Feed forward neural networks and their applications in motor validation

Nicholas Parise 7242530

***Abstract*—This report investigates the classification performance of a multi layer feed forward neural network trained on motor acceleration FFT data. Several experiments were conducted to determine how different factors influenced generalisation and accuracy. These experiments included: dataset size, number of hidden nodes, use of multiple hidden layers, dynamic learning rates, momentum, and softmax at the output layer. A stratified three fold training method was used to make sure test and training sets had the correct proportions of good and bad motors. To measure the performance, the NN was randomly initialised and ran with certain hyper parameters, this data was then averaged, ranked and compared. Results show that larger hidden layers generally reduced the amount of training needed, but caused test accuracy to plummet. Experiments were then run to see if a five layer NN would increase classification, experiments were run with and without softmax. While they were slightly better than the three layer NN, it was not marginally better. Overall the model had classification rates of 74**

I. Introduction

FEED forward neural networks are commonly used for pattern recognition because of their ability to find non linear classifications. In this report, FFT motor data is used to determine if a motor is good or bad. The goal is to determine what hyper parameters lead to the most successful network that has the highest classification rate. To do this multiple experiments are performed tweaking each parameter to maximize success. Parameters include: number of hidden nodes, number of hidden layers, dynamic learning rate, momentum and the use of softmax. All tests are performed using a stratified 3 fold approach to maintain the ratio of good and bad motors across all folds. In addition, data points which are extremely similar are always kept in the training set to increase test accuracy. For each of these experiments the network was trained 20 times to ensure outliers do not skew the results, these results were averaged, ranked and compared. By systematically changing one factor at a time, we will be able to identify the best parameters and choose the values that work the best.

II. Part A

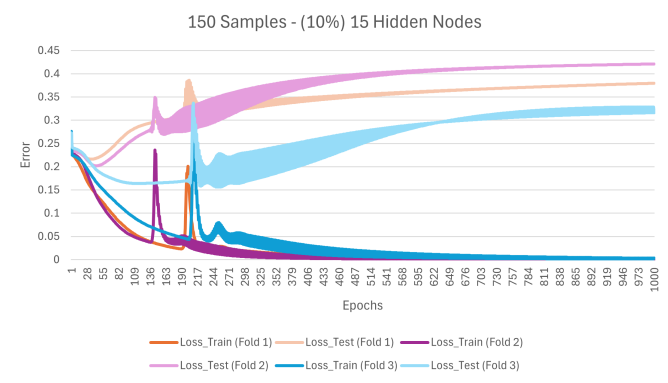
The Neural network is in the form of n - n/4 - 2. Where the input layer is the number of samples in the dataset, the hidden layer is ¼ of the size of the input layer, this increases training speed and helps to reduce overfitting. The output layer is two neurons, one corresponding to a good motor the other corresponding to a bad motor.

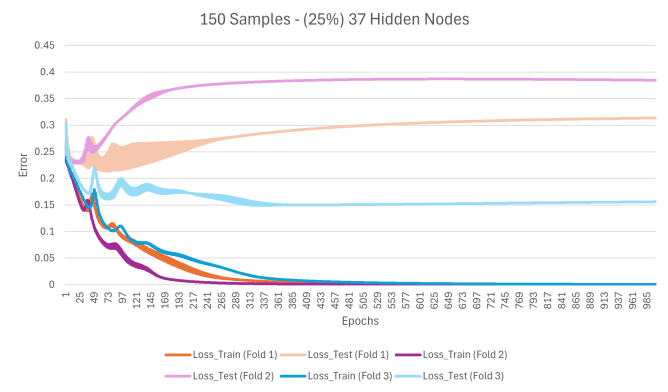
The activation function is sigmoid, it is easily derivable and simple. The mean squared error is used as the loss function. The learning rate is set as 0.1 initially and is dynamically decaying each epoch.

To aid in convergence a few strategies are used. First the inputs are normalised so large values don't destabilise the network, second the weights are initialised in a small range of [-0.25, 0.25] so the values are close to the optimal range of the sigmoid function. Finally a dynamic learning rate with a slowly decaying rate to allow the network to not overshoot its convergence.

III. Part B

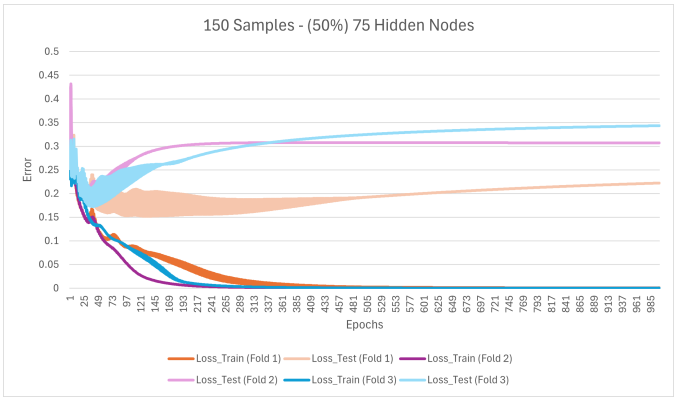
Findings

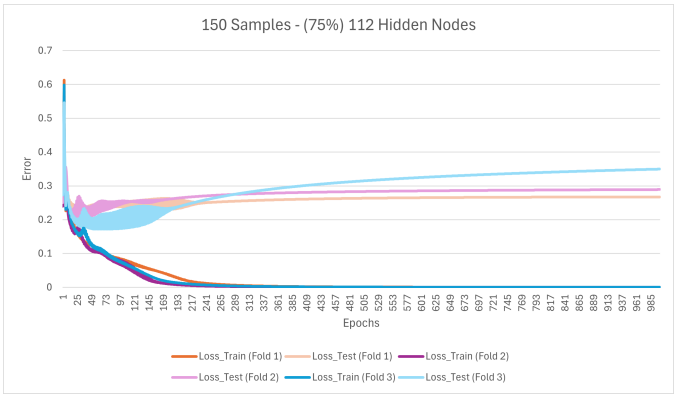


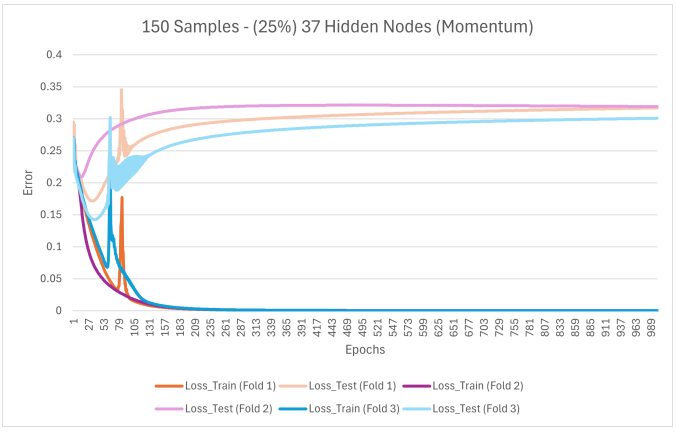


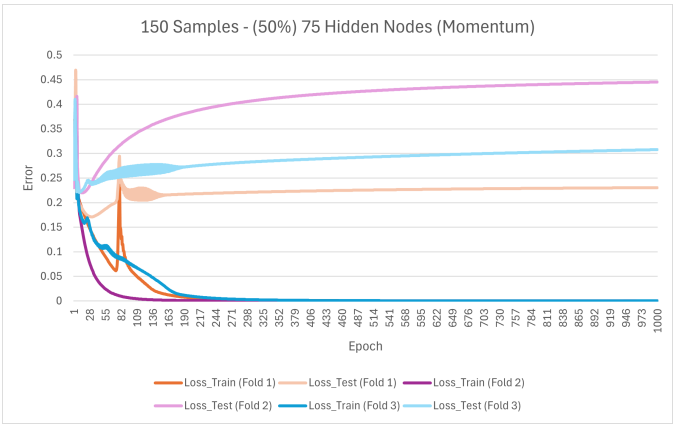
A. Setup

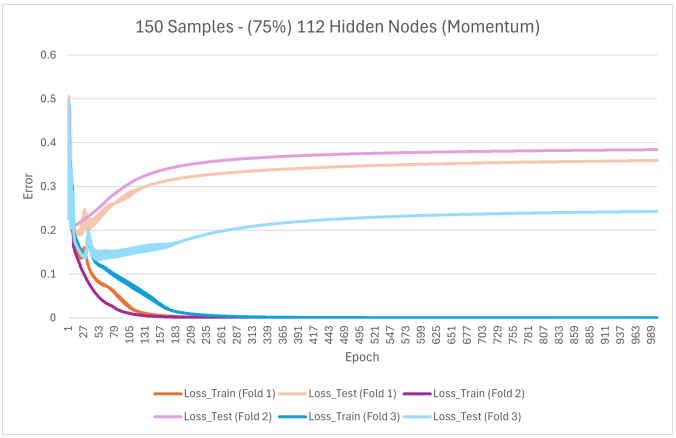
This experiment was performed to determine how well a 3 layer neural network with different hidden layer sizes affect convergence, generalisation and classification. The 150-sample motor dataset file was used along with a 3-fold cross validation. For these experiments a dynamic learning rate was used starting at 0.1 and slowly decaying.









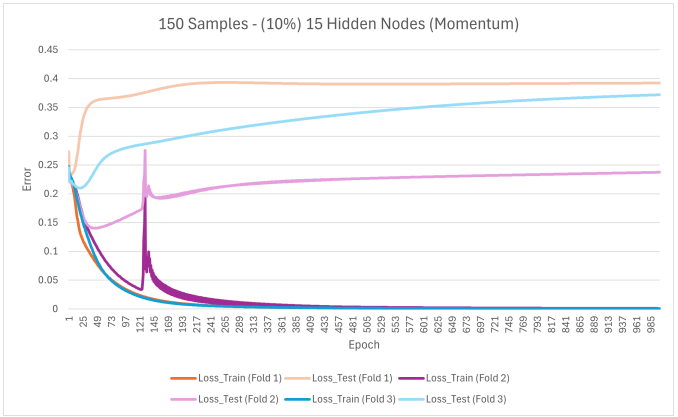


B. Observations

Convergence speed on the training set was very dependent on the hidden layer size. Where larger hidden node sizes, for instance the 112 hidden node size approached 0 at ˜200 epochs while when the hidden node had a size of 15 it approached 0 at ˜450 epochs

Overall the best generalization of all tests were around the same, all round 0.15 error rate for the test sets, with the 112 hidden node dataset hovering closer to 0.18 at a minimum. Overall the trend in the data suggests that while a higher hidden node size allows for faster training it doesn't allow for better generalization, which was the expected result.

IV. Part C



A. Setup

This experiment was performed to determine how well a 3 layer neural network with different hidden layer sizes and momentum affects training time.

B. Observations

This experiment was very successful in showing the use of momentum. Not only did it greatly increase training speed it helped to reduce oscillations. For instance In the 15 hidden node experiment got to an error rate of 0.05 around the 110th epoch and afterwards had lots of oscillations that took a long time to stabilise, conversely the same experiment now with

momentum got to an error rate of 0.05 around the 70th epoch and afterwards is had almost no oscillations at all.

V. Part D

A. Setup

This experiment was performed to determine which training set has the best classification. The three selected datasets are 64, 150 and 1000 sample files. Each data file then cycles through four weights 17%, 25%, 50% and 75% of the input datapoint size. This was performed 20 times each, the data was then averaged and sorted.

B. Results

filename	hidden_size	Test Accuracy
L30fft_64.out	10	0.75625
L30fft150.out	25	0.75069467
L30fft_64.out	48	0.74930533
L30fft150.out	75	0.745139
L30fft_64.out	32	0.74013867
L30fft150.out	37	0.738472
L30fft150.out	112	0.72138933
L30fft_64.out	16	0.721389
L30fft1000.out	166	0.7125
L30fft1000.out	250	0.69583333
L30fft1000.out	500	0.69375
L30fft1000.out	750	0.68333333

Before averaging the min and max was saved

	filename	Hidden size	Test Accuracy
L30fft_64.out	10	0.8	Max
L30fft1000.out	750	0.6625	Min

C. Observations

Overall the 64 sample and 150 sample datasets produced almost test accuracy, with both achieving a 75% test accuracy. The 1000 sample dataset consistency performed worse despite it containing more information.

This is most likely due to the 1000 sample neural network becoming overfit much faster than the other datasets due to its higher hidden node size. So while the larger network reduced training loss faster their performance did not improve the test loss at the same rate. This is also evident that our minimum achieved was with the highest amount of hidden nodes with an accuracy of 66% with 750 hidden nodes.

VI. Part E

A. Setup

This experiment was performed to determine how increasing the neural network to 5 layers impacts classification accuracy. This experiment changed to tanH to improve training in a larger network. Because of the change in activation function the learning rate was changed to 0.01 to account for this.

Only the 150 sample file was selected. The number of hidden nodes test per layer were: 5, 10, 15, 20, 25, 30 Each configuration was performed 20 times each, the data was then averaged and sorted by accuracy.

filename	Hidden size	Test average
L30fft150.out	5	0.76375
L30fft150.out	10	0.749722333
L30fft150.out	15	0.763194667
L30fft150.out	20	0.751805333
L30fft150.out	25	0.752917333
L30fft150.out	30	0.742916667

filename	Hidden size	test accuracy	
L30fft150.out	30	1	Max
L30fft150.out	15	0.6	Min

B. Results

Before averaging the min and max was saved

C. Observations

These results indicate that increasing layer size did not produce a significantly increased generalisation performance. While every value is higher than its counterpart with 3 layers, this increase is only around ˜1.

The increase in hidden layers only seemed to noticeably increase variance in the test set. While one run rescued 100% accuracy, the others in the same category only had an average success rate of 74%.

This experiment shows that an increase of depth does not guarantee an increase in generalization especially on datasets with few datapoints.

VII. Part F

A. Setup

This experiment was performed to determine how softmax in a 5 layer feed forward neural network to 5 layers impacts classification accuracy. Compared to one without softmax. Only the 150 sample file was selected. The number of hidden nodes test per layer were: 5, 10, 15, 20, 25, 30

Each configuration was performed 20 times each, the data was then averaged and sorted by accuracy.

B. Results

filename	Hidden size	Test average
L30fft150.out	5	0.758888667
L30fft150.out	10	0.752916333
L30fft150.out	15	0.752638667
L30fft150.out	20	0.736388333
L30fft150.out	25	0.746388667
L30fft150.out	30	0.754999667

Before averaging the min and max was saved

filename	Hidden size	Test accuracy	
L30fft150.out	30	1	MAX
L30fft150.out	5	0.6	MIN

C. Observations

This experiment did not have a meaningful increase in test accuracy. While individual runs occasionally spiked to 100% accuracy, the overall behaviour remained the same with an accuracy range of 74% to 76%.

For this dataset softmax was not the issue when it came to generalization and accuracy. Instead it is a systemic issue with the architecture of the neural network or lack of data to train on.

VIII. Conclusion

Across all experiments, test accuracy remained generally consistent hovering around 75%, regardless of number of layers and size of those layers. Larger hidden layers did reduce the amount of time to train, but did not meaningfully increase the classification rate. At the same time smaller hidden layers took more time to train, but also did not meaningfully increase the classification rate either. Increasing depth to five layers and enabling softmax at the output layer, did not produce meaningful gains either. This suggests that the limiting factor in this problem is not the network but rather the dataset itself. The input values are very similar and the output is not easily mapped to these slight changes in input. Overall, the network plateaus early, all experiments start off very promising, but the training and test sets always diverge once they hit a mean square error of 0.2. The training set can continue training but the test set will almost never get below 0.15 error. To improve performance more data will be needed to train on.

References

[1] Koyiljon, Valiev, *A gentle introduction to feed-forward neural networks with NumPy* July 2024