# Neural Networks

**Te-Wei (David) Wu** TWU55@JH.EDU

## 1. Introduction

A neural network is a machine learning model that seek to emulate human neurons. Generally, they are comprised of layers nodes connected to each other via weights. An input example is put into the network and propagated through to the end, whereupon we see an output. In this paper, we will be looking at the comparative performance of several neural networks

We identify three types of neural networks in this paper for evaluation and comparison, a simple linear network, a feed forward network (FNN) with two hidden layers, and a FFN with two hidden layers where the first hidden layer is trained from an autoencoder. These algorithms will be evaluated on six data sets from the University of California Irvine Machine Learning Repository (UCI MLR), consisting of three classification tasks and three regression tasks. We will comparatively evaluate these trees against the performance of a simple majority/average algorithm.

Overall, we expect all networks to perform at least on par with a simple majority algorithm for classification or a simple average for regression. We expect that performance of both the FNN with two hidden layers to perform better than the simple linear network. We expect that there will be no significant difference between FFN with two hidden layers and the autoencoder enhanced FFN.

## 2. Experimental Approach

Our experimental approach is as follows. 20% of each data set will be held out as a validation set for tuning parameters such as learning rate and hidden layer size. The rest of the data set will be used for obtaining experimental results through 5-fold CV. In other words, during 5-fold CV, 16% of the original data set is to be used as a test fold, 64% is to be used as training, and 20% is to be used as validation. The 16% which acts as the testing set will be swapped with the other folds in the 64%, until 5 trials have been completed. The 20% held out for tuning remains the same during 5-fold CV.

For regression tasks, the output of each node is a single real value. For classification tasks, the probability of each class is estimated using softmax, and the highest probability is chosen as the class.

The results of regression tasks are measured through Mean Squared Error (MSE) and the results of classification tasks are measured through accuracy. Performance results are compared statistically through a T-test for two independent means.

In terms of assumptions that we have made, we assume a parametric form of the output variable. For example, in the simple linear network we assume that the output variable follows the form:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

Where $f(\mathbf{x})$ represents the value to be predicted, and $\mathbf{x}$ represents the feature values of that observation.

As for classification, we assume that the sample point to be estimated can be modelled by one draw of a multinomial trial where $\mathbf{K}$ is equal to the number of classes within the data set. That is:

$$\mathbf{r}^t|\mathbf{x}^t \sim Mult_K(1, \mathbf{y}^t)$$

where $y_i^t \equiv P(C_i|\mathbf{x}^t)$. $\mathbf{r}^t$ represents the real class of the sample point, and $\mathbf{x}^t$ represents the feature values of that sample point. In other words, we are modelling the probabilities that the sample point belongs to the classes.

Similar assumptions are made with the FNNs at the output layers. The difference being, within each hidden layer, outputs of those layers are pushed through an activation function (sigmoid) and given as an output.

For the FNNs with two hidden layers, hidden layer sizes are tuned using the validation set until the best achievable performance metrics are found. These tuned parameters are then used in 5-fold CV across all folds.

For the hidden layers with the first layer replaced by an autoencoder, the the parameters for the autoencoder are tuned using a single testing fold and the validation set. The autoencoder is trained until minimum reconstruction error is reached. The weights of this autoencoder are then used as the input weights for a regular FNN with two hidden layers.

The neural networks are trained using gradient descent. The loss functions that we are minimizing are MSE for regression and cross entropy for classification. The simple linear network utilizes batch updates. The deep neural networks with two hidden layers utilize stochastic gradient descent.

## 3. Results

The results of the experiments are as follows. Across five folds, the performance of the previously specified metrics (classification accuracy for classification tasks, MSE for regression tasks) are displayed with the associated algorithm that produced them. The performance metric for all of the folds were all averaged to obtain a 5-fold CV average.

Results for the simple average and simple majority algorithm are included for comparison.

Names are abbreviated as follows: Simple Linear Network (SLN), FeedForward Network with two hidden layers (FFN-2HL), Feed Forward Network with first hidden layer replaced by auto encoder (FFN-AE).

### 3.1 House Votes Data Set Results

SLN : Network trained at learning rate 0.3 over 500 epochs.

FFN-2HL: Network trained with 15 hidden nodes in the first layer, 15 hidden nodes in the second layer, learning rate 0.05 over 50 epochs.

FFN-AE: Autoencoder: trained with 13 hidden nodes, learning rate 0.2, over 25 epochs. Network trained with 15 hidden nodes in the second hidden layer, learning rate 0.05 over 50 epochs.

| Fold | SLN | FFN-2HL | FFN-AE | Simple |
|---|---|---|---|---|
| Fold 1 | 0.943 | 0.957 | 0.957 | 0.6136 |
| Fold 2 | 0.971 | 0.971 | 0.971 | 0.6136 |
| Fold 3 | 0.957 | 0.943 | 0.957 | 0.6092 |
| Fold 4 | 0.957 | 0.957 | 0.942 | 0.6163 |
| Fold 5 | 0.985 | 0.971 | 0.971 | 0.6163 |
| Average | 0.963 | 0.960 | 0.960 | 0.6138 |

The performance of the simple linear network was the same as the FFN with two hidden layers, with a one-tailed hypothesis (t(8) = 0.31623, $p$ = 0.379961, not significant at $p$ <0.05).

The performance of the simple linear network was the same as the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = 0.33516, $p$ = 0.373062, not significant at $p$ <0.05).

The performance of the simple linear network was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = 48.07172, $p$ = 0.00001, significant at $p$ <0.05).

The performance of the FFN with two hidden layers was the same as the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = 0.02658, $p$ = 0.489721, not significant at $p$ <0.05).

The performance of the FFN with two hidden layers was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = 64.11036, $p$ = 0.00001, significant at $p$ <0.05).

The performance of the FFN with two hidden layers where the first layer was replaced with an autoencoder was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = 62.26135, $p$ = 0.00001, significant at $p$ <0.05).

### 3.2 Car Evaluation Data Set Results

SLN : Network trained at learning rate 0.8 over 500 epochs.

FFN-2HL: Network trained with 5 hidden nodes in the first layer, 5 hidden nodes in the second layer, learning rate 0.07 over 70 epochs.

FFN-AE: Autoencoder: trained with 5 hidden nodes, learning rate 0.03, over 10 epochs. Network trained with 5 hidden nodes in the second hidden layer, learning rate 0.07 over 70 epochs.

| Fold | SLN | FFN-2HL | FFN-AE | Simple |
|---|---|---|---|---|
| Fold 1 | 0.824 | 0.881 | 0.845 | 0.699 |
| Fold 2 | 0.817 | 0.896 | 0.878 | 0.699 |
| Fold 3 | 0.841 | 0.888 | 0.895 | 0.699 |
| Fold 4 | 0.811 | 0.847 | 0.84 | 0.699 |
| Fold 5 | 0.815 | 0.822 | 0.898 | 0.703 |
| Average | 0.821 | 0.867 | 0.871 | 0.700 |

The performance of the simple linear network was worse than the FFN with two hidden layers, with a one-tailed hypothesis (t(8) = -3.02614, $p$ = 0.008203, significant at $p$ <0.05).

The performance of the simple linear network was worse than the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = -3.72291, $p$ = 0.002923, significant at $p$ <0.05).

The performance of the simple linear network was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = 22.77531, $p$ = 0.00001, significant at $p$ <0.05).

The performance of the FFN with two hidden layers was the same as the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = -0.237, $p$ = 0.409307, not significant at $p$ <0.05).

The performance of the FFN with two hidden layers was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = 11.93527, $p$ = 0.00001, significant at $p$ <0.05).

The performance of the FFN with two hidden layers where the first layer was replaced with an autoencoder was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = 13.98636, $p$ = 0.00001, significant at $p$ <0.05).

### 3.3 Breast Cancer Data Set Results

SLN : Network trained at learning rate 0.1 over 150 epochs.

FFN-2HL: Network trained with 8 hidden nodes in the first layer, 8 hidden nodes in the second layer, learning rate 0.01 over 40 epochs.

FFN-AE: Autoencoder: trained with 8 hidden nodes, learning rate 0.002, over 100 epochs. Network trained with 8 hidden nodes in the second hidden layer, learning rate 0.01 over 40 epochs.

| Fold | SLN | FFN-2HL | FFN-AE | Simple |
|---|---|---|---|---|
| Fold 1 | 0.912 | 0.655 | 0.655 | 0.652 |
| Fold 2 | 0.973 | 0.973 | 0.652 | 0.657 |
| Fold 3 | 0.919 | 0.658 | 0.658 | 0.657 |
| Fold 4 | 0.973 | 0.973 | 0.658 | 0.655 |
| Fold 5 | 0.946 | 0.946 | 0.658 | 0.655 |
| Average | 0.944 | 0.841 | 0.656 | 0.655 |

The performance of the simple linear network was the same as the FFN with two hidden layers, with a one-tailed hypothesis (t(8) = 1.35283, $p$ = 0.10655, not significant at $p$ <0.05).

The performance of the simple linear network was greater than the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = 22.24392, $p$ = 0.00001, significant at $p$ <0.05).

The performance of the simple linear network was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = 22.361, $p$ = 0.00001, significant at $p$ <0.05).

The performance of the FFN with two hidden layers was greater than the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = 2.44788, $p$ = 0.020035, significant at $p$ <0.05).

The performance of the FFN with two hidden layers was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = 2.46125, $p$ = 0.019621, significant at $p$ <0.05).

The performance of the FFN with two hidden layers where the first layer was replaced with an autoencoder was the same as the simple majority algorithm, with a one-tailed hypothesis (t(8) = 0.66227, $p$ = 0.263207, not significant at $p$ <0.05).

### 3.4 Forest Fires Data Set Results

SLN : Network trained at learning rate 0.02 over 60 epochs.

FFN-2HL: Network trained with 13 hidden nodes in the first layer, 17 hidden nodes in the second layer, learning rate 0.02 over 40 epochs.

FFN-AE: Autoencoder: trained with 12 hidden nodes, learning rate 0.02, over 80 epochs. Network trained with 17 hidden nodes in the second hidden layer, learning rate 0.02 over 20 epochs.

| Fold | SLN | FFN-2HL | FFN-AE | Simple |
|---|---|---|---|---|
| Fold 1 | 1127.64 | 8186.12 | 1172.46 | 6003.36 |
| Fold 2 | 569.58 | 1492.65 | 7148.75 | 564.74 |
| Fold 3 | 6678.46 | 15074.93 | 15064.46 | 968.04 |
| Fold 4 | 15120.96 | 610.18 | 218.71 | 11677.79 |
| Fold 5 | 1167.54 | 479.12 | 645.21 | 1072.27 |
| Average | 4932.84 | 5168.60 | 4849.92 | 4057.24 |

The performance of the simple linear network was the same as the FFN with two hidden layers, with a one-tailed hypothesis (t(8) = -0.05912, $p$ = 0.477153, not significant at $p$ <0.05).

The performance of the simple linear network was the same as the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = 0.02083, $p$ = 0.491944, not significant at $p$ <0.05).

The performance of the simple linear network was the same as the simple majority algorithm, with a one-tailed hypothesis (t(8) = 0.24913, $p$ = 0.404769, not significant at $p$ <0.05).

The performance of the FFN with two hidden layers was the same as the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = 0.07896, $p$ = 0.469502, not significant at $p$ <0.05).

The performance of the FFN with two hidden layers was the same as the simple majority algorithm, with a one-tailed hypothesis (t(8) = 0.31061, $p$ = 0.382017, not significant at $p$ <0.05).

The performance of the FFN with two hidden layers where the first layer was replaced with an autoencoder was the same as the simple majority algorithm, with a one-tailed hypothesis (t(8) = 0.22209, $p$ = 0.414905, not significant at $p$ <0.05).

### 3.5 Computer Hardware Data Set Results

SLN : Network trained at learning rate 0.0002 over 15 epochs.

FFN-2HL: Network trained with 5 hidden nodes in the first layer, 4 hidden nodes in the second layer, learning rate 0.0003 over 120 epochs.

FFN-AE: Autoencoder: trained with 5 hidden nodes, learning rate 0.05, over 25 epochs. Network trained with 4 hidden nodes in the second hidden layer, learning rate 0.0003 over 120 epochs.

| Fold | SLN | FFN-2HL | FFN-AE | Simple |
|---|---|---|---|---|
| Fold 1 | 4419.27 | 20269.21 | 10474.23 | 10648.78 |
| Fold 2 | 3501.63 | 2231.40 | 6751.28 | 16765.44 |
| Fold 3 | 5944.87 | 24889.24 | 47827.83 | 14884.05 |
| Fold 4 | 9574.23 | 5841.39 | 1708.22 | 29963.10 |
| Fold 5 | 13200.83 | 2759.18 | 4756.02 | 57598.05 |
| Average | 7328.16 | 11198.08 | 14303.52 | 25971.88 |

The performance of the simple linear network was the same as the FFN with two hidden layers, with a one-tailed hypothesis (t(8) = -0.76293, $p$ = 0.233702, not significant at $p$ <0.05).

The performance of the simple linear network was the same as the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = -0.80279, $p$ = 0.222644, not significant at $p$ <0.05).

The performance of the simple linear network was better than the simple majority algorithm, with a one-tailed hypothesis (t(8) = -2.13634, $p$ = 0.032573, significant at $p$ <0.05).

The performance of the FFN with two hidden layers was the same as the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = -0.319, $p$ = 0.37895, not significant at $p$ <0.05).

The performance of the FFN with two hidden layers was the same as the simple majority algorithm, with a one-tailed hypothesis (t(8) = -1.51231, $p$ = 0.084454, not significant at $p$ <0.05).

The performance of the FFN with two hidden layers where the first layer was replaced with an autoencoder was the same as the simple majority algorithm, with a one-tailed hypothesis (t(8) = -0.96834, $p$ = 0.180618, not significant at $p$ <0.05).

### 3.6 Abalone Data Set Results

SLN : Network trained at learning rate 0.1 over 20 epochs.

FFN-2HL: Network trained with 6 hidden nodes in the first layer, 5 hidden nodes in the second layer, learning rate 0.003 over 50 epochs.

FFN-AE: Autoencoder: trained with 9 hidden nodes, learning rate 0.2, over 40 epochs. Network trained with 5 hidden nodes in the second hidden layer, learning rate 0.003 over 40 epochs.

| Fold | SLN | FFN-2HL | FFN-AE | Simple |
|---|---|---|---|---|
| Fold 1 | 7.131 | 4.558 | 4.566 | 10.1432 |
| Fold 2 | 7.058 | 5.485 | 4.791 | 10.2588 |
| Fold 3 | 7.897 | 4.673 | 4.297 | 10.0823 |
| Fold 4 | 8.059 | 4.643 | 4.025 | 10.5208 |
| Fold 5 | 7.656 | 3.887 | 5.651 | 10.9701 |
| Average | 7.561 | 4.649 | 4.666 | 10.3951 |

The performance of the simple linear network was worse than the FFN with two hidden layers, with a one-tailed hypothesis (t(8) = 8.99154, $p$ = 0.00001, significant at $p$ <0.05).

The performance of the simple linear network was worse than the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = 8.44195, $p$ = 0.000015, significant at $p$ <0.05).

The performance of the simple linear network was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = 10.976, $p$ = 0.00001, significant at $p$ <0.05).

The performance of the FFN with two hidden layers was the same as the FFN with two hidden layers where the first layer was replaced with an autoencoder, with a one-tailed hypothesis (t(8) = -0.04465, $p$ = 0.48274, not significant at $p$ <0.05).

The performance of the FFN with two hidden layers was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = -19.07489, $p$ = 0.00001, significant at $p$ <0.05).

The performance of the FFN with two hidden layers where the first layer was replaced with an autoencoder was greater than the simple majority algorithm, with a one-tailed hypothesis (t(8) = -17.81133, $p$ = 0.00001, significant at $p$ <0.05).

## 4. Discussion

Overall, results were unexpected. though none of the algorithms performed worse than the simple average/majority algorithm, the comparative performance between the neural network algorithms were not in line with what was originally hypothesized.

The experiments that ran closest to expectations were the Car Evaluation data set and the Abalone data sets. In these experiments, all of the linear networks greatly outperformed the simple average/majority algorithm. Further, the deep neural networks with two hidden layers outperformed the simple linear network. However, there was no significant difference between the performance of the FFN with two hidden layers and the autoencoder enhanced FFN.

On the other end of the spectrum, the Forest Fires data set saw that the performance of the neural network algorithms were not at all better than the simple average algorithm. Not a single neural network algorithm managed to outperform the simple algorithm. Though multiple randomized runs and calibrations were conducted for this data set, as well as validation error being far lower than the displayed results, the final testing error was statistically indistinguishable from the simple average algorithm.

In the middle of the road were the Computer Hardware and House Votes data sets. All of the neural network algorithms outperformed the simple average/majority algorithm. However, there was no significant difference between the simple linear network and the deep neural networks. The performance across all three networks was essentially the same.

The most strange performance was seen with the Breast Cancer data set. Though none of the neural network algorithms were beaten by the simple average/majority algorithm, the deep neural networks were outperformed by the simple linear network. The autoencoder enhanced FFN was the worst performing neural network out of the three, with performance equivalent to the simple average/majority algorithm. For the deep neural networks, learning curves were plotted to get a deeper look at why performance was not up to expectations. As can be seen in Folds 1 and 3, of the FFN with two hidden layers, the network did not manage to descend the gradient at all, and accuracy did not improve across training epochs – accuracy remained at ∼65% for all epochs for both the validation and training set. As for the folds that did reach high performance, the plotted learning curve were entirely flat, up to a point where performance would jump from ∼65% to ∼95% in a single epoch. The reason for this behavior is not clear. Different trials were ran with varying learning rates, epochs, and hidden layer node sizes. All other parameter settings would result in the first scenario, where there was no gradient descent.

It is perhaps the case that certain problems do not benefit from adding complexity. The Breast Cancer data set saw high performance from just the simple linear network– the addition of more hidden layers seemed to hinder performance rather than improve it. The Forest Fires data set was still the most difficult data set to model. No reasonable change in learning rate, hidden layer size, or epochs, guaranteed reliable performance on this data set.

Overall, we expected that the autoencoder enhanced FFN would not improve performance on the regression or classification tasks. Though we were able to map the inputs from the data sets into a lower dimensional space in such a way that they are able to be reconstructed, it does not necessarily mean that the neural network is in a better position to solve the problem than it was before. In other words, even though we can conduct feature extraction, it is no guarantee that those features are more useful.

## 5. Conclusion

In this paper, we have conducted experiments on three different kinds of neural networks and compared their performance. We have looked at the performance of a simple linear network, as compared to a neural network with two hidden layers, as well as a neural network with two hidden layers, where the first layer is trained using an autoencoder. We found that all of the neural networks performed better than baseline, though the depth of the neural network did not guarantee a performance boost. We also found that the autoencoder was not very helpful for improving results.