Assignment 2 Report

Brayden Edwards
February 9, 2025

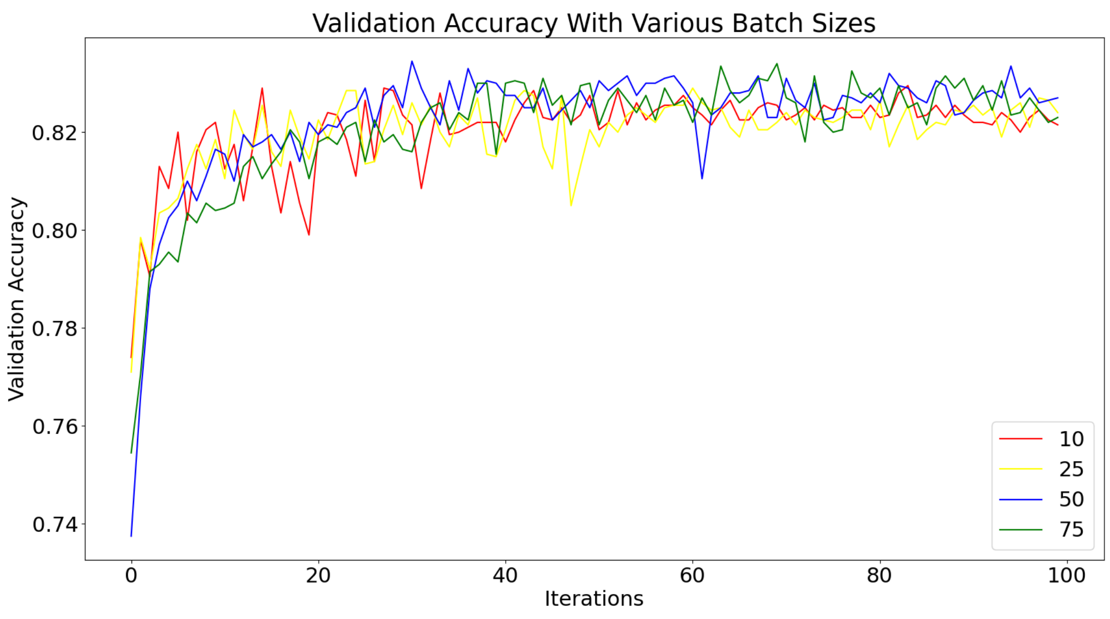
Introduction

The following report showcases my Neural Network's accuracy on the validation data with various hyperparameters. Along with that, I provide a final configuration of hyperparameters that I believe balances validation accuracy and loss.

Test Accuracy With Different Batch Size

The following chat was produced using these hyperparameters:

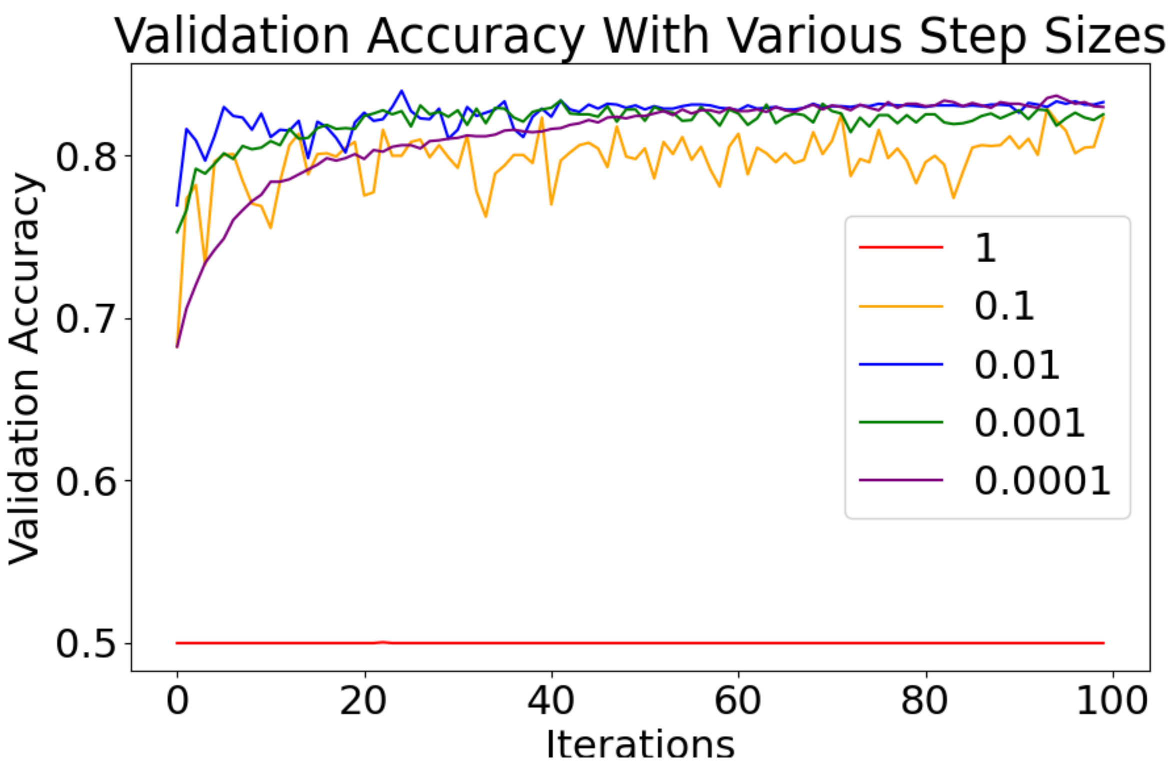
- max_epochs = 100
- step_size = 0.001
- number_of_layers = 3
- width_of_layers = 64
- weight_decay = 0.0001
- momentum = 0.8



Test Accuracy With Different Learning Rate

The following chat was produced using these hyperparameters:

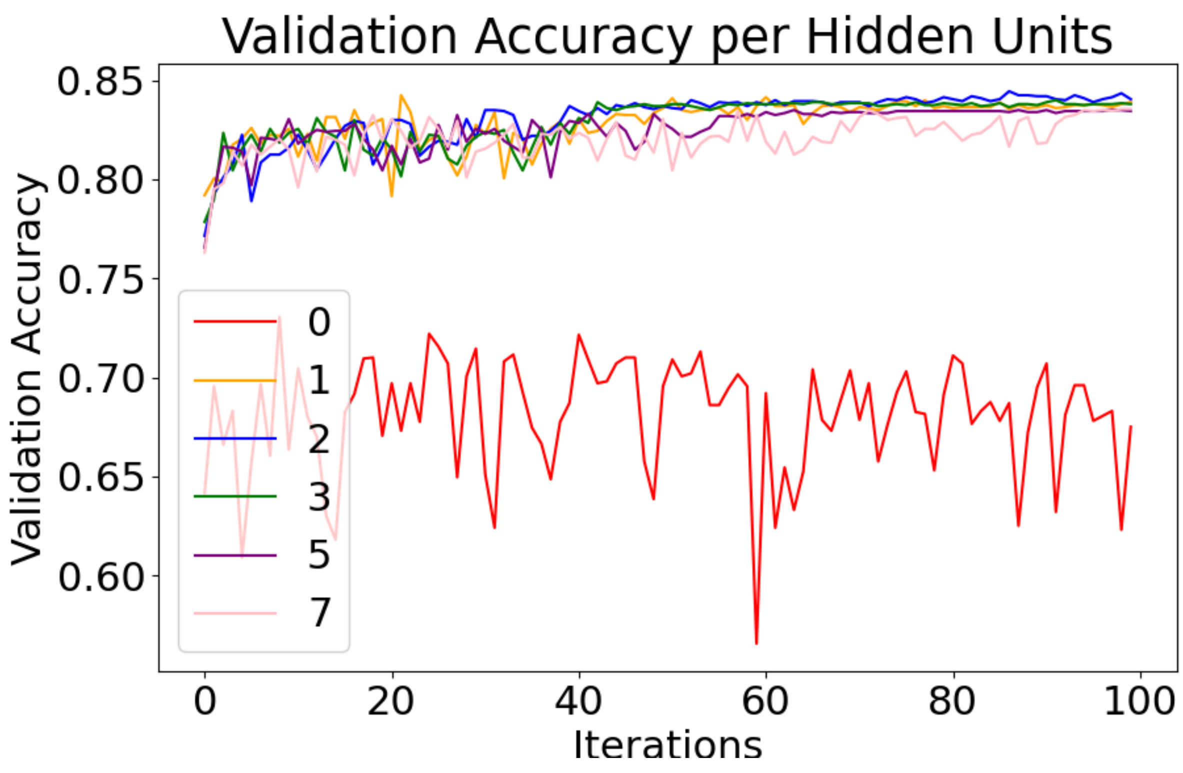
- max_epochs = 100
- number_of_layers = 3
- width_of_layers = 64
- weight_decay = 0.0001
- momentum = 0.8
- batch_size = 50



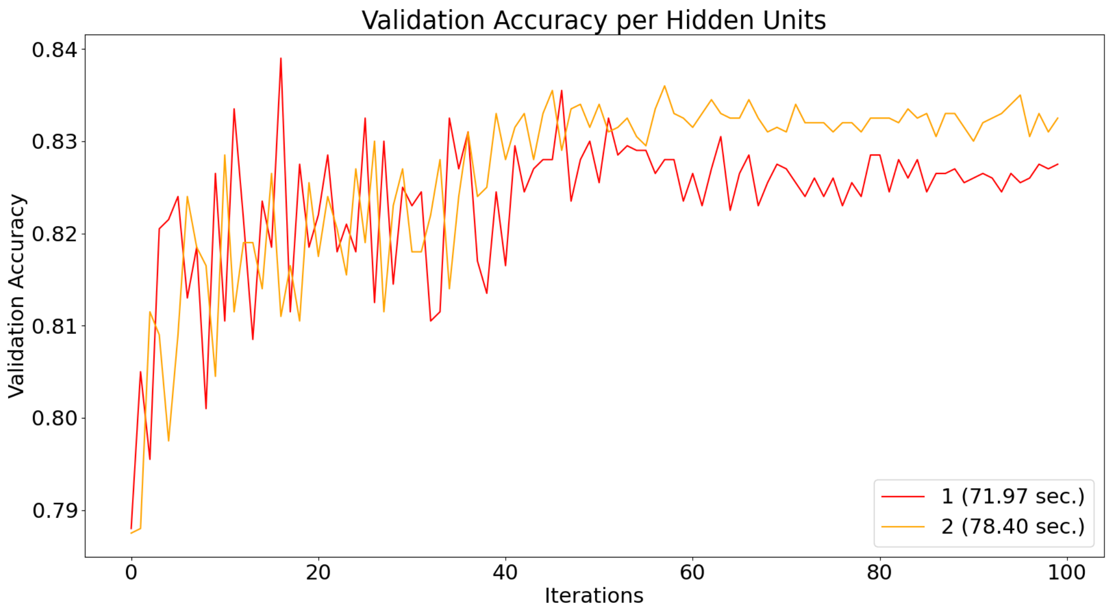
Test Accuracy With Different Number of Hidden Layers

The following chat was produced using these hyperparameters:

- max_epochs = 100
- step_size = 0.01
- width_of_layers = 64
- weight_decay = 0.0001
- momentum = 0.8
- batch_size = 50



Given the chart above, we see that 0 hidden layers performs as expected and is unable to properly fit the training/validation information, as a linear model lacks the needed complexity. Additionally, the model with 7 hidden layers is too volatile, possibly meaning the model is overfitting the training data. Looking at the chart, the best number of hidden layers looks to be either 1 or 2. While 2 hidden layers seems to be the preferred choice because the accuracy is slightly higher, I performed a time analysis to decide if the time trade-off of 2 hidden layers is worth the additional accuracy of roughly 0.01 when compared to 1 hidden layer.



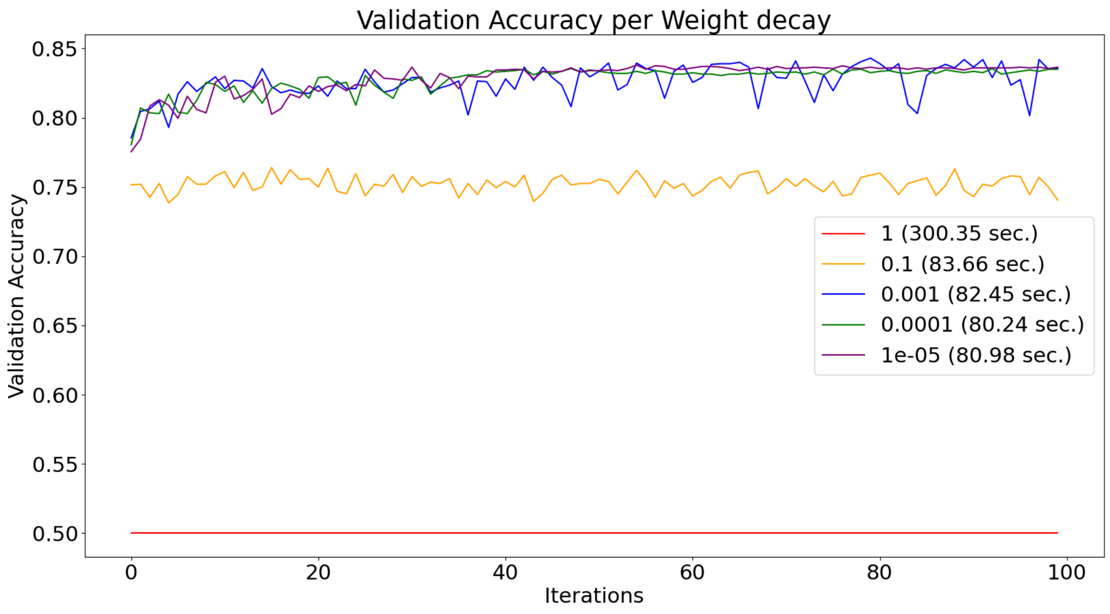
Based on the chart, we can see that 2 hidden layers (at 78.4 seconds) slightly outperforms 1 hidden layer (at 71.97 seconds). A difference of 6.43 seconds makes the question on efficiency negligible. Therefore, I have opted to use 2 hidden layers for future tests.

In addition to the testing specified in the assignment directions, I opted to test weight decay, momentum, and width of layers.

Test Accuracy With Various Weight Decays

The following chat was produced using these hyperparameters:

- max_epochs = 100
- step_size = 0.01
- number_of_layers = 3
- width_of_layers = 64
- momentum = 0.8
- batch_size = 50



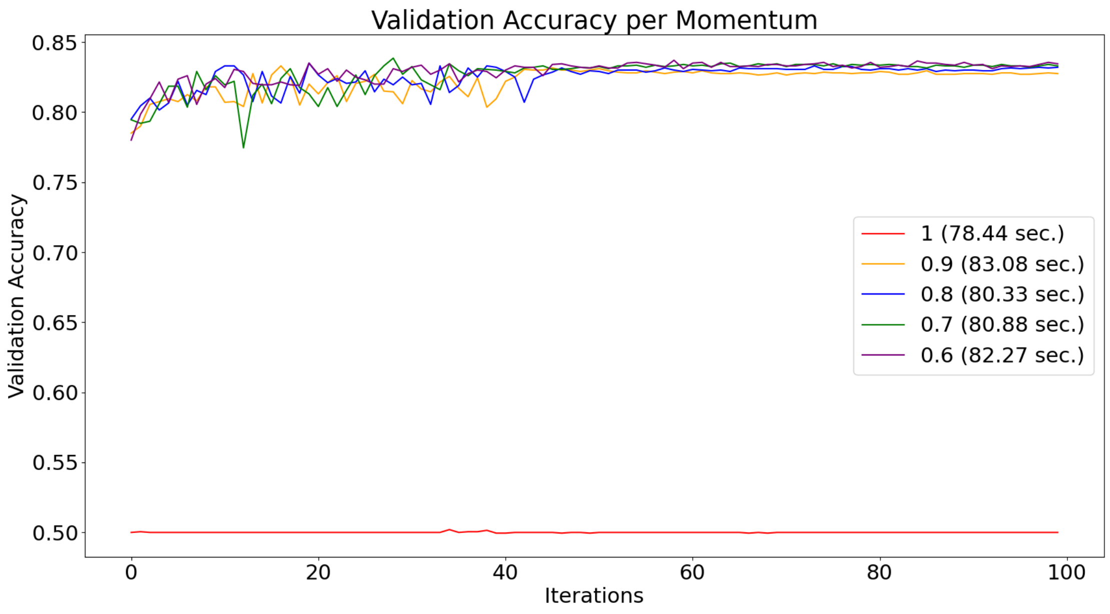
High weight decay, as shown here with the red and yellow line, causes the model to be overly penalized and underfit to the training and testing data. This is why we see a lower accuracy for weight decay of 1 and 0.1.

Based on the chart, the only weight decays that provide comparable accuracy is 0.001, 0.0001, and 0.00001. A weight decay of 0.001 (blue line) provides too much volatility. Weight decay of 0.0001 and 0.00001 prove to be the best weight decays and take roughly the same time to train. However, the weight decay of 0.00001 seems to provide slightly better accuracy, so I have chosen this value for future tests.

Test Accuracy With Various Momentums

The following chat was produced using these hyperparameters:

- max_epochs = 100
- step_size = 0.01
- number_of_layers = 3
- width_of_layers = 64
- weight_decay = 0.00001
- batch_size = 50

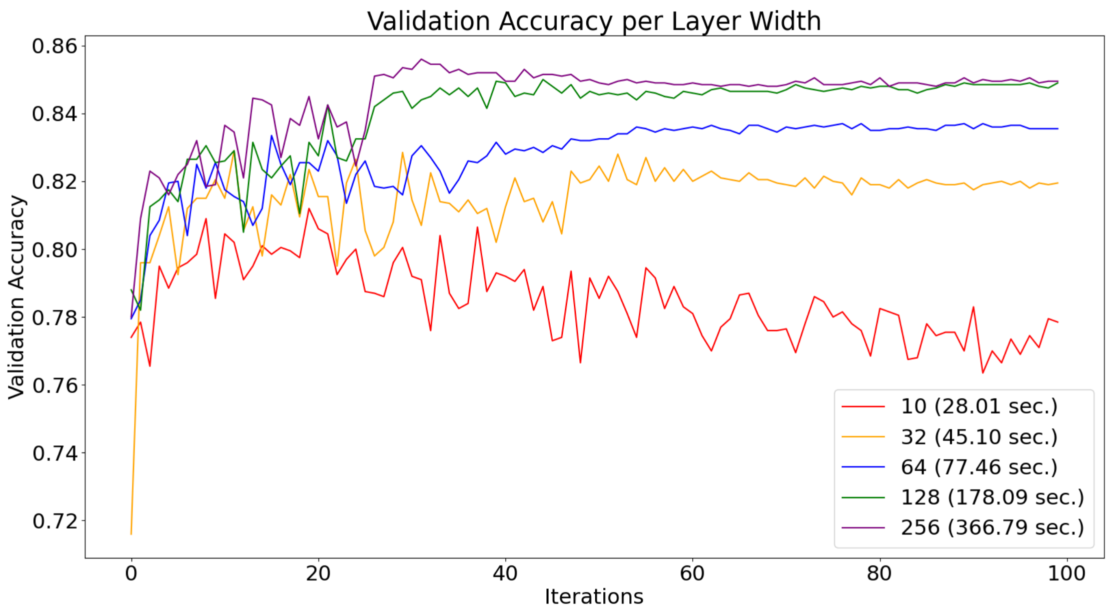


We can see here that the momentums of 0.9, 0.8, 0.7, and 0.6 are all roughly equal. Because they are roughly equal in accuracy, I am opting to utilize a momentum of 0.8 because the time taken to train is lower.

Test Accuracy With Various Layer Widths

The following chat was produced using these hyperparameters:

- max_epochs = 100
- step_size = 0.01
- number_of_layers = 3
- weight_decay = 0.00001
- momentum = 0.8
- batch_size = 50



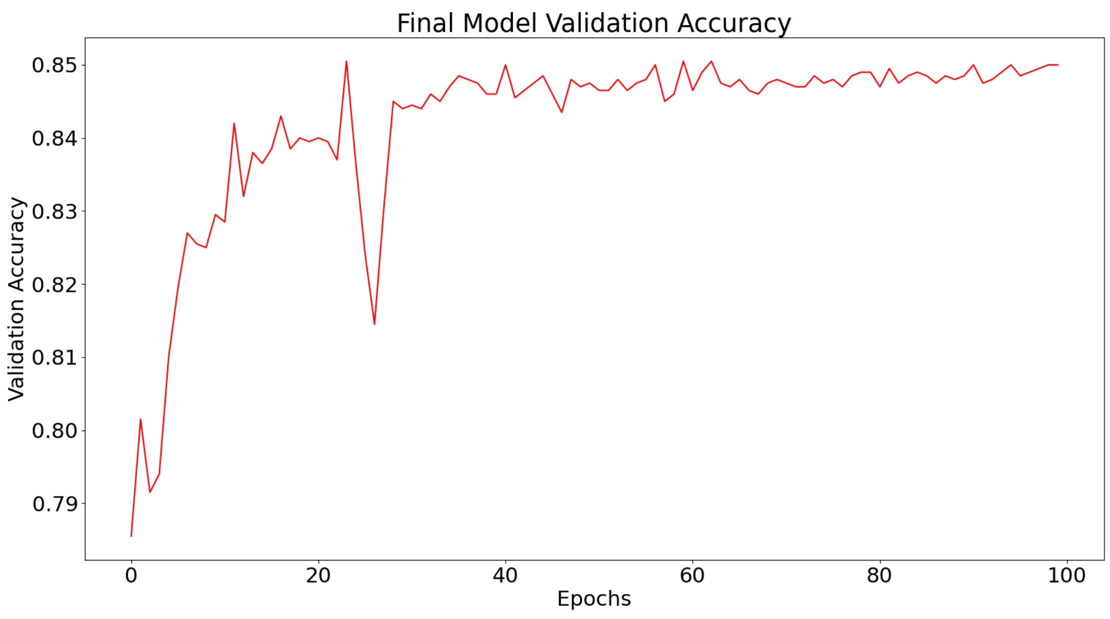
I prefer the 128 layer width given the results above. It slightly underperforms the model with a layer width of 256 while taking less than half the time to train. When layer width is 64, It takes less than half of the time to train as the layer width of 128 but performs noticeably worse. Therefore, the best trade-off of accuracy and time, while prioritizing accuracy, is a layer width of 128.

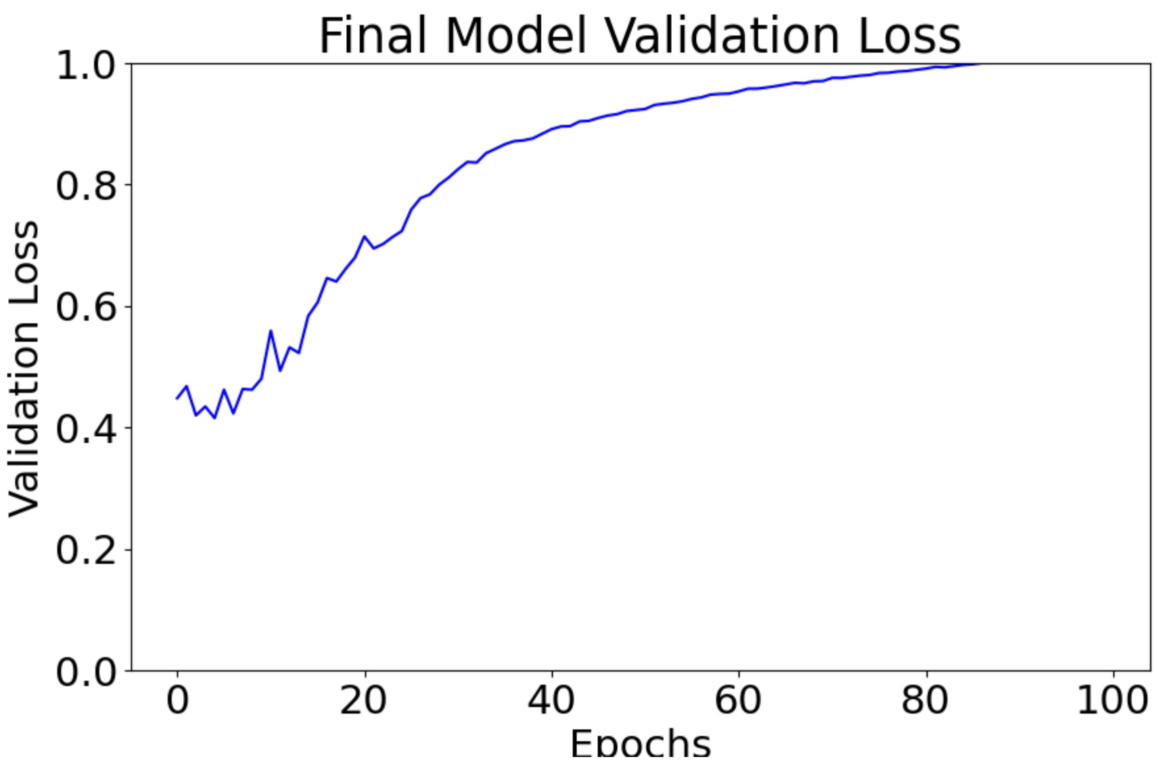
Final Model Test Accuracy

My final model has the following hyperparameters, which were selected from the results above:

- max_epochs = 100
- step_size = 0.01
- number_of_layers = 3
- width_of_layers = 128
- weight_decay = 0.00001
- momentum = 0.8
- batch_size = 50

Here is our final output of accuracy for our validation data.





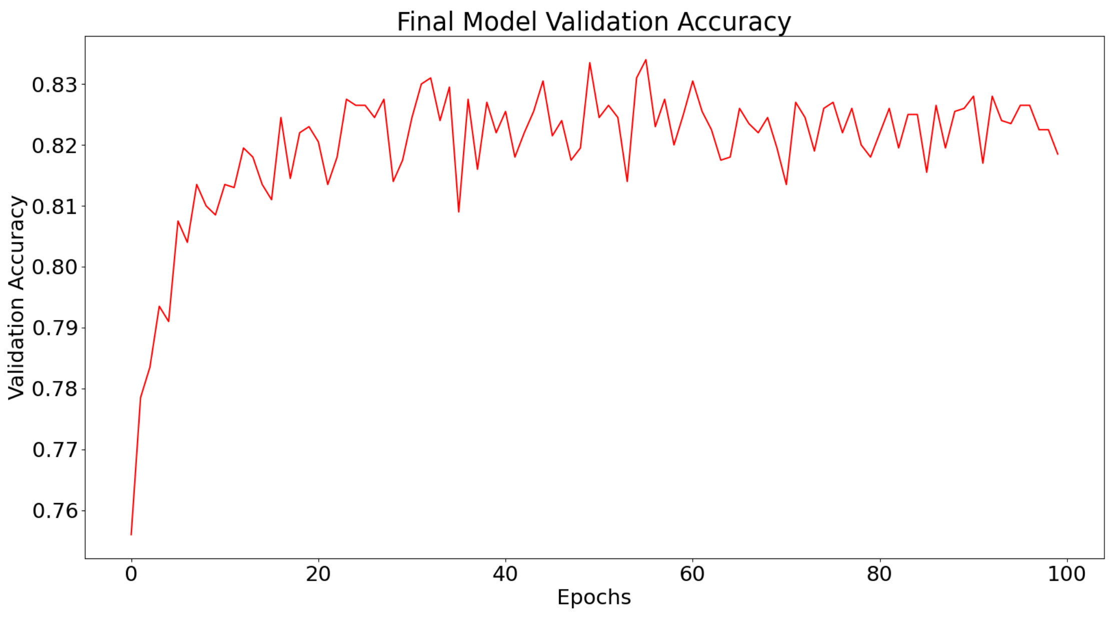
The validation loss suggests that the model is overfitting the training data causing it to struggle in generalizing to unseen data. To address this, further hyperparameter turning is needed, including assessing learning rate, width of layers, and weight decay values.

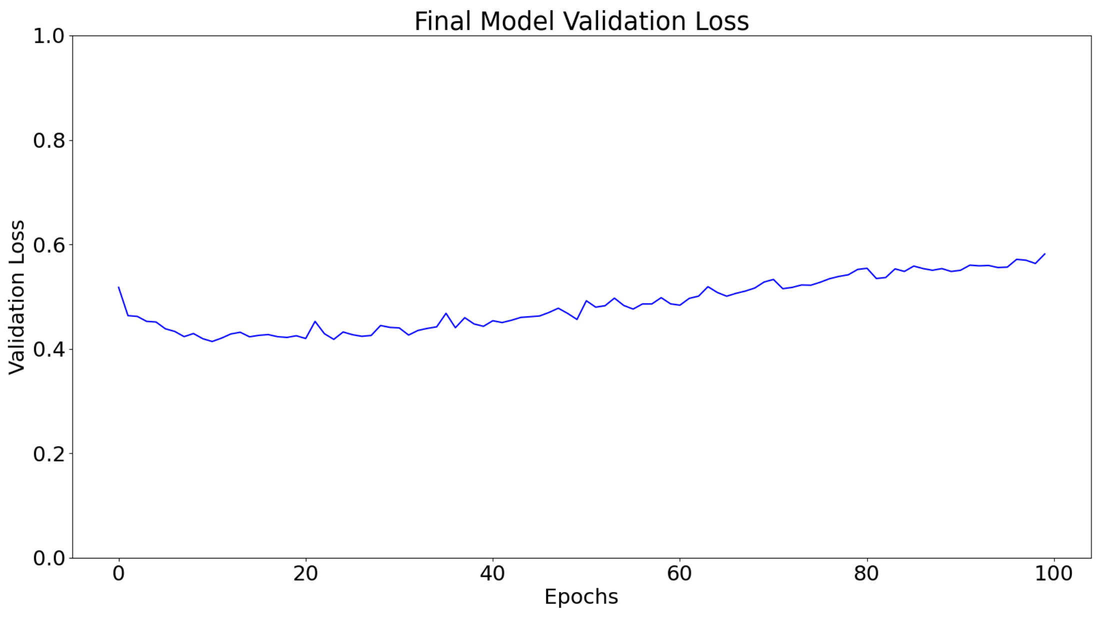
I was able to get the validation loss much lower through small changes in the hyperparameters, but this harmed the accuracy of my model on the validation data. This was done by choosing the hyperparameters that made my model less accurate. Here are the hyperparemeters for the lower loss model:

- max_epochs = 100
- step_size = 0.001

- number_of_layers = 2
- width_of_layers = 32
- weight_decay = 0.001
- momentum = 0.8
- batch_size = 32

Here are the charts for this model.





You will notice larger volatility in the validation accuracy but a much better model loss on the validation data.

With that, I believe that my new model is better in spite of the lower validation accuracy because the loss curve is much better.

Conclusion

More testing is needed to obtain the optimal hyperparameters for a balance in validation accuracy and loss. While the heuristic approach I used above saw vast improvements in the model, further testing systematic testing is needed.

Final Accuracies:

Optimally Accurate Model: 85%
Optimal Loss Model: 84.4%