Spring 2025: Neural Networks & Deep Learning – ICP -4

Assignment -4

Name: Veera Manikanta Kumar Allada

Student ID: 700756934

Github Link: https://github.com/maniallada9/Neural-Networks-deep-Learning

Video Link: https://drive.google.com/file/d/1dPtk69T4zs7JuyY-6rCB9onrpGJpcTqu/view?usp=drive_link

1)

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()

outputs:

1)

Model: "sequential_3"

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 20)	180
dense_7 (Dense)	(None, 30)	630
dense_8 (Dense)	(None, 1)	31

Total params: 2,525 (9.87 KB)
Trainable params: 841 (3.29 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 1,684 (6.58 KB)

None

6/6 ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - acc: 0.6556 - loss: 0.6371

[0.6192042827606201, 0.6614583134651184]

2)

```
Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 20)	620
dense_10 (Dense)	(None, 1)	21

```
 Total params: 1,925 (7.52 KB)
 Trainable params: 641 (2.50 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 1,284 (5.02 KB)
None
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - acc: 0.9262 - loss: 0.2677
[0.19420748949050903, 0.9440559148788452]
```

3)

```
Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 20)	620
dense_10 (Dense)	(None, 1)	21

```
 Total params: 1,925 (7.52 KB)
 Trainable params: 641 (2.50 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 1,284 (5.02 KB)
None
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - acc: 0.9262 - loss: 0.2677
[0.19420748949050903, 0.9440559148788452]
```

2.

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 sourcecode.

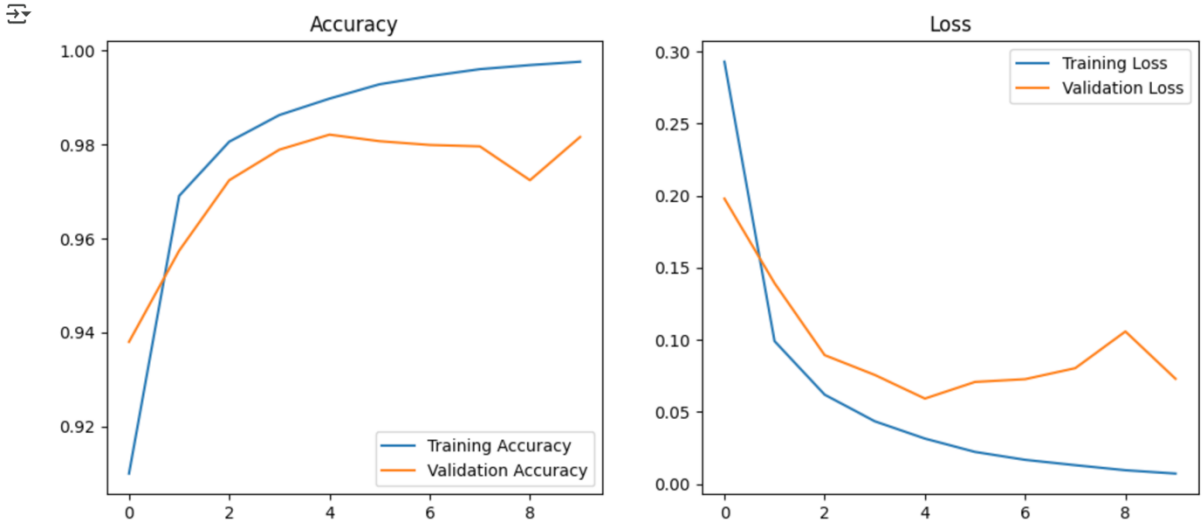
2.Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.

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?

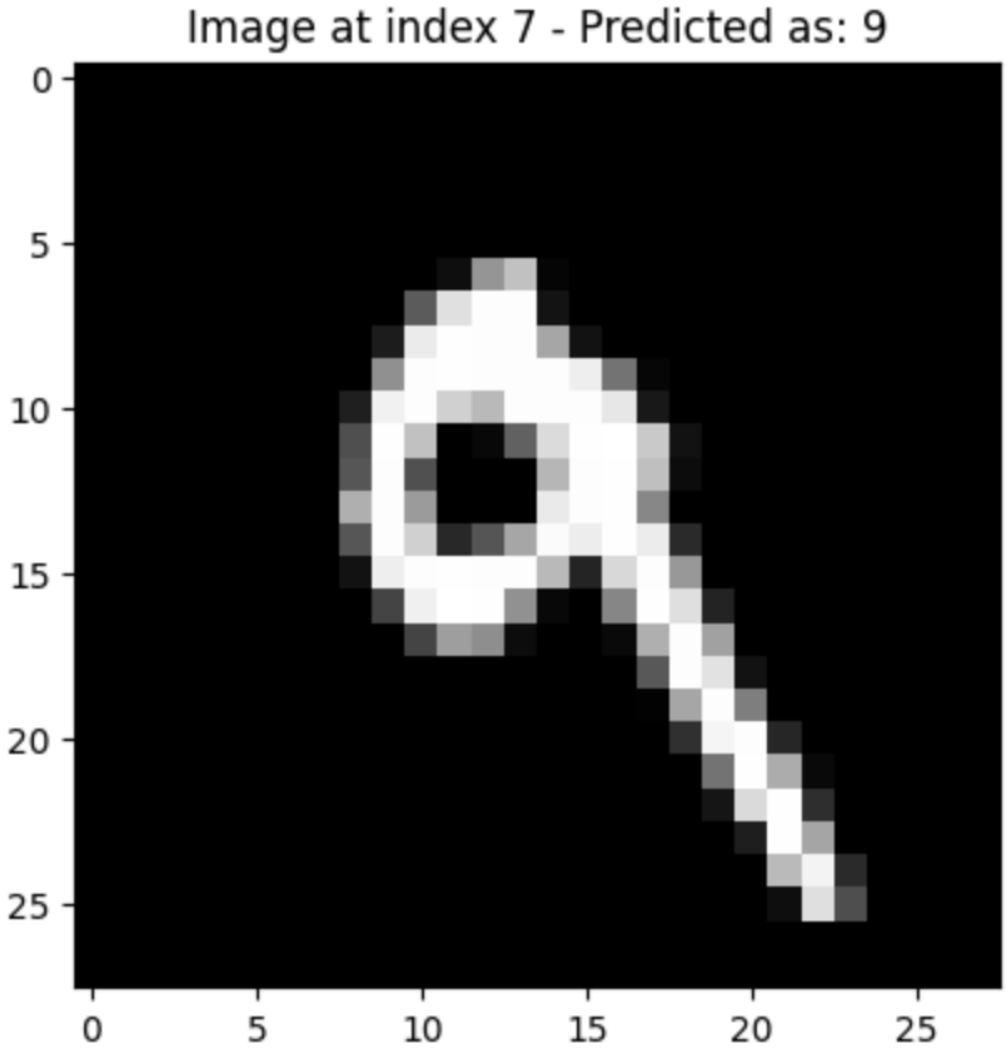
Outputs:

1.


Accuracy
1.00
0.98
0.96
0.94
0.92
0
2
4
6
8
Training Accuracy
Validation Accuracy
Loss
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
Training Loss
Validation Loss

2.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step
Predicted class for image at index 7: 9
```



3.

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 22ms/step - accuracy: 0.8240 - loss: 0.5951 - val_accuracy: 0.9020 - val_loss: 0.3220
Epoch 2/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - accuracy: 0.9272 - loss: 0.2529 - val_accuracy: 0.9412 - val_loss: 0.2006
Epoch 3/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 24ms/step - accuracy: 0.9483 - loss: 0.1798 - val_accuracy: 0.9507 - val_loss: 0.1659
Epoch 4/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 17ms/step - accuracy: 0.9604 - loss: 0.1377 - val_accuracy: 0.9539 - val_loss: 0.1456
Epoch 5/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - accuracy: 0.9697 - loss: 0.1059 - val_accuracy: 0.9686 - val_loss: 0.1088
Epoch 6/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 18ms/step - accuracy: 0.9745 - loss: 0.0852 - val_accuracy: 0.9691 - val_loss: 0.1012
Epoch 7/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 23ms/step - accuracy: 0.9803 - loss: 0.0688 - val_accuracy: 0.9703 - val_loss: 0.0981
Epoch 8/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - accuracy: 0.9841 - loss: 0.0563 - val_accuracy: 0.9720 - val_loss: 0.0927
Epoch 9/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - accuracy: 0.9860 - loss: 0.0492 - val_accuracy: 0.9764 - val_loss: 0.0772
Epoch 10/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 27ms/step - accuracy: 0.9874 - loss: 0.0435 - val_accuracy: 0.9771 - val_loss: 0.0705
<keras.src.callbacks.history.History at 0x7d47046cf450>
```

4.

```
(28, 28)
784
Epoch 1/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 35ms/step - accuracy: 0.7993 - loss: 18.4662 - val_accuracy: 0.8970 - val_loss: 1.1762
Epoch 2/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 10s 42ms/step - accuracy: 0.9434 - loss: 0.4300 - val_accuracy: 0.9443 - val_loss: 0.3352
Epoch 3/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 7s 28ms/step - accuracy: 0.9603 - loss: 0.2257 - val_accuracy: 0.9403 - val_loss: 0.3470
Epoch 4/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 10s 28ms/step - accuracy: 0.9674 - loss: 0.1779 - val_accuracy: 0.9505 - val_loss: 0.3241
Epoch 5/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 38ms/step - accuracy: 0.9727 - loss: 0.1520 - val_accuracy: 0.9330 - val_loss: 0.4994
Epoch 6/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 8s 28ms/step - accuracy: 0.9757 - loss: 0.1415 - val_accuracy: 0.9596 - val_loss: 0.2997
Epoch 7/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 10s 28ms/step - accuracy: 0.9784 - loss: 0.1308 - val_accuracy: 0.9651 - val_loss: 0.3033
Epoch 8/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 8s 33ms/step - accuracy: 0.9820 - loss: 0.1055 - val_accuracy: 0.9539 - val_loss: 0.4541
Epoch 9/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 7s 28ms/step - accuracy: 0.9825 - loss: 0.1142 - val_accuracy: 0.9636 - val_loss: 0.4374
Epoch 10/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 8s 33ms/step - accuracy: 0.9853 - loss: 0.1071 - val_accuracy: 0.9629 - val_loss: 0.4273
```