Project 3

Joshua Aney joshua.aney@student.montana.edu
Owen Cool owen.cool@student.montana.edu
Nicolas Perl nicolas.perl@student.montana.edu

Abstract

Multilayer feedforward neural networks are a foundational architecture in the field of machine learning, designed to solve complex problems through hierarchical feature learning. Unlike single-layer perceptrons, multiple hidden layers are used to approximate more complicated non-linear functions, making them suitable for a wide range of applications. Our experiment explored the relative performance of feedforward networks with 0, 1 and 2 hidden layers when tested on 6 different datasets from the UCI machine learning repository, including both classification and regression problems. Using 10-fold cross-validation, we found that the neural networks were often a suitable approach to the classification and regression problems. However, we also found that performance varied significantly with the number of hidden layers and different datasets tended to perform better or worse depending on the relative complexity of the model compared to the problem being addressed.

1 Introduction

A solution for the limitations of linearly separable rules are networks consisting of multiple interconnected perceptrons, specifically in our case multilayer feedforward networks. Feedforward refers to the flow of information along the directed edges of the network. All nodes are laid out in layers, where each node in one layer is connected to all the nodes in the next layer. Each "hidden" layer (layers that are neither inputs to the network nor outputs from the network) incorporates a sigmoid activation function; the output node(s) use different activation functions depending on the problem task, in our case either softmax for classification or a linear activation function for regression.

To approximate a decision rule, the network is trained in three steps until convergence:

1. During the forward phase, the inputs are propagated forward through the network to generate the output, whilst the weights of the network are fixed. The equation for this process is shown in equation 1, with z_{kh} equal to the value for a given node h on layer k (and $z_{0j} = x_j$, our input), w_{khj} equal to the weight from a given node j on layer $k-1$ to the node h on layer k, and $f(\cdot)$ equal to the activation function for that node:
$$z_{kh} = f(\sum_j w_{khj} z_{k-1,j}) \tag{1}$$
2. During the backward phase, the error is calculated at the end of the network and then propagated back towards the input. Its propagation through the network is given in

the equation 2 below, with e_kh equal to the error on a node h on layer k:

$$e_{kh} = \sum_{l} e_{k+1,l} w_{k+1,lh} \tag{2}$$

3. The weights are updated following the backpropagation process, using the values generated for each node during the forward phase and the error values generated for each node during the backward phase. The equation for the weight updates is shown as equation 3, with η equal to our learning rate:

$$\Delta w_{khj} = \eta e_{kh} f'(z_{kh}) z_{k-1,j} \tag{3}$$

Through this process, we essentially make a prediction for a given instance using our current weights in the neural network, get our error at the prediction, and assign credit or blame to any node h on the hidden layers by multiplying each of the errors of the following layer by the corresponding weights from h to those nodes, thereby weighting the following node's errors by h's "contribution" to those errors. We then get the weight updates using those errors and gradient descent.

2 Problem Statement and Hypothesis

Our experiment was conducted on 6 datasets from the UCI Machine Learning repository, namely "Wisconsin Breast Cancer," "Glass Identification", "Small Soybean," "Forest Fires," "Relative CPU Performance," and "Abalone." Our research focused on the question, "How do densely-connected feedforward networks algorithms perform on classification and regression problems with 0, 1 or 2 layers?"

We hypothesized that the amount of data available will have a significant impact on the performance of the algorithm if the individual datasets consist of a large amount of model parameters. Furthermore, datasets with good class separation and less complex problem statements should perform very well with less complex models (i.e. 0 layers).

We hypothesized that the neutral network would perform well on the Abalone dataset due to its normal distribution without significant skew to the left or right, many data entries and relatively few outliers. The algorithm using two layers should perform best.

Another part of our hypothesis is that the NN algorithm would perform very well on the breast cancer data, as it has numerous entries compared to its number of features, many examples of each class and its good class separation.

Our hypothesis also includes that, when working with the Soybean dataset, the neural network's performance with multiple layers should be impacted by the low amount of data entries available. Other than that, it should be a simple target to predict and perform very well.

We hypothesized that the performance of the NN algorithm on the Glass dataset would be influenced by the high amount of classes of the dataset in contrast to its low amount of entries. Due to the dataset's moderate complexity level, the algorithm should perform reasonably well with 1 or 2 layers.

We further hypothesized, that the algorithm would perform poorly on the Machine dataset due to its skew and the low amount of data entries versus the amount of features. Therefore, even thought the problem would need more layers due to its higher complexity, training that kind is not possible with good performance output.

We hypothesized that the NN would perform the worst on the Forest Fire dataset due to the impact of its big dimensionality, taking into account the performed one-hot-coding of the days and months features. In addition to that, the impact of features with only minimal relevance to the actual prediction values might also have a negative impact.

3 Experimental Approach and Program Design

We approached the problem stated above in a very sequential manner. Our first task was to create a hypothesis based on our prior knowledge for each of the datasets. We then created a design document that would help us formulate our approach on how we were going to attack the tasks at hand. This document contained instruction and design figures for how we were going to structure our code and test our hypotheses. This was then followed by understanding and coding our implementation of our neural network model. This model was meant to be generalizable to have an arbitrary number of hidden layers with an arbitrary number of nodes in each hidden layer.

To train our neural network model, we use a backpropagation method. This is done by:

1. Completing a feed forward pass to get a predicted value on the last layer. This is done by following the equation 1 in the introduction.

2. Next, the error for each node is calculated by following equation 2 in the introduction.

3. Finally, the weights are updated by following equation 3.

Our code is then set up to have a backpropagate function that will do all the work to complete one forward and backward pass for a singular datapoint. The backpropagate function will then be called by a train function that will call the backpropagate function on a training dataset. This will be repeated over a certain number of epochs until the performance on the holdout fold decreases or we get to the max number of epochs. Once the network is trained, you can then call the prediction function for the network, and it will return the network's prediction for the certain datapoint.

Outside of the network, we have our datasets classes that will do all the preprocessing of the data, and it will have functions that will get the data or the labels of that specific dataset.

We also have a set of cross validation functions, fold functions, and hyperparameter tuning functions that will complete all the necessary steps to tune the network and to get the final metrics of the network.

All of these functions will work together to pass the data to the network, tune the network, and get the final metrics of the network.

4 Experiment Results

The models with zero and one hidden layers performed well on the Breast Cancer dataset. The model two hidden layers performed drastically worse than the previous two models. The drop in precision and accuracy was very similar when moving to a two hidden layer model.

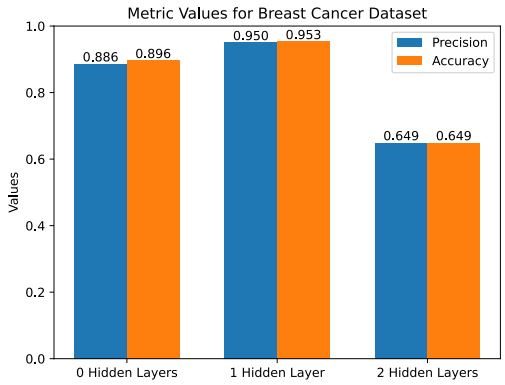


Figure 1: This bar graph represents the different average values of the metrics collected when testing the Breast Cancer Dataset with 0, 1 and 2 hidden layers.

Similar results were found for the Soybean dataset. It performed exceptionally well with zero and one hidden layer models, but performance decreased when going to a two hidden layer model. The drop was similar to the drop in performance on the Breast Cancer dataset.

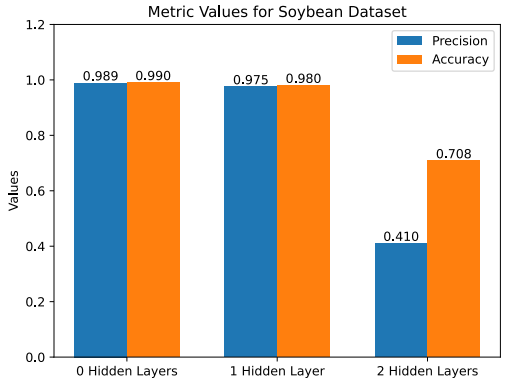


Figure 2: This bar graph represents the different average values of the metrics collected when testing the Soybean Dataset with 0, 1 and 2 hidden layers.

On the Glass dataset, performance metrics were lower across the board. Performance was very similar between zero and one hidden layer models, but on the two hidden layer model performance dropped. The drop was not as significant as the previous two datasets.

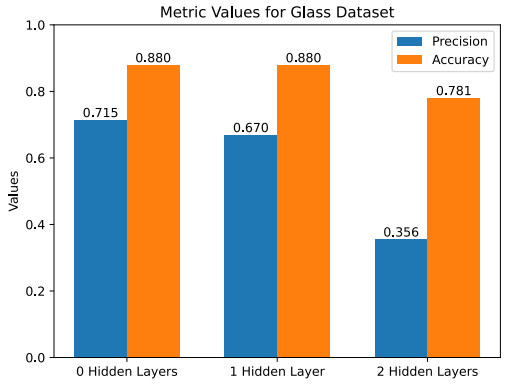


Figure 3: This bar graph represents the different average values of the metrics collected when testing the Glass Dataset with 0, 1 and 2 hidden layers.

The models with zero, one and two hidden layers performed well on the Abalone dataset. There were slight differences in the performance metrics, where the model with zero hidden layers performed best.

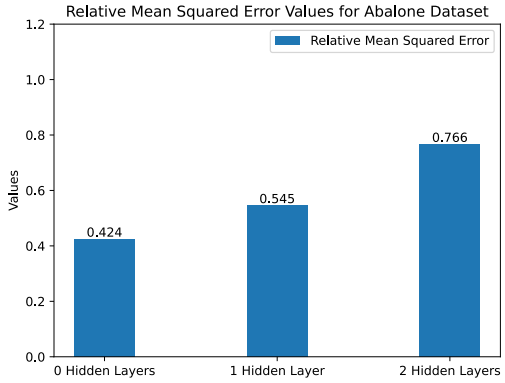


Figure 4: This bar graph represents the relative mean squared error collected when testing the Abalone Dataset with 0, 1 and 2 hidden layers.

The model with zero hidden layers performed exceptionally well on the Machine dataset. The models with one and two hidden layers performed poorly, with a drastic decrease in performance compared to the model with zero hidden layers.

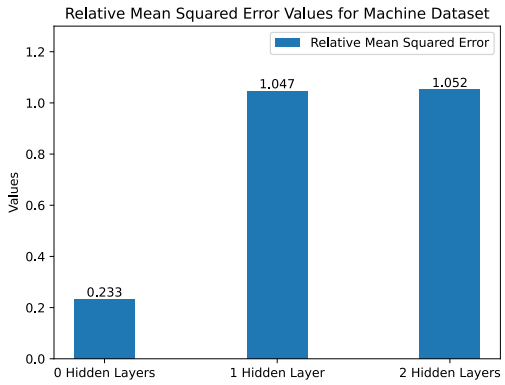


Figure 5: This bar graph represents the relative mean squared error when testing the Machine Dataset with 0, 1 and 2 hidden layers.

All the models performed poorly on the Forest Fire dataset. The models performed about the same with very minute changes in performance with zero, one, or two hidden layers.

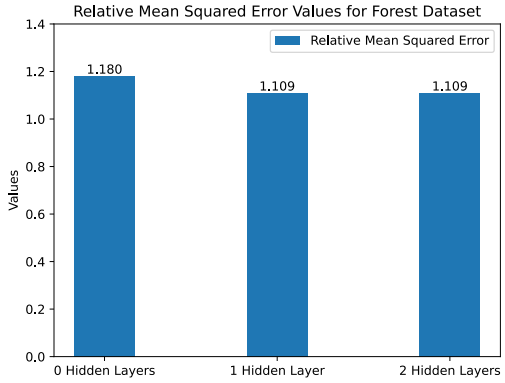
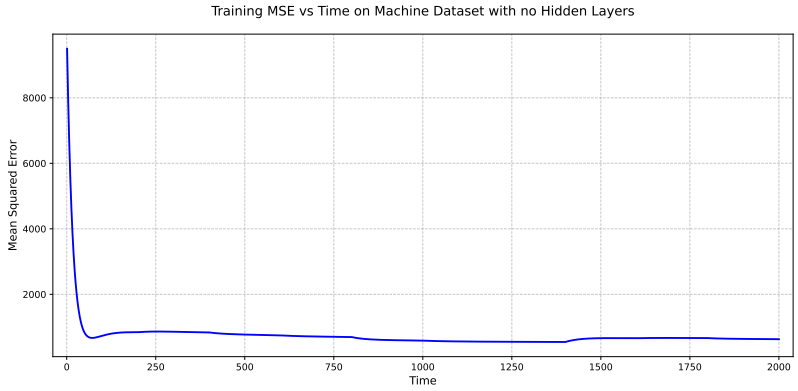
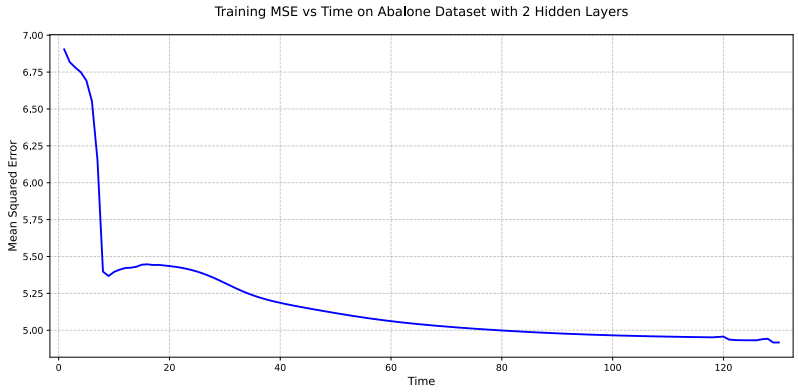


Figure 6: This bar graph represents the relative mean squared error when testing the Forest Fire Dataset with 0, 1 and 2 hidden layers.

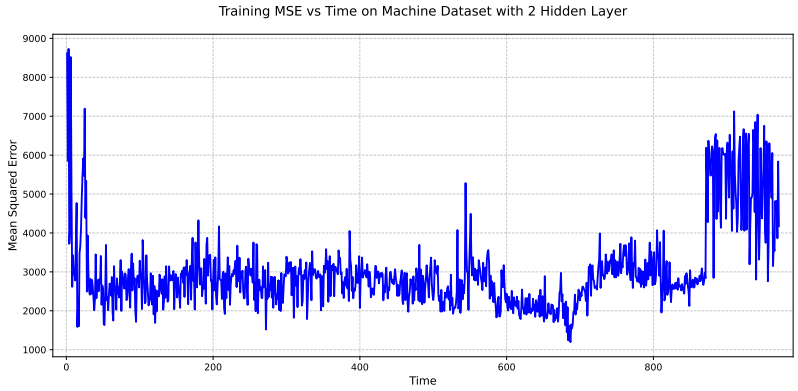
For the datasets that the model performed well on, the model had a smooth and predictable training convergence graph, but on the models it performed poorly on, it had a much more disjoint and noisy training convergence graph.



(a) This line plot represents the mean squared error over time when training the NN on the Machine dataset with 0 hidden layers.



(b) This line plot represents the mean squared error over time when training the NN on the Abalone dataset with 2 hidden layers.



(c) This line plot represents the mean squared error over time when training the NN on the Machine dataset with 2 hidden layers.

Figure 7: Training results on different datasets with different # of hidden layers

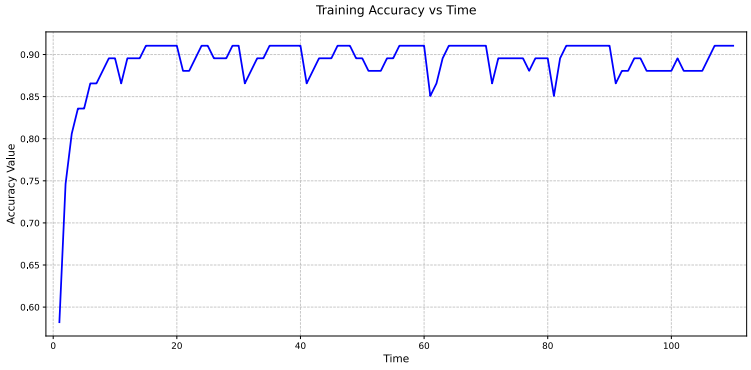


Figure 8: This line plot represents the accuracy over time when training on the Breast Cancer dataset with 1 hidden layers.

5 Discussion

As shown in figures 1-5, our neural network algorithm performed well on most of the given data sets when using only one hidden layer, while many saw diminished performance with more hidden layers. There are a few potential reasons for these general findings. Firstly, this may indicate that many of these datasets are relatively "simple" datasets, and adding more layers, therefore, only adds unnecessary complexity to the model. Further, many of our datasets are relatively small, and therefore may not contain a sufficient number of datapoints to adequately train the number of parameters present in the larger neural networks. This becomes an especially significant problem for the machine and soybean datasets, containing only 209 and 47 instances, respectively. For datasets like the Abalone dataset, meanwhile, we do not see such a significant decrease in performance from the presence of additional hidden layers (though we do see some decrease in performance), likely due to its relatively large 4177 instances.

As shown in figure 6, however, the neural network failed to outperform the null model with any number of hidden layers on the forest dataset. In addressing these findings, there are a variety of issues worth noting. Firstly, after one-hot encoding, this dataset has 30 dimensions. This causes the so-called "curse of dimensionality" to interfere badly with the performance of this dataset. Further, the problem itself may simply be difficult to adequately predict with the variables present, no matter how well-trained the model is. While we cannot confirm conclusively that this is the case, the notably poor performance of this dataset on this and past models may indicate that the features given are in some way irrelevant to the value being predicted. This is an essential thing to recognize in the deployment and analysis of machine learning models; if our features are not sufficient to predict our desired value, nothing can be done to enhance our predictions through training or more sophisticated models. Even if some or most features are relevant to the predicted value, those that are not can introduce noise into the dataset, and therefore interfere with our predictions.

Looking specifically at figure 4, we can see that the results are roughly consistent with our hypothesis, with the important caveat that the model does not perform best with two hidden layers, instead performing best with one hidden layer. This indicates a potential oversight in previous thinking and a valuable lesson for future research into or implementation of these models. While it may be tempting to think that more parameters or a more complicated model will inherently be more effective in addressing a given classification or regression problem, it is important to recognize the potential problems that come with more complicated models. While they have been addressed somewhat previously, it is worth noting the problem of overfitting that can come from having too many parameters in your model. Outside of model complexity, we can see in figure 7b that we fall into a local minima when training the 2-hidden-layer model on the Abalone dataset. Thus, further research should investigate ways to address this issue through different tuning possibilities for the learning rate or other methods. While these reasons provide some explanation for our poorer performance with more hidden layers, they do not fully explain the issue. Thus, further research into the relationship between dataset size and optimal number of hidden layers is likely needed.

Looking at figure 5, we can see that the results are roughly consistent with our hypothesis. While the neural network performed better than expected with no hidden layers, it performed relatively poorly as the complexity of the model grew. While this is likely partly due to the small size of the dataset as mentioned previously, it is likely a combination of this with the relatively high skew in this dataset. Thus, the problem of overfitting becomes more pressing as more significant noise is present in the dataset in the form of skew, as the model is fitting to this noise that brings on significant errors when incorrect predictions are made (as we likely predict further from the correct value given the magnitude of the skew). The impact of this noise can also be seen in figure 7c. Where other datasets saw relatively smooth graphs, as seen in figures 7a and 7b, we can see that the noise in the machine dataset has a significant impact on the convergence of the 2-hidden-layer model, making it somewhat unstable. This confluence of factors provides a reasonable explanation of the relative performances of our models on this dataset, though - again - further research into the relationship between model size and dataset size is needed before drawing definitive conclusions.

Looking at figure 3, we can see that our results for the glass dataset were relatively consistent with our hypothesis. While, like other datasets, we did not anticipate such a significant drop in performance for the 2-hidden-layer model, we can see by comparing this figure against other datasets that this dataset was especially resilient to any diminished performance from additional hidden layers. While this is not exactly what we hypothesized, it may demonstrate that the dataset's relative "complexity" makes it better suited to more hidden layers, and that perhaps more complicated datasets would be even better suited to more hidden layers. Further research could take a wider variety of datasets, emphasizing a wide and dense distribution of complexities in the datasets, so that clearer connections between this complexity and the performance with more hidden layers can be seen.

Looking at figures 2 and 3, we see relatively similar results for both the breast cancer and soybean datasets. Both have been seen in previous experiments to be relatively well-formed datasets, cooperative with past models. However, we can see that both exhibit a significant drop in performance with two hidden layers. The potential reasons for this

have been outlined previously and do not need to be restated. Each of these are datasets that exhibit very high performance on simpler models and do not seem to benefit from the additional complexity of a two-hidden-layer network. Further, we can see in figure 8 that the model converges relatively quickly with one hidden layer on the breast cancer dataset. Again, the lack of noise, intermediate size, and roughly even distribution of this dataset make it very friendly to the models we have tested thus far. The soybean set sees similar benefits, but its relatively small size makes it difficult to apply more complicated models.

6 Conclusion

Our experiment was roughly consistent with past research and theory, demonstrating Neural Networks as a valuable tool to be used in a variety of problems, especially when consideration is given to the necessary size of the model given the complexity or size of the dataset. We found that Neural Networks performed well on most datasets, but performance varied significantly in performance with the number of hidden layers used. While this has been seen in the past, our results exhibited more significant drops in performance with more hidden layers than previous research has indicated. Further research into the relationship between the traits that a dataset exhibits and the performance of neural networks of different sizes should be carried out to see what exactly causes these changes in performance. Further, more advanced methods related to the training process should be investigated, including minibatch or batch learning as opposed to stochastic learning, a momentum term, or bias nodes. These methods may be able to compensate for the negative impacts of more hidden layers or change the impacts of those layers to more effectively contribute to performance for our neural networks.

7 Appendix

Group work distribution

Paper — The paper was written by a combination of everyone. Nick wrote the introduction, abstract, and problem statement. Josh wrote the results and the program design. Owen wrote the rest.

Code — The code was written by a combination of everyone.

Design Document — The design document was completed as a group with all members involved to ensure everyone understood the timeline and distribution of labor for the project.

Video Essay — The video essay was recorded and edited by all of the team members.