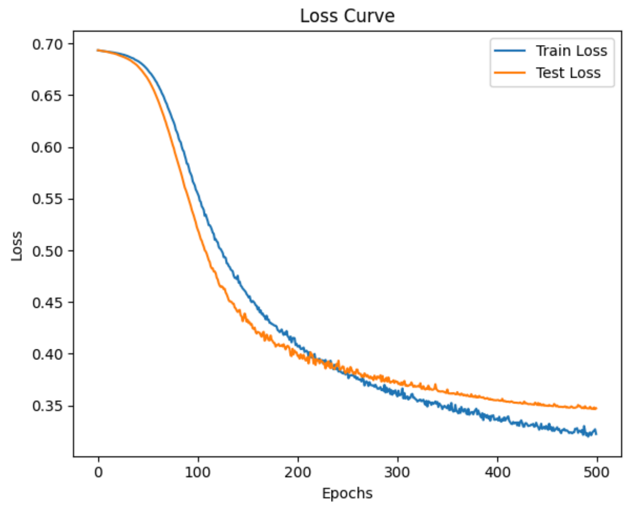
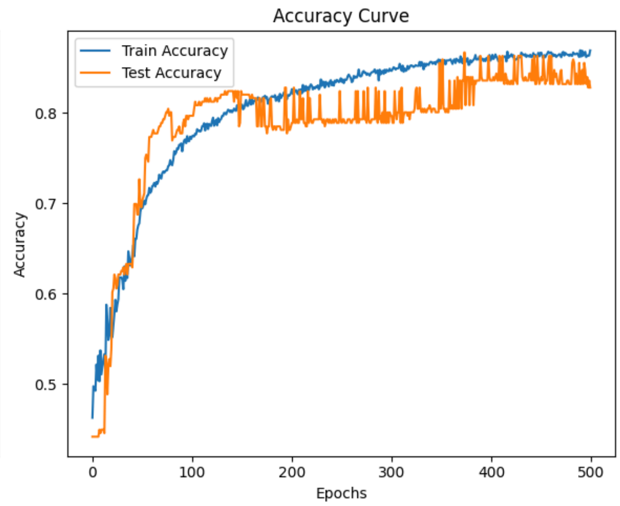
HW 1: Report - Ananya Agrawal (ananyaa2)





Loss Curve Analysis

1. Both training and test loss decrease consistently over the epochs, thus confirming that the model is learning properly.
2. Training loss consistently decreases indicating that the model keeps improving on training data.
3. Test loss follows a similar decreasing trend but stabilizes around 300/400 epochs. This suggests the model has reached its optimal generalization capability.
4. Test loss does not increase after a certain point, suggesting minimal overfitting, which is ideal.

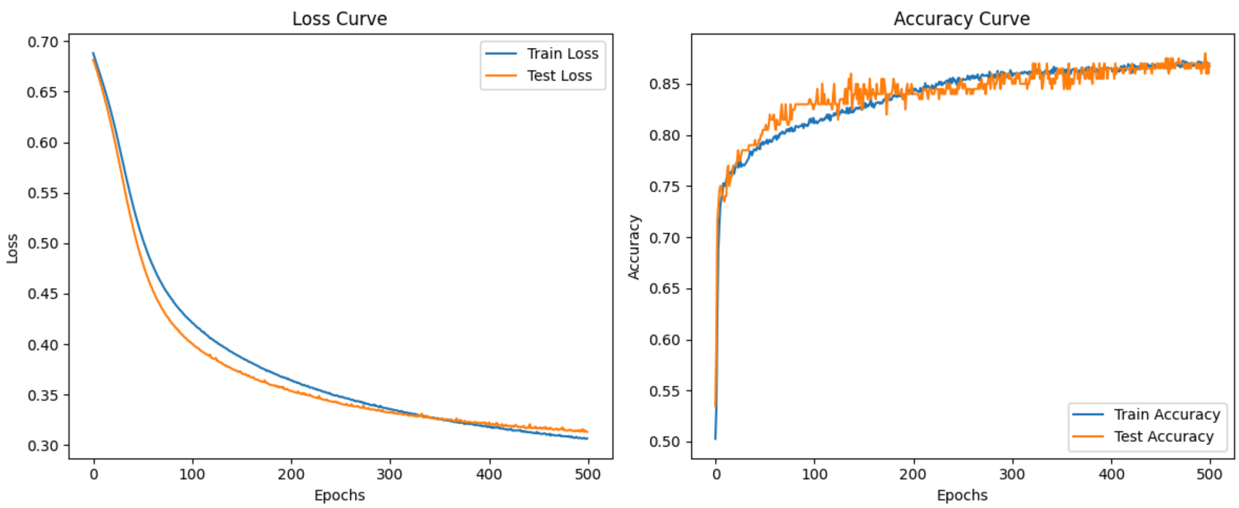
Accuracy Curve Analysis

1. Training accuracy steadily increases, reaching over 80% by the 500th epoch.
2. Testing accuracy fluctuates more than training accuracy but generally follows the same trend.
3. Test accuracy is slightly lower than training accuracy due to generalization differences.
4. A stable gap between training and test accuracy indicates a positive outcome that the model is not overfitting significantly.
5. The fluctuations in test accuracy could be due to batch variability as seen in smaller datasets.

Observations

1. The model is learning well, as observed by steadily decreasing loss and increasing accuracy.

2. Minimal overfitting is seen, which means the model generalizes well to unseen data.
3. In order to reduce the test accuracy fluctuations, we can use a lower learning rate, implement L2 regularization and improve stability, or average over multiple evaluation runs to smooth out test accuracy.
4. In case a higher accuracy is required, we can do so by adding more hidden units in the MLP, experimenting with different activation functions, or fine-tuning the learning rate.



Comparison: Manual vs. PyTorch Implementation

Loss Curve Analysis

1. Both implementations show a steady decrease in training and testing loss, thereby indicating successful learning.
2. As observed, the training losses for both implementations are nearly similar:
   nn.py (scratch implementation): Train Loss = 0.3225, Test Loss = 0.3473
   reference.py (pytorch implementation): Train Loss = 0.3067, Test Loss = 0.3132

   The PyTorch implementation has a slightly lower loss because of optimizers which are more numerically stable or due to minor differences in weight initialization.

3. Neither of the two models exhibits overfitting as such.

Accuracy Curve Analysis

1. Accuracy trends are nearly identical across implementations.
2. Comparing the accuracy values:
   nn.py (scratch implementation): Train Accuracy = 0.8692, Test Accuracy = 0.8281
   reference.py (pytorch implementation): Train Accuracy = 0.8673, Test Accuracy = 0.8700

3. As observed, PyTorch achieves a slightly higher accuracy due to built-in optimization functions which are imported as torch.nn.Linear and torch.optim.SGD, automatic weight initialization, and better numerical stability.
4. The test accuracy fluctuations as observed in the nn.py implementation are slightly higher because of manual backpropagation introducing small instabilities, and a slightly different order of operations in computing gradients.

Observations

1. Both implementations perform similarly thereby proving that the manual implementation of backpropagation is correct.
2. PyTorch model is slightly more optimized
3. There is little to no overfitting in both models.

Conclusion

<u>The manual implementation is validated successfully by the PyTorch reference.</u>