Rishikesh

R11643260

March 3, 2025

CS 5388 – D01: Neural Networks

Assignment No.1 Coding Part Write-up

Disclaimer: The model structure given in the project description was given as below:

Model 1

```
net = NeuralNetwork()
net.add(FullyConnectedLayer(input_size,num_classes))
net.add(SigmoidActivationLayer())
net.add(SoftmaxLayer())
```

Model 2

```
net = NeuralNetwork()
net.add(FullyConnectedLayer(input_size,hidden_layer_size))
net.add(ReLUActivationLayer())
net.add(FullyConnectedLayer(hidden_layer_size,num_classes))
net.add(SoftmaxLayer())
```

But the model structure pre-defined in main.py was written as this:

```
# Initialize model 1
net = NeuralNetwork()
net.add(FullyConnectedLayer(input_size, num_classes))
net.add(ReLUActivationLayer())
net.add(SoftmaxLayer())
# Initialize model 2
net = NeuralNetwork()
net.add(FullyConnectedLayer(input_size,hidden_layer_size))
net.add(SigmoidActivationLayer())
net.add(FullyConnectedLayer(hidden_layer_size,num_classes))
net.add(SoftmaxLayer())
```

Which is different than the project description. So, I have changed the model structure in main.py to match the project description.

I have changed the main function in the main.py to also return the learning rate parameter to put that information on the plots. I have also added a print statement in main.py to print time taken for the model to complete training to the terminal.

Model Experiments:

Set Parameters:

Batch Size = 64

Number of Epochs = 30

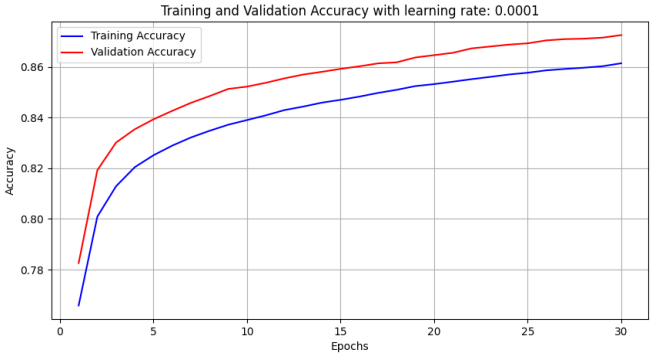
Hidden Layer Size = 128

Learning Rate = [0.1, 0.01, 0.001, 0.0001]

Time taken = I mean this as average time taken to complete training 3 times on my machine.*

Model 1:

LR = 0.0001

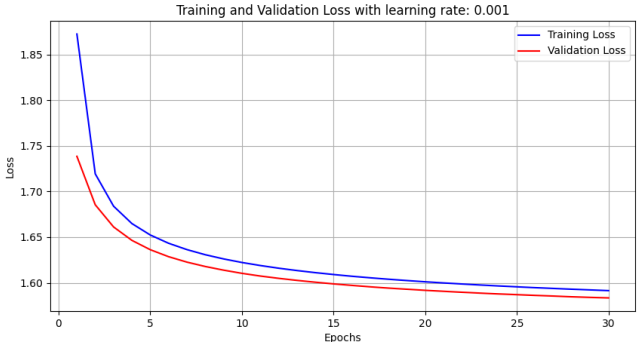


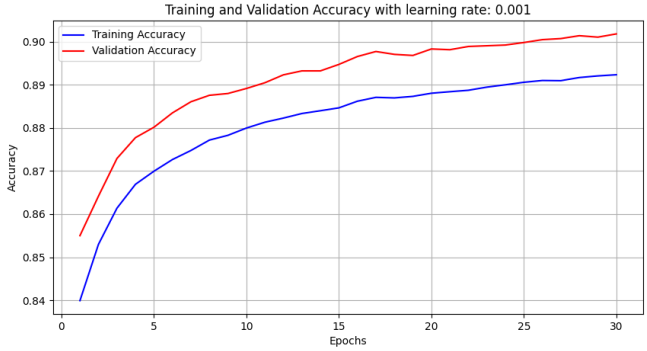
Training Accuracy: 0.8614

Test Accuracy: 0.8727

Time taken: 58.86 seconds

LR = .001



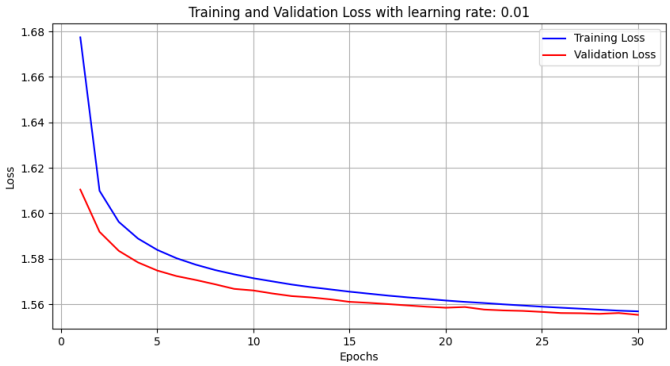


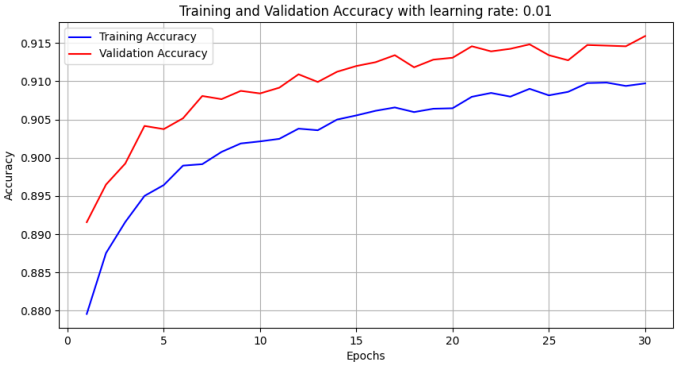
Training Accuracy: 0.8924

Test Accuracy: 0.9002

Time taken: 59.23 seconds

LR = 0.01



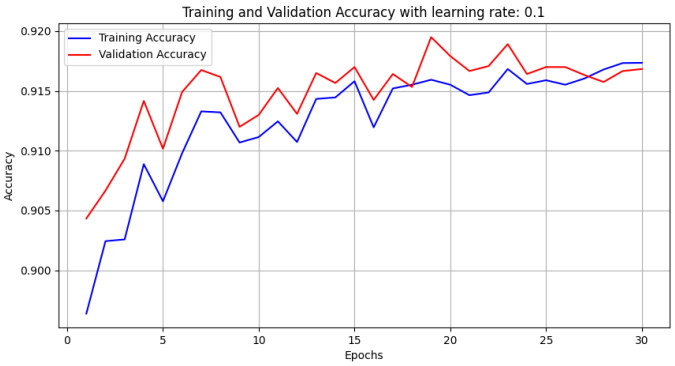


Training Accuracy: 0.9097

Test Accuracy: 0.9107

Time taken: 60.74 seconds

LR = 0.1



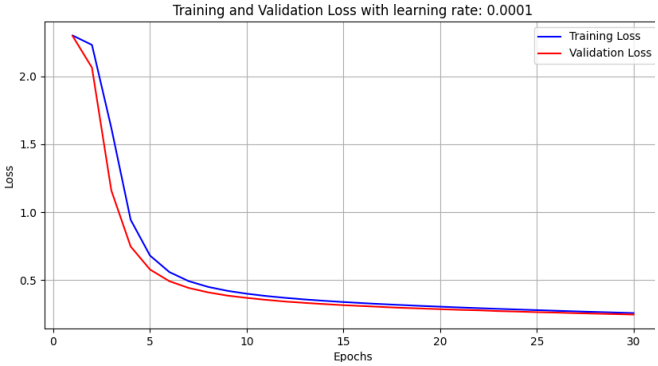
Training Accuracy: 0.9174

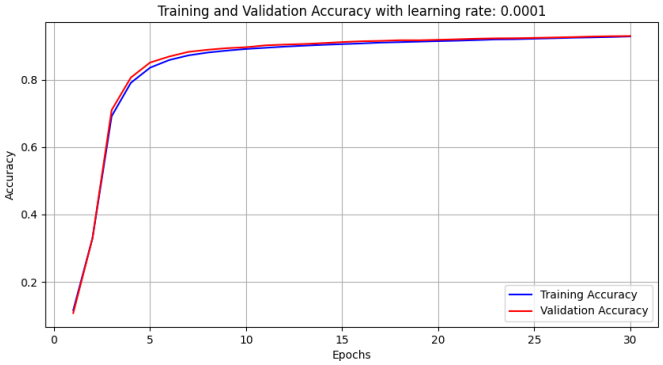
Test Accuracy: 0.9135

Time taken: 68.47 seconds

MODEL 2

LR = 0.0001



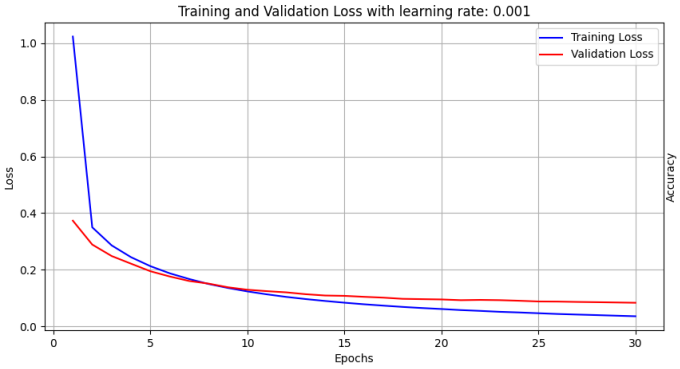


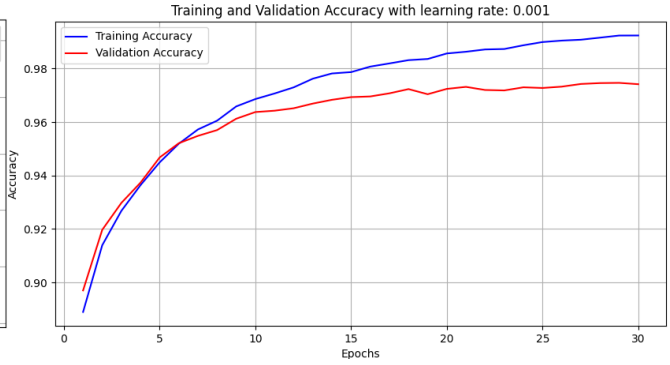
Training Accuracy: 0.9282

Test Accuracy: 0.9291

Time taken: 213.02 seconds

LR = 0.001



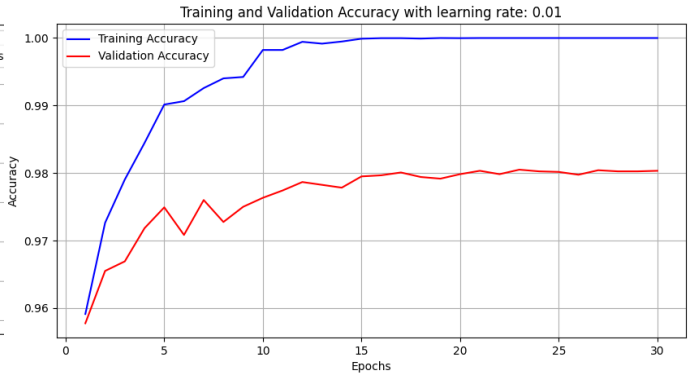


Training Accuracy: 0.9924

Test Accuracy: 0.9775

Time taken: 227.18 seconds

LR = 0.01

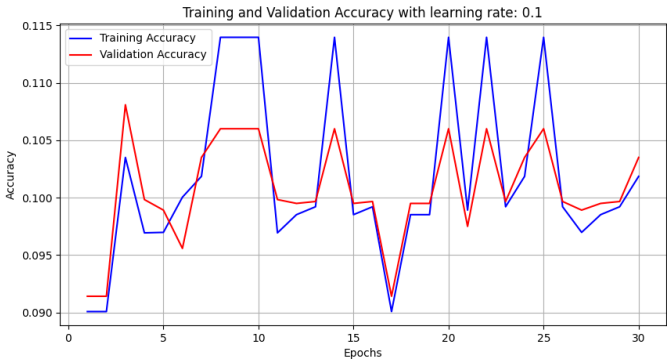


Training Accuracy: 1.0000

Test Accuracy: 0.9810

Time taken: 233.22 seconds

LR = 0.1



Training Accuracy: 0.1019

Test Accuracy: 0.1010

Time taken: 228.55 seconds

Conclusion:

Model 1:

Higher Learning rates generally improved model performance for model 1 with the best test accuracy at 91.35% achieved with LR = 0.1. The time taken to train the model 1 also slightly kept increasing with higher learning rate. The loss curves don't show any dramatic decreases for higher learning rates, yet the model accuracy gets better. With LR=0.0001, there's steady improvement through all 30 epochs. With LR=0.1, the model converges faster but shows oscillations in the accuracy curve, indicating less stable training. Despite these oscillations, the higher learning rate achieves better final results.

Model 2:

At LR=0.1, the model completely fails to learn. At LR=0.01, it achieves perfect training accuracy (100%) and excellent test accuracy (98.10%). With LR=0.01, the gap between training accuracy (100%) and test accuracy (98.10%) suggests some overfitting. The validation loss increases slightly while training loss continues to decrease, another sign of overfitting. Model 2 takes significantly longer to train (~213-233s) compared to Model 1 (~59-68s) but this is expected due to the extra hidden layer and parameters.

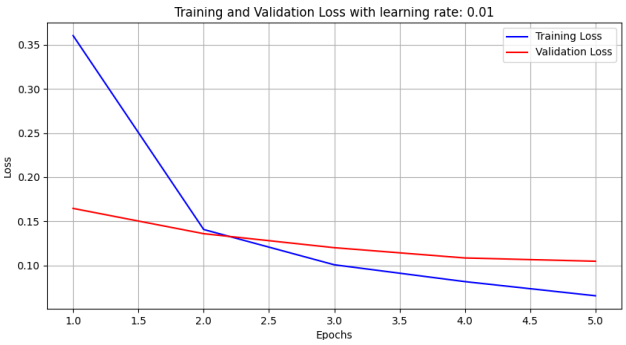
MODEL 2 : HYPERPARAMETER TUNING

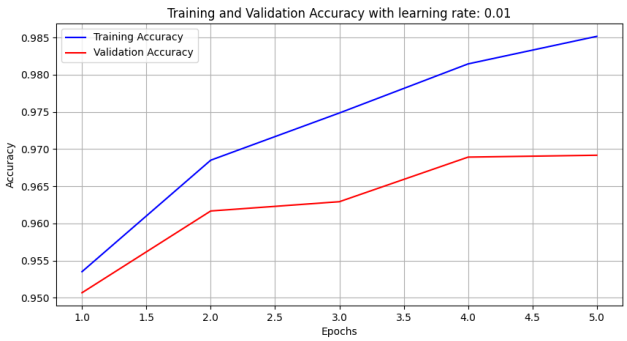
"Best" param that I found:

Batch Size = 32

Learning Rate = 0.01

Hidden Layer Size = 64





Training Accuracy: 0.9852

Test Accuracy: 0.9722

Time taken: 50.96 seconds

The Model achieved very high accuracy with only 5 epochs. The validation accuracy is also very closely tracking the training accuracy so there isn't any significant overfitting either.