MAE C263F Homework 3

Jake Kremer – Fall 2024 – Professor M. Khalid Jawed

***Abstract*— This assignment was to implement code to create a feedforward neural network to solve classification problems regarding reading grayscale images.**

I. Introduction

The rise of big data has revolutionized machine learning, enabling models to achieve outstanding performance across a variety of tasks by learning from extensive datasets. One such application is classification, a learning process that assigns categorical labels to input data. In this project, we apply machine learning to the classification of handwritten digits. Specifically, we implement a feedforward neural network to analyze grayscale images and predict the digit represented by each image.

The process involves preprocessing 28×28 pixel images by flattening them into vectors of length 784, which serve as inputs to the neural network. Through computations across hidden layers, activation functions, and a softmax function, the network produces a 10-dimensional output vector. This output identifies the predicted digit, with a single active element corresponding to the classified digit. For example, an active first element represents the digit 0, while an active second element indicates the digit 1. This report details the steps involved in building, training, and testing the feedforward neural network, illustrating the power and practicality of machine learning in solving real-world classification problems.

Figure 1 below shows a flowchart of the code architecture.

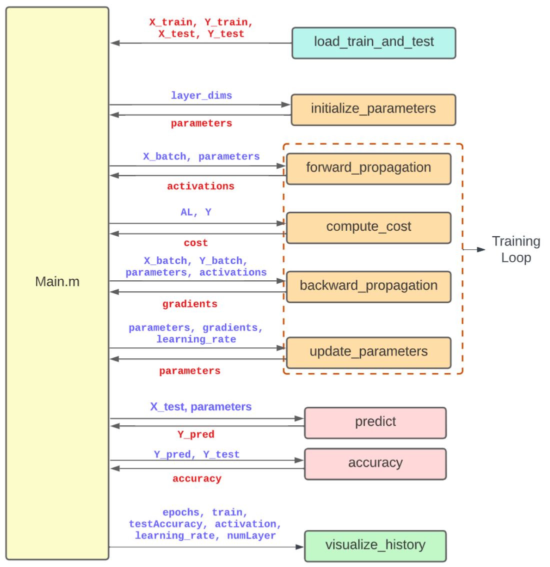

Figure 1 – Code Architecture Flowchart

II. Results and Figures

Figure 2 below shows training progress of the network with 50 epochs, a learning rate of 0.01, and 2 hidden layers.

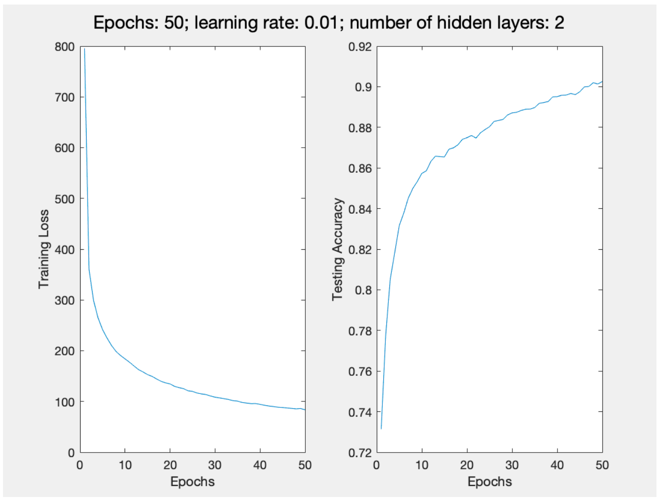

Figure 2 – Training Progress with 50 Epochs

Figure 3 below shows training progress of the network with 150 epochs, a learning rate of 0.01, and 2 hidden layers.

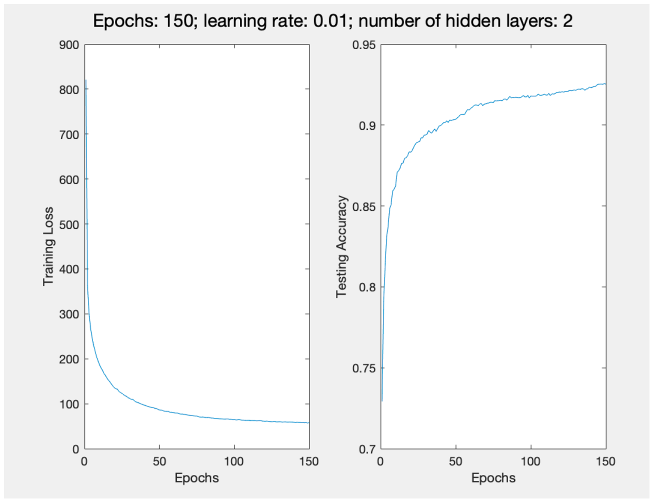

Figure 3 – Training Progress with 150 Epochs

Figure 4 below shows training progress of the network with 300 epochs, a learning rate of 0.01, and 2 hidden layers.

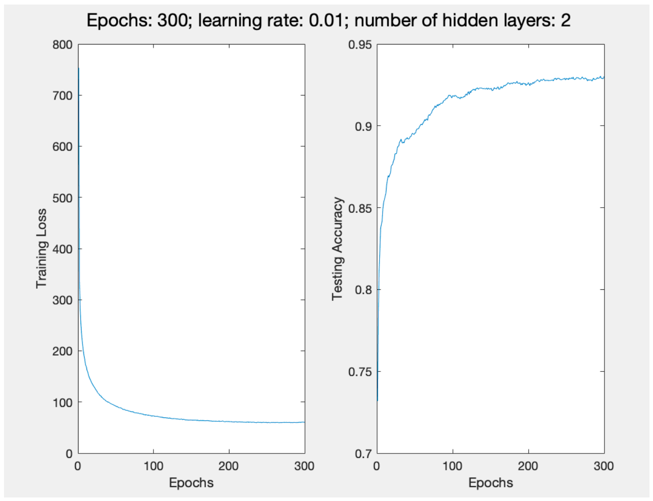

Figure 4 – Training Progress with 300 Epochs

Figure 5 below shows training progress of the network with 150 epochs, a learning rate of 0.1, and 2 hidden layers.

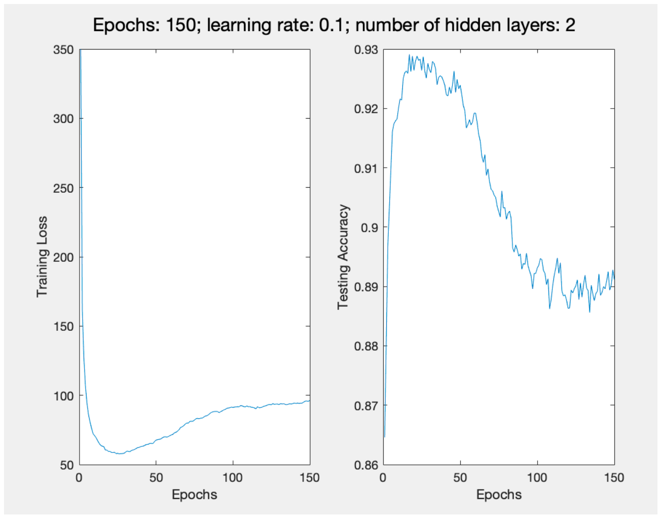

Figure 5 – Training Progress with Learning Rate of 0.1

Figure 6 below shows training progress of the network with 150 epochs, a learning rate of 0.001, and 2 hidden layers.

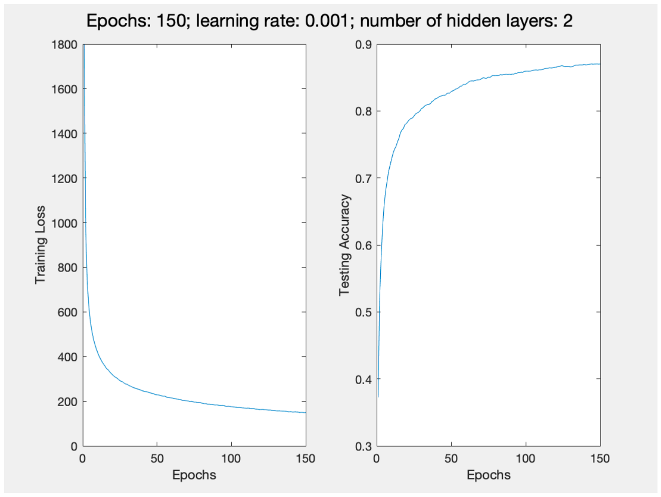

Figure 6 – Training Progress with Learning Rate of 0.001

Figure 7 below shows training progress of the network with 150 epochs, a learning rate of 0.01, and 3 hidden layers.

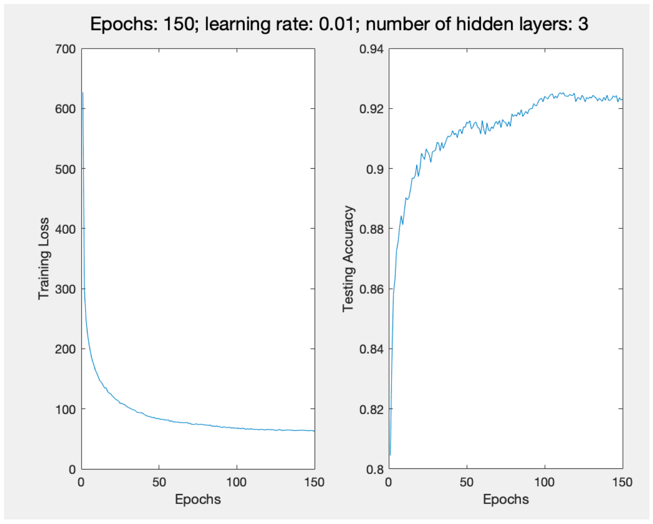

Figure 7 – Training Progress with 3 Hidden Layers

Figure 8 below shows training progress of the network with 150 epochs, a learning rate of 0.01, and 5 hidden layers.

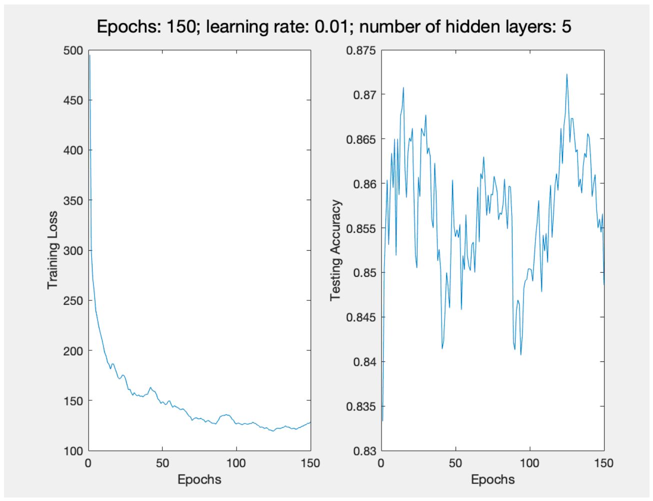

Figure 8 – Training Progress with 5 Hidden Layers

III. Discussion

Changing the number of epochs shows that an increased number of epochs leads to higher testing accuracy and better results. This is expected, as the more times you pass the dataset through the algorithms, the more accurate the results will be. We also notice that there are diminishing returns as you increase the number of epochs. The testing accuracy increases from a final value of around 0.902 with 50 epochs to a final value of around 0.9203 with 150 epochs. This is a decent increase in testing accuracy. However, doubling the number of epochs from 150 to 300 only increases the training accuracy to a final value of around 0.927. This change is not very large considering the number of epochs was changed so substantially.

Altering the learning rate of the network leads to varied outcomes in the testing accuracy. We see that with a learning rate of 0.01, the testing accuracy plateaus to around 0.925. With a rate of 0.1, the testing accuracy climbs to 0.93, then drops to about 0.89. With a rate of 0.001, the accuracy plateaus to around 0.88. These varied outcomes show there is an ideal learning rate sweet spot that must be determined for each neural network.

Finally, increasing the number of hidden layers leads to lower final values in testing accuracy. We can see that at 2 hidden layers the final testing accuracy value is around 0.925. With 3 this becomes 0.921, and the rate increases at a less constant rate. With 5 the final learning rate becomes about 0.855, and plot shows very sporadic behavior due to the increased number of hidden layers.

IV. Conclusion

In this assignment, we successfully implemented a feedforward neural network to classify handwritten digits, demonstrating the practical applications of machine learning for real-world tasks. Through extensive experimentation, we evaluated the effects of key hyperparameters, including the number of epochs, learning rates, and hidden layers, on the network's performance.

Our findings revealed that while increasing epochs improved testing accuracy, diminishing returns became apparent with excessive iterations. The learning rate played a critical role in balancing convergence speed and accuracy, emphasizing the importance of tuning this parameter for optimal performance. Additionally, increasing the number of hidden layers led to reduced accuracy and less stable results, underscoring the challenges of overfitting and model complexity in neural networks. Overall, this project provided valuable insights into designing and optimizing feedforward neural networks for classification tasks.