Exploring the Impacts of Hidden Layers on Feed-Forward Neural Networks

Braeden Hunt BRAEDEN.HUNT@STUDENT.MONTANA.EDU
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Tyler Koon TYLER.KOON@STUDENT.MONTANA.EDU
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Editor: Braeden Hunt and Tyler Koon

Abstract

This paper details a set of six experiments which explore the effects of hidden layers on the performance of feed-forward neural networks. In particular, each experiment considered the relative difference in model performance when trained using a network of zero, one, and two hidden layers. Model performance was quantified by the 0-1 Loss, F_1 Score, and Mean Squared Error metrics, which were computed using a stratified ten-fold cross-validation technique. Each experiment was based on one of six independent and widely varied datasets, three of which represent the classification setting and three of which represent the regression setting. The results from each of these experiments were compared to draw general conclusions regarding the effect of the number of hidden layers on underlying prediction performance across many contexts.

Keywords: Feed-Forward Neural Network, Hidden Layers, Cross Validation, Regression, Classification

1. Introduction

Recent developments in computation technology has invited a renewed interest in deep learning techniques and their applications. Despite the overwhelming adoption of this learning method, there exists well-founded concern regarding the scope of interpretability of results from neural networks. This concern generally stems from limited knowledge on what occurs in the hidden layers: the so-called black box problem. Understandably, the machine learning community is researching the behavior of neural networks so as to improve this understanding and hopefully develop more transparent models that mitigate the aforementioned concerns. Our study attempts to broadly contribute to this research goal by exploring the impact of hidden layers on the training process, and how those layers affect model performance.

In particular, this study considers the impact of hidden layers on the performance of feed-forward neural networks in both the classification and regression setting. To address this research question we quantified model performance as an aggregate of 0-1 Loss, F_1

Score, and Mean Squared Error metrics which were compared for simple neural networks consisting of zero, one, and two hidden layers. Our experimental design implemented a tuning process that found semi-optimal values for the network shape, learning rate, training momentum, batch size, and number of epochs which were then used to compute the aforementioned performance metrics using a stratified ten-fold cross validation technique. Initially, we hypothesized that a network with one hidden layer would perform better than a network with zero hidden layers due to the additional representation power. Additionally, we predicted that networks with two hidden layers would offer little performance improvement over the network with one hidden layer due to the vanishing gradient problem. That said, we believed that networks with more hidden layers would tend to converge in fewer epochs as a result of better representing the underlying function. Our study offered interesting results which tended to support these predictions while simultaneously suggesting that additional research may be required for certain contexts.

The following sections outline our experimental process, implementation, and results from the study. We begin with an in-depth exploration of our experimental design in Section 2. This is followed by a presentation of experimental results in section 3, which is accompanied by discussion on the implications and limitations of the study in section 4. Finally, we conclude in Section 5 with a summary of the study as well as considerations for future research on the subject.

2. Experimental Design

This section explores our experimental approach to the addressing the proposed research question. It begins with an exploration of our data sets and their composition (Section 2.1), followed by a discussion of our program and its design (Section 2.2), a review of our evaluation methodology (Section 2.3), an overview of our implemented feed-forward neural network algorithm (Section 2.4), and lastly a description of our experimental design (Section 2.5).

2.1 Data Sets

Our study used six independent data sets sourced from the UCI Machine Learning repository to explore our hypothesis. Of these data sets, three contained classification data and three contained regression data. The three classification data sets were the Wisconsin Breast Cancer Database (January 8, 1991), Glass Identification Database, and Small Soybean Database. The three chosen regression data sets for this study were Abalone, Computer Hardware, and Forest Fires. Each of these data sets offered a unique combination of features, sample sizes, and response values, improving the breadth of our study by affording a more diverse set of experiments. The diversity of these six data sets offered an appropriate platform to evaluate our hypothesis with respect to many different contexts in both a classification and regression setting. Accompanying this variability in the composition of these data was a number of issues that needed to be addressed through data wrangling and preprocessing. This included the need to handle categorical features of varying complexity, account for missing data, and drop irrelevant features. The details of these challenges and our solutions are discussed at length in Section 2.5.

2.2 Program Design

To perform this study, we developed a robust set of software tools that facilitated our experiments. These tools were divided into three categories: Algorithms, Evaluation, and Utilities. The Algorithms category contains our implementation of a feed-forward neural network and the back-propagation algorithm. This is accompanied by a generic *Layer* class that encapsulates information regarding the number of nodes, activation function, weights, and biases for a given layer. These layer objects are passed into the *NeuralNetwork* class, allowing us to create networks of varying shapes and properties. This approach offered a scalable solution to representing neural networks while simultaneously allowing for quick modification and tuning during our experiments.

The Evaluation component of our program contains tools for computing the performance of a trained network. This includes a *CrossValidation* class that exposes tools for performing stratified k-fold cross validation on a given network. This class contains methods for folding a dataset using stratification, holding out data for tuning, and computing the average performance across folds. Additionally, this component includes an *EvaluationMeasure* class which provides methods for computing the metrics defined in Section 2.3. Abstracting this logic to class methods offered a scalable approach to defining and implementing our evaluation methods.

Lastly, the Utilities component contains classes that broadly support our experiments. This includes the *Preprocess*, *TuningUtility*, and *Utilties* classes. The *Preprocess* class facilitates reading in CSV data, dropping irrelevant features and missing data, and performing necessary modifications to the underlying data. The *TuningUtility* exposes a useful tool for tuning the shape of our networks to find semi-optimal layer and node composition before training. Lastly, the *Utilties* class offers a handful of miscellaneous tools used throughout our program, such as normalizing features to be in the range$[0, 1]$ and one-hot encoding that augment the training data within each cross-validation fold.

2.3 Evaluation Methodology

To compare the network subjects in our experiments, we chose a set of robust metrics to quantify network performance. For classification data, we considered a composite of the 0-1 Loss and F_β score. We chose $\beta = 1$ as it equally considers the precision and recall of the model, which was desirable given our goal of comparing *general* model performance. Additionally, these two metrics are measured on a standardized scale which simplified the comparison process. For regression data, we considered the Mean Squared Error of the predicted values. This metric was chosen for its ability to weight the resulting error, offering a polynomial increase in error for predictions that deviate further from the ground truth.

Collectively, these metrics offer a holistic overview of network performance while remaining computationally simple and easily comparable. That being said, future research in the area might consider additional performance measures as a means of exploring the more granular impacts of hidden layers. One such alteration might consider different values for β so as to focus on impacts to precision *or* recall as opposed to their aggregation.

2.4 Algorithms

Our study explores feed-forward neural networks which implement the backpropagation algorithm. This algorithm works by propagating the network output error through each hidden layer and updating the weights that influence the layer's action potential along the way. This update process is driven by stochastic gradient descent, which converges to the locally optimal weight configuration. To control this learning process and prevent oscillation around local minima, the algorithm exposes tunable learning rate and momentum hyperparameters that augment the rate and fortitude of descent. Additionally, the behavior of the algorithm can be influenced by activation functions that modify the action potential for each hidden layer. Our implementation used the non-linear logistic activation (sigmoidal) function for hidden layers, offering a continuous gradient bounded by 0 and 1. In the classification setting our implementation used the soft-max activation in the output layer so as to return a probability distribution of prediction, which is desirable for multi-class problems. For the regression setting, we simply used the logistic activation for the output layer.

2.5 Experiments

Our study consisted of six independent experiments, each of which considered the performance of a neural network with zero, one, and two hidden layers in a diverse context. These contexts were defined by each of the underlying datasets described in Section 2.1, offering exposure to both classification and regression settings. The experiment structure was composed of three stages: Preprocessing, Tuning, and Evaluation

The preprocessing stage was responsible for reading and wrangling the training data. For our experiments, we decided to simply drop samples which were missing data. Though more advanced solutions such as data imputation were considered, there were so few affected samples that the benefits offered by more advanced methods were deemed negligible. Additionally, this stage removed irrelevant features and handled any necessary conversions of the underlying data.

The tuning stage was responsible for finding semi-optimal hyperparameters for each neural network in the experiment. Given the time-constrained nature of this research project, the learning rate, momentum, batch size, epoch parameters were tuned using an experimental approach that considered performance from two fold cross validation across a range of parameter values. To tune the shape of each network, a stratified tuning holdout set was created from 10% of the training data. A set of network configurations was then generated consisting of all permutations of layers where no layer had fewer nodes than the previous. Each configuration was then trained on the remaining 90% of the training data. For each fold, the network with the best performance on the tuning set was selected based on the 0-1 Loss (for classification data) and Mean Squared Error (for regression data) computed for the resulting models. The tuning holdout was held constant for all configurations, offering standardized results which were easily comparable and made selecting the best model for each fold independent of the final testing data set. The configurations that presented the best performance on the tuning data set were selected for the evaluation stage.

In order to reduce the network shape space to a reasonable size, we applied some restrictions on the network shapes. In particular, we only considered shapes whose layers had fewer nodes than their preceding layer. This constraint helped mitigate bottlenecks in the networks related to the deeper hidden layers being unnecessarily large. Tough this may be desirable in certain situations, we considered it negligible due to the limited complexity of our underlying training data. An additional constraint imposed on the two-hidden layer regression problems involved limiting the number of nodes in each hidden layer to be equal. This was done to minimize the network shape space, which tends to be substantially larger in the regression setting.

Lastly, the evaluation stage computed the performance of each selected model for the folds against the holdout fold in 10-fold cross validation fashion. This was done for all the three network depths and their tuned hyper parameters. Using the computed 0-1 Loss, F_1 Score, and Mean Squared Error performance metrics, we could compare the relative performances of networks with zero, one and two hidden layers in the given context.

3. Results

This section provides additional context for the underlying experiments, and reports the results obtained in the study.

3.1 Breast Cancer Identification Results

The Wisconsin Breast Cancer Database (January 8, 1991) describes samples of breast tumors. Each sample has nine continuous features (encoded from 1-10) and a tenth classification feature representing whether or not the tumor is malignant or benign. For the F_β score, we considered malignant to be the "positive" class and benign to be the "negative" class. In total, 699 samples are described by the data set: 241 malignant, and 458 benign. Of these samples, 16 are missing a single attribute and zero are missing two or more attributes. Given the relatively small proportion, we decided to drop all instances missing any feature data. The results from the experiment are as follows:

0 Hidden Layers				
Fold	Shape	Tuning Loss	Testing Loss	Testing F_1 Score
1	9, 2	0.942	0.903	0.864
2	9, 2	0.942	0.887	0.851
3	9, 2	0.942	0.903	0.880
4	9, 2	0.942	0.903	0.875
5	9, 2	0.942	0.871	0.840
6	9, 2	0.942	0.934	0.909
7	9, 2	0.942	0.918	0.878
8	9, 2	0.942	0.951	0.927
9	9, 2	0.942	0.885	0.857
10	9, 2	0.942	0.950	0.927
Avg.	—	0.942	0.911	0.881

Table 1: Results from network with 0 hidden layers trained using a learning rate of 0.1, momentum of 0, and batch size of 10 for 1000 epochs

1 Hidden Layer				
Fold	Shape	Tuning Loss	Testing Loss	Testing F_1 Score
1	9, 9, 2	0.986	0.984	0.977
2	9, 9, 2	0.986	0.984	0.978
3	9, 9, 2	1.000	0.984	0.978
4	9, 9, 2	0.986	0.968	0.957
5	9, 9, 2	1.000	0.952	0.933
6	9, 9, 2	0.986	1.000	1.000
7	9, 9, 2	1.000	0.984	0.977
8	9, 9, 2	0.986	0.984	0.977
9	9, 9, 2	1.000	1.000	1.000
10	9, 9, 2	1.000	1.000	1.000
Avg.	—	0.993	0.984	0.978

Table 2: Results from network with 1 hidden layers trained using a learning rate of 0.1, momentum of 0, and batch size of 10 for 1000 epochs

2 Hidden Layers				
Fold	Shape	Tuning Loss	Testing Loss	Testing F_1 Score
1	9, 9, 9, 2	0.986	0.984	0.977
2	9, 9, 9, 2	0.986	0.984	0.978
3	9, 7, 7, 2	1.000	0.984	0.978
4	9, 9, 9, 2	0.986	0.968	0.957
5	9, 4, 4, 2	1.000	0.968	0.957
6	9, 8, 8, 2	1.000	1.000	1.000
7	9, 9, 9, 2	0.986	0.984	0.977
8	9, 9, 9, 2	0.986	0.984	0.977
9	9, 8, 8, 2	1.000	1.000	1.000
10	9, 9, 9, 2	0.986	1.000	1.000
Avg.	—	0.991	0.985	0.980

Table 3: Results from network with 2 hidden layers trained using a learning rate of 0.1, momentum of 0, and batch size of 10 for 1000 epochs

3.2 Soybean Identification Results

The Small Soybean Database describes samples of soybeans through 35 nominalized categorical features. Each sample is classified into one of four output classes: D1 (10 instances), D2 (10 instances), D3 (10 instances), and D4 (17 instances). For our F_β computations, we considered the D1 class as "positive", and all other classification levels as "negative". In total the dataset describes 47 samples, none of which are missing feature data. The results from the experiment are as follows:

0 Hidden Layers				
Fold	Shape	Tuning Loss	Testing Loss	Testing F_1 Score
1	35, 4	1.000	1.000	1.000
2	35, 4	1.000	1.000	1.000
3	35, 4	1.000	1.000	1.000
4	35, 4	1.000	1.000	1.000
5	35, 4	1.000	1.000	1.000
6	35, 4	1.000	1.000	1.000
7	35, 4	1.000	1.000	1.000
8	35, 4	1.000	1.000	1.000
9	35, 4	1.000	1.000	1.000
10	35, 4	1.000	1.000	1.000
Avg.	—	1.000	1.000	1.000

Table 4: Results from network with 0 hidden layers trained using a learning rate of 0.01, momentum of 0.1, and batch size of 1 for 200 epochs

1 Hidden Layer				
Fold	Shape	Tuning Loss	Testing Loss	Testing F_1 Score
1	35, 35, 4	1.000	1.000	1.000
2	35, 35, 4	1.000	0.800	1.000
3	35, 34, 4	1.000	1.000	1.000
4	35, 35, 4	1.000	1.000	1.000
5	35, 35, 4	1.000	1.000	1.000
6	35, 34, 4	1.000	1.000	1.000
7	35, 33, 4	1.000	1.000	1.000
8	35, 34, 4	1.000	1.000	1.000
9	35, 32, 4	1.000	1.000	1.000
10	35, 35, 4	1.000	1.000	1.000
Avg.	—	1.000	0.980	1.000

Table 5: Results from network with 1 hidden layers trained using a learning rate of 0.01, momentum of 0, and batch size of 10 for 500 epochs

2 Hidden Layers				
Fold	Shape	Tuning Loss	Testing Loss	Testing F_1 Score
1	35, 35, 35, 4	1.000	1.000	1.000
2	35, 35, 35, 4	1.000	1.000	1.000
3	35, 35, 35, 4	1.000	1.000	1.000
4	35, 35, 35, 4	1.000	1.000	1.000
5	35, 35, 35, 4	1.000	1.000	1.000
6	35, 35, 35, 4	1.000	1.000	1.000
7	35, 35, 35, 4	1.000	1.000	1.000
8	35, 35, 35, 4	1.000	1.000	1.000
9	35, 35, 35, 4	1.000	1.000	1.000
10	35, 35, 35, 4	1.000	1.000	1.000
Avg.	—	1.000	1.000	1.000

Table 6: Results from network with 2 hidden layers trained using a learning rate of 0.1, momentum of 0, and batch size of 10 for 500 epochs

3.3 Glass Identification Results

The Glass Identification Database describes samples of glass fragments through nine continuous features (note, the 'ID' feature was dropped during our preprocessing stage due to irrelevancy) describing the physical properties of the fragment. Each sample is classified into one of seven classes (encoded as integers 1-7) representing the type of glass. For out F_β computation we considered the class 2 to be "positive", and the remaining classes to be "negative". In total, the dataset contains 214 samples, none of which are missing feature data. The results from the experiment are as follows:

0 Hidden Layers				
Fold	Shape	Tuning Loss	Testing Loss	Testing F_1 Score
1	9, 7	0.696	0.682	0.706
2	9, 7	0.609	0.667	0.571
3	9, 7	0.652	0.524	0.588
4	9, 7	0.652	0.600	0.500
5	9, 7	0.696	0.650	0.714
6	9, 7	0.696	0.474	0.333
7	9, 7	0.783	0.556	0.533
8	9, 7	0.739	0.611	0.625
9	9, 7	0.696	0.750	0.727
10	9, 7	0.565	0.812	0.800
Avg.	—	0.678	0.633	0.610

Table 7: Results from network with 0 hidden layers trained using a learning rate of 0.02, momentum of 0.1, and batch size of 10 for 2000 epochs

1 Hidden Layer				
Fold	Shape	Tuning Loss	Testing Loss	Testing F_1 Score
1	9, 9, 7	0.783	0.773	0.667
2	9, 9, 7	0.652	0.762	0.714
3	9, 9, 7	0.696	0.714	0.706
4	9, 9, 7	0.652	0.700	0.667
5	9, 9, 7	0.826	0.800	0.875
6	9, 7, 7	0.739	0.789	0.769
7	9, 9, 7	0.826	0.722	0.800
8	9, 7, 7	0.826	0.833	0.769
9	9, 9, 7	0.826	0.938	0.909
10	9, 7, 7	0.783	0.938	1.000
Avg.	—	0.761	0.797	0.788

Table 8: Results from network with 1 hidden layers trained using a learning rate of 0.1, momentum of 0, and batch size of 10 for 2000 epochs

2 Hidden Layers				
Fold	Shape	Tuning Loss	Testing Loss	Testing F_1 Score
1	9, 8, 8, 7	0.696	0.591	0.571
2	9, 9, 7, 7	0.826	0.810	0.857
3	9, 9, 9, 7	0.870	0.667	0.769
4	9, 9, 8, 7	0.609	0.800	0.833
5	9, 9, 8, 7	0.652	0.800	0.824
6	9, 8, 8, 7	0.783	0.842	0.933
7	9, 9, 8, 7	0.826	0.722	0.750
8	9, 9, 9, 7	0.783	0.778	0.714
9	9, 9, 7, 7	0.783	0.875	0.857
10	9, 9, 7, 7	0.739	0.938	0.923
Avg.	—	0.757	0.782	0.803

Table 9: Results from network with 2 hidden layers trained using a learning rate of 0.1, momentum of 0.1, and batch size of 10 for 2000 epochs

3.4 Forest Fires

The Forest Fires data set describes 517 instances, none of which contain missing feature values. There are a total of 13 attributes, two of which are cyclical. The rest are continuous values that represent the conditions of the recorded fire. We attempt to calculate the logarithmic transformed burned area of the forest from these values: $\hat{y} = log(y + 1)$. This transformation helps addresses the skew towards an area of zero. We also restricted the tuning of the two hidden layer experiment to just consider shapes that had the same number of nodes in both hidden layers. The results from the experiment are as follows:

0 Hidden Layers			
Fold	Shape	Tuning MSE	Testing MSE
1	12, 1	1.433	1.490
2	12, 1	1.499	1.985
3	12, 1	1.453	1.769
4	12, 1	1.490	1.866
5	12, 1	1.446	1.992
6	12, 1	1.463	2.104
7	12, 1	1.454	2.039
8	12, 1	1.470	1.978
9	12, 1	1.442	2.107
10	12, 1	1.472	2.244
Avg.	—	1.462	1.957

Table 10: Results from network with 0 hidden layers trained using a learning rate of 0.01, momentum of 0, and batch size of 10 for 1000 epochs

1 Hidden Layer			
Fold	Shape	Tuning MSE	Testing MSE
1	12, 6, 1	1.207	1.110
2	12, 11, 1	1.259	1.752
3	12, 2, 1	1.425	1.724
4	12, 2, 1	1.276	1.665
5	12, 2, 1	1.252	1.795
6	12, 6, 1	1.173	1.990
7	12, 6, 1	1.365	2.035
8	12, 3, 1	1.280	2.069
9	12, 2, 1	1.213	1.950
10	12, 3, 1	1.219	2.141
Avg.	—	1.267	1.823

Table 11: Results from network with 1 hidden layers trained using a learning rate of 0.01, momentum of 0, and batch size of 10 for 1000 epochs

2 Hidden Layers			
Fold	Shape	Tuning MSE	Testing MSE
1	12, 3, 3, 1	1.394	1.467
2	12, 8, 8, 1	1.476	1.802
3	12, 4, 4, 1	1.430	1.690
4	12, 8, 8, 1	1.449	1.758
5	12, 2, 2, 1	1.267	1.689
6	12, 4, 4, 1	1.376	1.989
7	12, 8, 8, 1	1.443	2.016
8	12, 8, 8, 1	1.430	1.931
9	12, 5, 5, 1	1.447	1.982
10	12, 3, 3, 1	1.369	2.158
Avg.	—	1.408	1.848

Table 12: Results from network with 2 hidden layers trained using a learning rate of 0.01, momentum of 0, and batch size of 10 for 1000 epochs

3.5 Computer Hardware

The Computer Hardware data set describes 209 instances, none of which are missing feature values. There are a total of ten attributes, two of which were dropped due to being identifiers. The rest are continuous values that describe computer hardware. We attempt to calculate the Estimated Relative Performance (ERP) from these values. We restricted the tuning of the two hidden layer experiment to just consider shapes that had the same number of nodes in both hidden layers. The results from the experiment are as follows:

0 Hidden Layers			
Fold	Shape	Tuning MSE	Testing MSE
1	7, 1	213.102	942.533
2	7, 1	295.980	1784.776
3	7, 1	265.082	154.658
4	7, 1	279.906	588.317
5	7, 1	260.231	197.171
6	7, 1	278.879	326.871
7	7, 1	265.239	679.116
8	7, 1	263.739	1043.151
9	7, 1	246.227	1199.112
10	7, 1	254.256	1466.526
Avg.	—	262.264	838.223

Table 13: Results from network with 0 hidden layers trained using a learning rate of 0.01, momentum of 0, and batch size of 10 for 1000 epochs

1 Hidden Layer			
Fold	Shape	Tuning MSE	Testing MSE
1	7, 2, 1	1659.809	5075.479
2	7, 2, 1	1572.856	4449.832
3	7, 2, 1	1463.582	1198.781
4	7, 3, 1	1816.455	2332.283
5	7, 2, 1	1502.046	1837.208
6	7, 2, 1	1556.447	1643.638
7	7, 3, 1	1407.676	868.378
8	7, 4, 1	1681.995	3594.455
9	7, 2, 1	1411.338	16611.754
10	7, 2, 1	1473.279	24070.285
Avg.	—	1554.548	6168.209

Table 14: Results from network with 1 hidden layers trained using a learning rate of 0.005, momentum of 0, and batch size of 10 for 1000 epochs

2 Hidden Layers			
Fold	Shape	Tuning MSE	Testing MSE
1	7, 6, 6, 1	31.493	267.128
2	7, 7, 7, 1	104.837	426.054
3	7, 6, 6, 1	63.198	25.103
4	7, 6, 6, 1	17.641	15.216
5	7, 7, 7, 1	17.815	27.175
6	7, 5, 5, 1	28.804	84.793
7	7, 6, 6, 1	9.597	11.832
8	7, 6, 6, 1	14.846	1130.375
9	7, 7, 7, 1	7.296	1655.088
10	7, 7, 7, 1	32.946	2948.656
Avg.	—	32.847	659.142

Table 15: Results from network with 2 hidden layers trained using a learning rate of 0.0001, momentum of 0, and batch size of 10 for 1000 epochs

3.6 Abalone

The Abalone data set describes 4177 instances, none of which are missing feature values. There are a total of nine attributes. One is a nominal value ('sex') that we implemented one-hot-encoding for. The rest are continuous values about the size and weight of the abalones. We attempt to calculate the abalone rings (which corresponds to the sample's age) from these values. We restricted the tuning of the two hidden layer experiment to just consider shapes that had the same number of nodes in both hidden layers. The results from the experiment are as follows:

0 Hidden Layers			
Fold	Shape	Tuning MSE	Testing MSE
1	8, 1	5.010	13.939
2	8, 1	5.272	4.669
3	8, 1	5.227	5.765
4	8, 1	5.235	5.686
5	8, 1	5.375	4.774
6	8, 1	5.244	5.957
7	8, 1	5.218	23.874
8	8, 1	5.225	5.590
9	8, 1	5.217	5.380
10	8, 1	5.210	5.362
Avg.	—	5.223	8.100

Table 16: Results from network with 0 hidden layers trained using a learning rate of 0.001, momentum of 0.01, and batch size of 5 for 1000 epochs

1 Hidden Layer			
Fold	Shape	Tuning MSE	Testing MSE
1	8, 8, 1	4.127	3.747
2	8, 8, 1	4.090	3.977
3	8, 8, 1	4.100	4.104
4	8, 8, 1	4.102	4.162
5	8, 8, 1	4.148	3.255
6	8, 8, 1	4.064	4.625
7	8, 8, 1	4.061	5.359
8	8, 7, 1	4.083	4.267
9	8, 8, 1	4.099	4.996
10	8, 8, 1	4.067	4.367
Avg.	—	4.094	4.286

Table 17: Results from network with 1 hidden layers trained using a learning rate of 0.001, momentum of 0.01, and batch size of 5 for 1000 epochs

2 Hidden Layers			
Fold	Shape	Tuning MSE	Testing MSE
1	8, 8, 8, 1	4.051	3.592
2	8, 8, 8, 1	4.083	4.178
3	8, 8, 8, 1	4.129	4.075
4	8, 8, 8, 1	4.036	4.809
5	8, 4, 4, 1	4.116	3.266
6	8, 8, 8, 1	4.092	4.781
7	8, 8, 8, 1	4.029	5.587
8	8, 4, 4, 1	4.079	4.332
9	8, 7, 7, 1	4.115	3.914
10	8, 6, 6, 1	4.062	4.342
Avg.	—	4.079	4.288

Table 18: Results from network with 2 hidden layers trained using a learning rate of 0.001, momentum of 0.01, and batch size of 5 for 1000 epochs

4. Exploration of Results

This section discusses our experimental results and their implications. We also consider the limitations of our study, and propose ideas for additional research on the subject.

4.1 Discussion

Our initial hypotheses predicted that networks with one or more hidden layers would perform better than similar networks with zero hidden layers due to the added representational power that accompanies additional layers. Additionally, we believed that the performance gain offered by a second hidden layer would be minimal due to the vanishing gradient problem. Generally, the results from our experiments agreed with these initial predictions. For the three classification data sets, the average increase in accuracy and F_1 Score between the networks with zero and one hidden layer was 7.233% and 9.167% respectively. For the networks with one hidden layer and two hidden layers, this increase in accuracy and F_1 Score was only 0.2% and 0.567% respectively. These results clearly indicate that performance tends to increase when adding one hidden layer, and then plateaus when adding additional hidden layers. It should be noted that the Soybean Identification experiment deviated slightly from this trend, offering a 2% lower testing accuracy for the network with one hidden layer. However given the small magnitude of the difference we consider this deviation negligible.

The results from our regression dataset offered less evidence for hypothesis, indicating an average increase in Mean Squared Error between networks with zero and one hidden layer of 1775.353 and an average decrease of 1836.347 between networks with one and two hidden layers. However these values are highly skewed by the results from our Computer Hardware experiment which provided exceptionally large MSE for the network with one hidden layer. We attribute this large MSE to the high variability in the underlying Machine dataset which includes a number of high-valued outliers. Additional research might consider removing these outliers before training and evaluating the networks and their performance. Ignoring the results from the Computer Hardware experiment, the average change in MSE between networks with zero and one hidden layer became -1.316, and the average change in MSE between networks with one and two hidden layers became 0.009. These values align much better with our initial prediction, one again suggesting that performance tends to increase when adding one hidden layer and remain relatively constant when adding additional hidden layers.

4.2 Limitations

Our study has certain limitations that prevent the results from being fully generalizable. For example, we heavily restricted the shapes that we tested for the networks. While we aimed to restrict them in meaningful ways, further research should consider less restricted network shapes. Additionally, we only worked with three regression and three classification data sets. These are an incredibly small subset of problems that researchers aim to solve using machine learning, and our results cannot necessarily be generalized to all regression and classification contexts.

Additionally, we performed manual tuning of certain hyperparameters, including learning rate, momentum, batch size, and epochs. We optimized these values to the best of our abilities, but it is possible that we did find semi-optimal values which adds uncertainty to our results. More research for this problem should consider more robust and broad tuning methods for these parameters.

Finally, the regression data sets that we used were very difficult to learn, as there were large outliers in a couple of them. These outliers are difficult to learn and heavily skew sensitive measures like Mean Squared Error. However, we believe that these don't significantly detract from our results as we are comparing the relative performance between each network depth. Since they all were subjected to this issue, it shouldn't change their relative performances.

5. Summary

This study attempts to contribute to the research goal of improving the general understanding of neural networks and their behavior. In particular, we considered how the number of hidden layers in a feed forward neural network impacts the performance of the network in various contexts. To explore this question, we developed a robust set of software tools that allowed us to implement six experiments based on six independent and widely varied data sets covering both the classification and regression setting. Each experiment considered three networks having zero, one, and two hidden layers respectively, the shapes of which were tuned using a grid search-like method. Stratified ten-fold cross validation was then used to compute the mean performance of each network, which we defined through three robust metrics: 0-1 Loss and F_1 score for classification data, and Mean Squared Error for regression data. We initially predicted that adding one hidden layer would provide significant increases in performance while adding additional hidden layers would offer minimal improvements to performance due to the vanishing gradient problem. The results from our experiments provided strong evidence supporting this hypotheses, indicating that 0-1 Loss, F_1 Scores, and Mean Squared Error tend to improve between networks with zero and one hidden layer, and offer minimal change between networks with one and two hidden layers. Though this study provides useful insight into the behavior of neural networks with up to two hidden layers, it introduces many limitations which warrant further research of the underlying question.

References

Anthony J. Papagelis, Dong Soo Kim. Multi-Layer Perceptron. University of New South Wales. Available electronically at https://www.cse.unsw.edu.au/ cs9417ml/MLP2/

Matt Mazur. *A Step by Step Backpropagation Example*, Available electronically at https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/