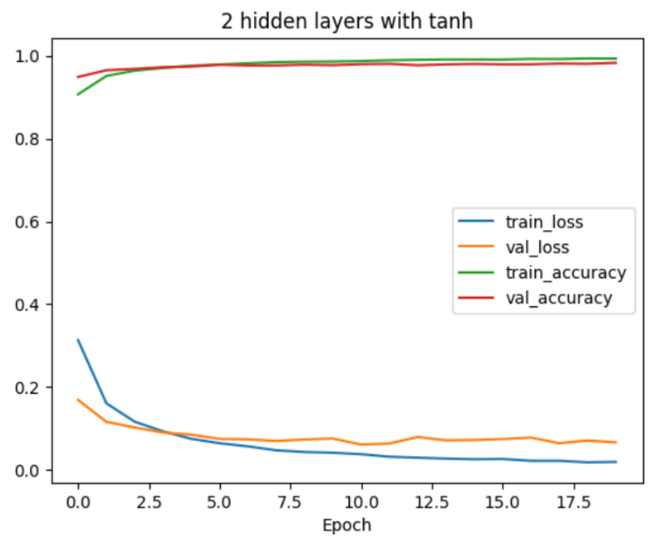
```
1/1 [==============================] - 0s 156ms/step
Model prediction: 7
```



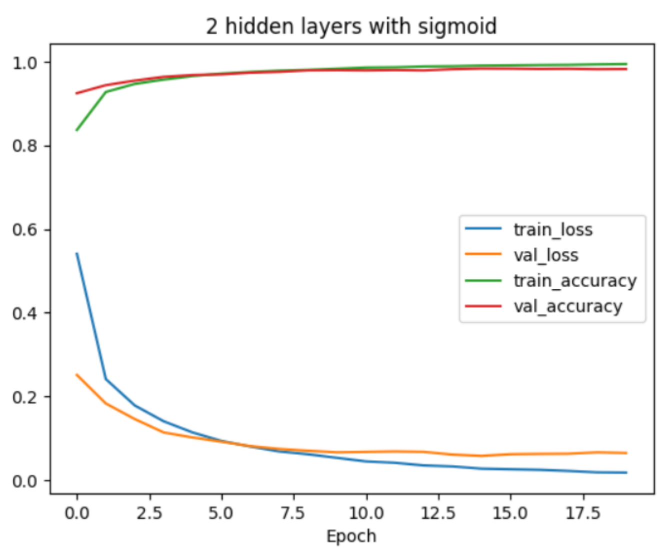

```
1 hidden layer with tanh - Test loss: 0.0711, Test accuracy: 0.9801
```



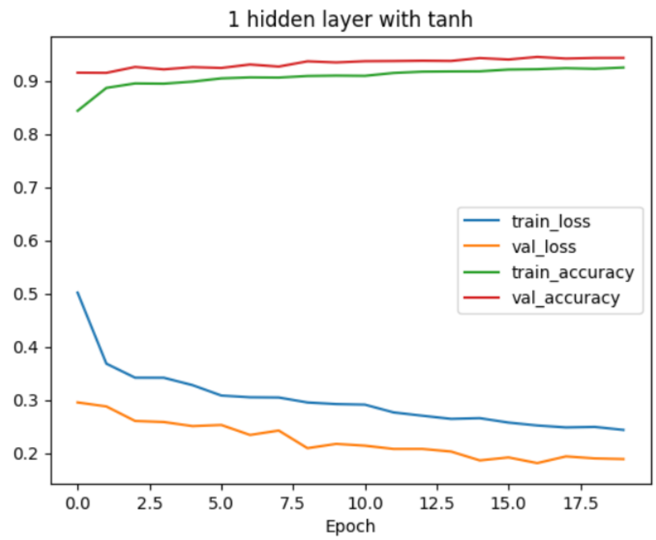
1 hidden layer with sigmoid - Test loss: 0.0594, Test accuracy: 0.9829



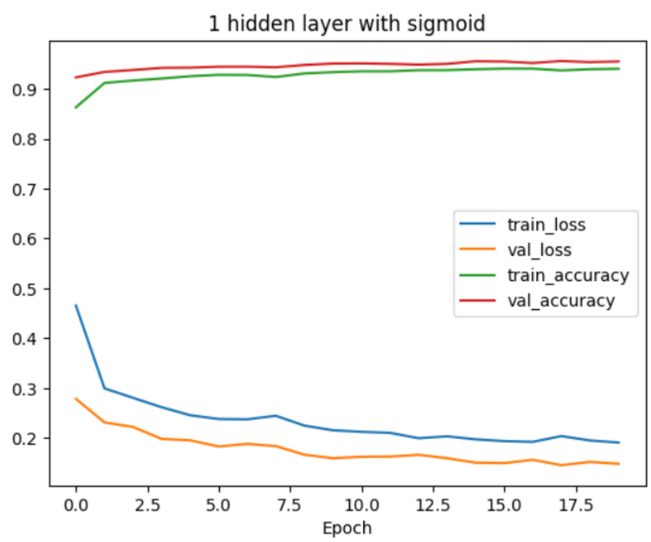
2 hidden layers with tanh - Test loss: 0.0669, Test accuracy: 0.9829



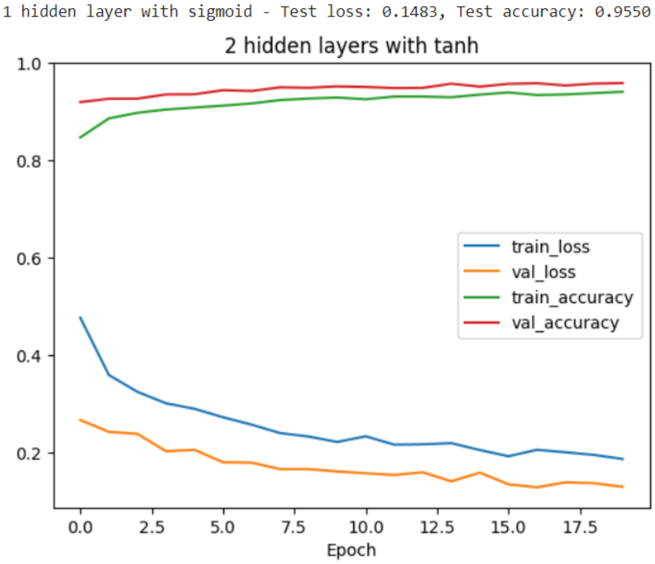
2 hidden layers with sigmoid - Test loss: 0.0641, Test accuracy: 0.9825



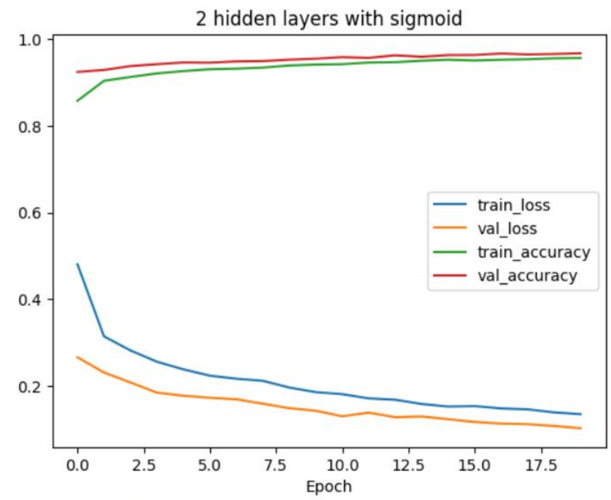

```
1 hidden layer with tanh - Test loss: 0.1886, Test accuracy: 0.9429
```




```
1 hidden layer with sigmoid - Test loss: 0.1483, Test accuracy: 0.9550
```




```
2 hidden layers with tanh - Test loss: 0.1298, Test accuracy: 0.9587
```




```
2 hidden layers with sigmoid - Test loss: 0.1023, Test accuracy: 0.9671
```