MAE 263F: Homework 01

Pranav Sawant

Abstract— **This document contains the report of the Homework 03, option 2, for the course MAE 263F taught at UCLA by Prof. M. Khalid Jawed. This assignment investigates the impact of key hyperparameters—training epochs, learning rate, and number of hidden layers—on the performance of a machine learning model.**

I. Introduction

In this assignment we are asked to discuss about how different hyperparameters, like the number of training epochs, learning rates, and the number of hidden layers, affect the way our model learns during training. The starting values for these hyperparameters are: Number of Training Epochs: 150, Learning Rate: 0.1 and Number of hidden layers: 2.

II. Influence of Number of Training Epochs

First, lets discuss the influence of number of trainings epochs, here at the end of simulation at Learning Rate of 0.01 and the number of hidden layers being 2, the results we get are as follows:

At the end of 50 epochs : TA = 0.905; TL = 85
At the end of 150 epochs : TA = 0.917; TL = 68
At the end of 300 epochs : TA = 0.928; TL = 51

The terms TA and TL here mean Testing Accuracy and Training Loss respectively.

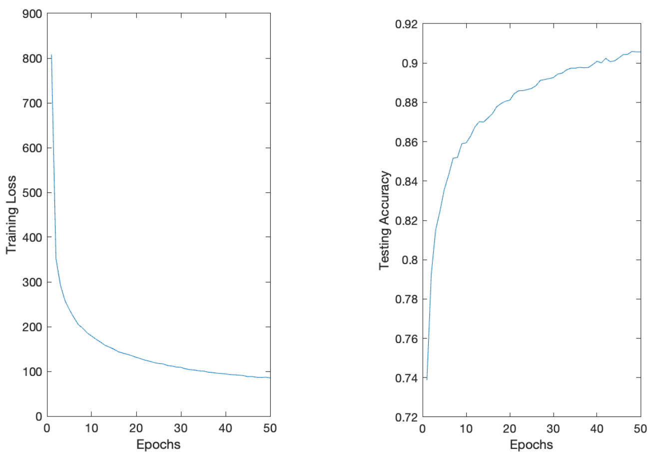

Fig 1. Training Loss and Testing Accuracy trends with respect to 50 training epochs.

The figures above and below shows the model performance with different number of training epochs. In the first case where epochs are limited to 50, the model accuracy is quite low. However, when the number of epochs is increased to 150, there is a significant improvement in accuracy, increasing the epochs further to 300 still improves accuracy, but the improvement is less noticeable.

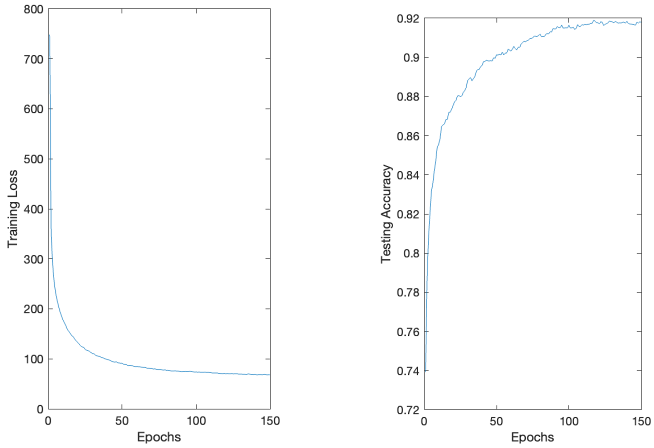

Fig 2: Training Loss and Testing Accuracy trends with respect to 150 training epochs.

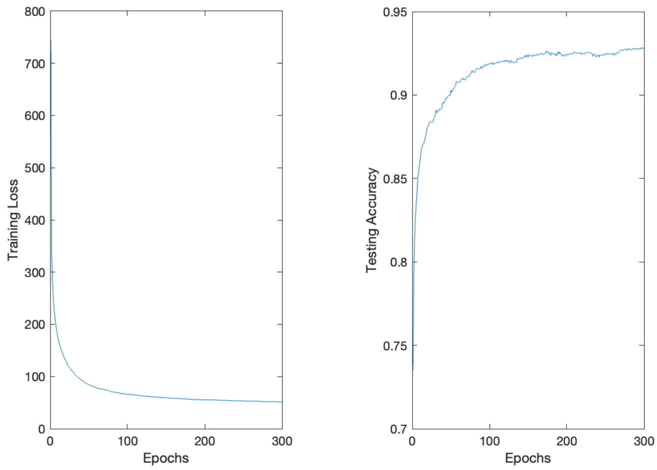

Fig 3: Training Loss and Testing Accuracy trends with respect to 300 training epochs.

These results clearly show a asymptotic trend between testing accuracy and the number of pocks. This aligns with the idea that the longer a module trained the accuracy tends to improve gradually and reaches a point where further improvement becomes smaller. Similarly, the training loss decreases as the number of ox increases getting closer to 0 overtime if this trend continues as po increase indefinitely, the accuracy would eventually approach one and the training loss would approach zero.

III. INFLUENCE OF LEARNING RATES

Now for the next part, we will discuss the influence of learning rates at the end of the simulation, At the end of 150 epochs for the number of hidden layer =2, the results we get are as follows:

For Learning Rate = 0.1 : TA = 0.884; TL = 101
For Learning Rate = 0.01 : TA = 0.923; TL = 64
For Learning Rate = 0.001 : TA = 0.872; TL = 142

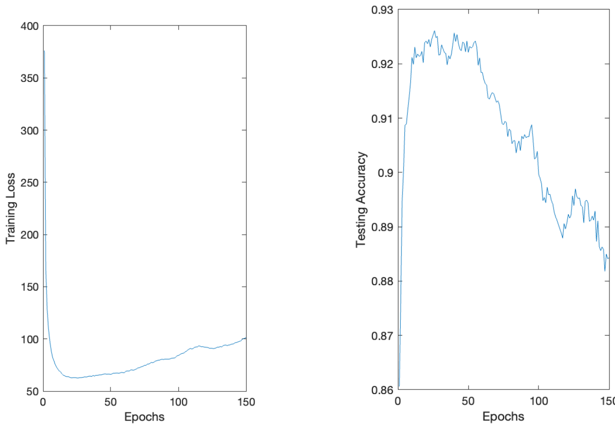

Fig 4: Training Loss and Testing Accuracy trends with respect to Learning rate of 0.1

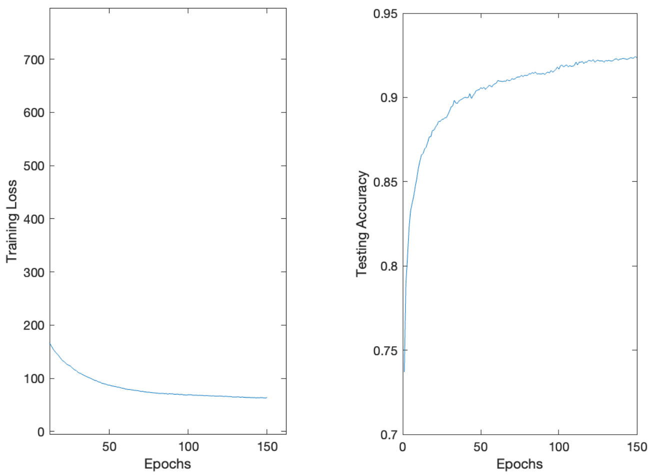

Fig 5: Training Loss and Testing Accuracy trends with respect to Learning rate of 0.01

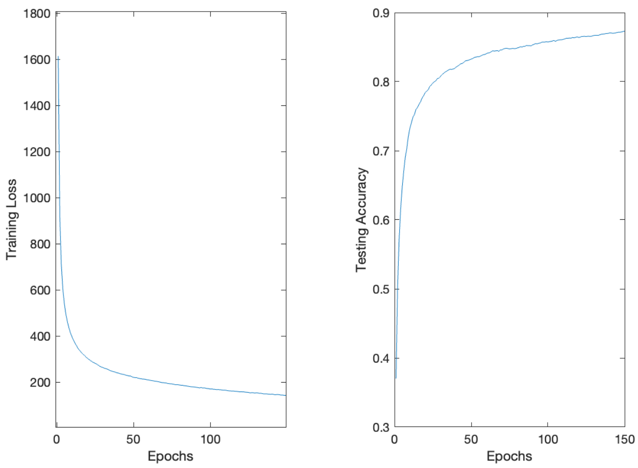

Fig 6: Training Loss and Testing Accuracy trends with respect to Learning rate of 0.001

Lets discuss the first case, when we set the learning rate to 0.1, the model initially performs well but starts losing accuracy after a few epochs. This happens because a high learning rate causes the gradient descent algorithm to take large steps, overshooting the optimal solution. As a result, the model either converges too quickly to a less-than-ideal solution or becomes unstable. From the graph, it's clear the model missed the optimal point, which should have been around epoch 25. So, a learning rate of 0.1 is too high for this model.

While if we consider the third case, when the learning rate is 0.001, the model takes a very long time to converge. A very small learning rate makes progress slow, and the model might either take too long to find the optimal solution or get stuck in a poor one. The graph shows that even after 150 epochs, the model hasn't converged. This indicates that 0.001 is too low for our needs.

Thus, if we analyse the results, we can conclude that the model works best with a learning rate of 0.01. It achieves better accuracy compared to the other two cases, converges steadily, and minimizes the training loss effectively.

IV. INFLUENCE OF NUMBER OF HIDDEN LAYERS

Finally, we now discuss the influence of number of hidden layers, the results we get at the end of the simulation are as follows:

For no. of hidden layers = 2 : TA = 0.93; TL = 60
For no. of hidden layers = 3 : TA = 0.925; TL = 61
For no. of hidden layers = 5 : TA = 0.863; TL = 123

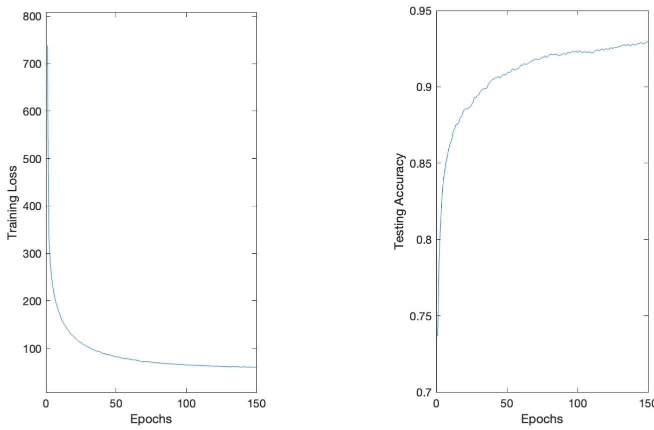

Fig 7: Training Loss and Testing Accuracy trends with respect to number of hidden layers of 2.

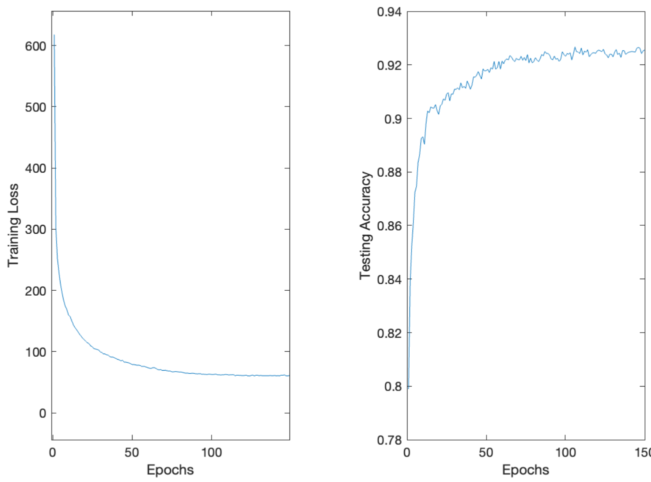

Fig 8: Training Loss and Testing Accuracy trends with respect to number of hidden layers of 3.

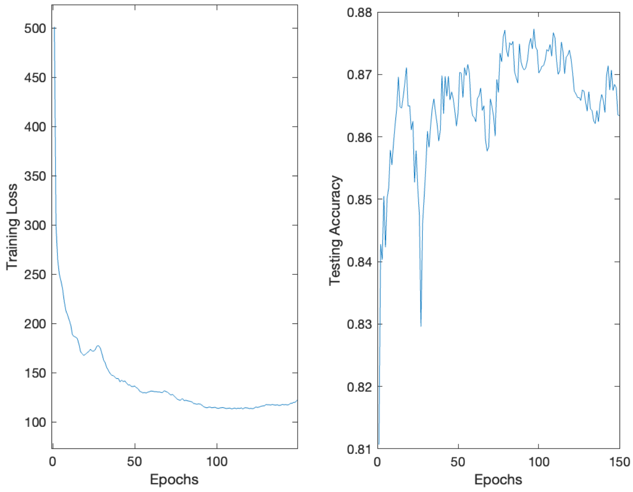

Fig 9: Training Loss and Testing Accuracy trends with respect to number of hidden layers of 5.

Adding too many hidden layers beyond what is necessary can reduce the test set's accuracy. This happens because the model starts to overfit the training data—it memorizes it rather than learning the underlying patterns. As a result, the model struggles to generalize to new, unseen data. This concept is illustrated in the image below.

The first case has just the right level of complexity, allowing it to capture the data's trend and generalize effectively. However, in the right image, the model overfits by matching the data too closely without learning the trend, making it unable to handle new data properly.

Our model encounters the same issue when the number of hidden layers exceeds the optimal amount. From our analysis, the model achieves its best performance with two hidden layers, as this is sufficient to represent the data accurately. However, adding more layers causes overfitting, leading to poorer generalization.

V. Conclusion

Results show that increasing epochs improves accuracy and reduces loss, but with diminishing returns after 150 epochs. A learning rate of 0.01 provides the best balance, while higher and lower rates lead to instability or slow convergence. The model performs optimally with two hidden layers, as adding more layers causes overfitting and reduces generalization. These findings emphasize the importance of careful hyperparameter tuning for effective and robust model training.

References

The code used to produce these results is sourced from the homework 3 on Bruinlearn.