Supervised Learning using Decision Trees and Neural Networks

***Abstract*—This paper presents two methods of supervised learning: learning by use of decision trees and learning by use of neural networks. By comparing two types of each, some of the parameters typically employed and the structures of each can be explored, providing some analysis of their import in the accuracy and efficiency of learning. The goal of this report is twofold: to test an implementation of each supervised learning method and collect information on idealized parameters and/or structures of each.**

Keywords—ID3, C4.5, decision trees, stochastic gradient descent, backpropagation, resilient backpropagation, neural networks, learning, classification, training, testing, validation.

I. Introduction

Machine learning is a discipline of computer science which focuses on artificial intelligence algorithms which employ statistical models and algorithms. The general process of any machine learning application or implementation is the collection of data, processing said data, and using that data to train some algorithm to generate a model [1]. These models are often used to classify or make predictions about novel data.

Two popular types of machine learning are decision trees and neural networks, both of which are commonly used to classify data. This report will cover a specific type of these models: supervised decision trees and neural networks.

The difference in unsupervised and supervised learning is simple: with supervised learning, information about the data, such as classification or output, is known [2]. Unsupervised, contrastingly, is when this information is either incomplete or not know. Unsupervised learning is more so for learning new information about the data (for example, clustering to find similar classifications) and instead supervised learning is to model the known data.

Both decision trees and neural networks are used to this effect to model a system where novel information can then be classified once enough information about the data is gathered. While the model is created using known data, it can classify unknown data once trained (and provided adequate parameters in model creation are used).

II. Supervised Learning

A. Decision Trees

Decision trees, a type of supervised learning, is as its name implies, a tree-based structure used for the classification of data. You might use decision tree based learning if the data you wish to classify is constructed of instances describable by attribute-value pairs [3]; in other words, you have an explicit list of attributes or features on which the describe the data, as well as a decision value classifying the data.

To use an example of flower classification, many "knowns" are known about an example: the stem length, the petal color, et cetera, as well as an answer to the species or genus of plant. Using known data like this can generate a model in which novel instances of data can be generalized and classified.

Figure I.

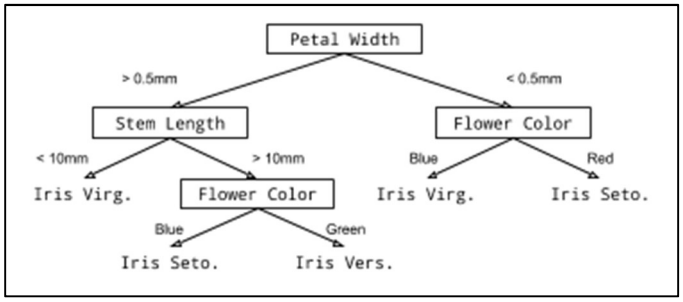


Fig. I. An example decision tree.

The tree structure of a decision tree is human readable and (more or less) simple to understand. Each node of the tree can be considered a "question" about the data, and each "answer" will lead to further nodes. Terminal nodes contain decisions which is the classification for the novel data; it can be a binary yes and no decision, or one with more granularity.

One example of a common decision tree model algorithm is the ID3 (Iterative Dichotomizer 3) algorithm, as invented by Ross Quinlan. This algorithm begins with a root node and successively splits the data based on data attributes such that it can correctly classify data on which it was trained upon:

```
procedure ID3 (train, dec, attributes):
  A := root note
  if ∀train > 0: return A as +
  else if ∀train < 0: return A as -
  else if train is empty: return A
  A := max(IG(attributes))
  for each value v of A:
    A += branch b for value v
    for each example e in train with value = v:
      if e is empty: b := dec
      else: A += ID3(train(v), dec, attributes – A)
  return A
```

Where `IG()` is a function which returns the information gain of a potential split on an attribute. Recursively, the `ID3()` function divides training data on attributes which provide the idealized splits and generates a tree using said splits, until there are not more attributes to divide upon.

Another example of a decision tree model algorithm is the C4.5 algorithm [4], which is an improvement over the ID3 algorithm by the ability to classify both continuous and discrete attributes. The algorithm is not too dissimilar to the ID3 algorithm, although with a few improvements:

```
procedure C4.5 (train, dec, attributes):
  A := root note
  if ∀train > thresh+: return A as +
  else if ∀train < thresh-: return A as -
  else if train is empty: return A
  A := max(IG(attributes))
  for each value v of A:
    if v is categorical:
      A += distinct categories of v
    else v is numerical:
      A += branch b for value v
    for each example e in train with value = v:
      if e is empty: b := dec
      else: ID3(train(v), dec, attributes – A)
  return A
```

Additionally, the C4.5 algorithm recurses through the tree structure to prune branches where a decision may better be classified elsewhere in the tree.

For both algorithms, an important statistic needed is the information gain (IG) of an attribute. When a split is made in the tree to further categorize data, this is not done blindly; instead, the algorithm chooses an attribute to split upon based on the maximum of this value among attributes [3]:

$$IG(S, A) = H(S) - H(S|A)$$

$$H(S) = \sum_{x \in X} (-p(x) * \log_2 p(x))$$

Each split creates a hypothesis and these split hypotheses are validated using the information gain equation. Not dissimilar to a Minimax or similar tree search, the potential information gain on a split is evaluated and the algorithm chooses the best one [3]. By performing this recursively, the tree is generated using the best hypothesis for each branch.

B. Neural Networks

Neural networks, contrastingly, have a biological component to them. While it is also a machine learning technique, the goal of a neural network is to realize a system using the human brain as a model [8]. Not unlike the human brain, it is comprised of many individual parts which may not be much on their own, but the sum of its parts is far greater than the individual.

While decision trees are relatively straight forward and can be somewhat summarized as just a continued junction on attributes, neural networks are more abstract. While not technically a black box, their internal workings are not quite human readable barring rudimentary and/or small systems.

Neural networks have many applications but are classically used for classification problems. Questions such as "does this piece of data belong to this category of data".

The specific type of neural network which this report will explore are *feed forward neural networks*. This is a network structure where information travels through each "layer", starting from the "input layer" to the "output layer". A layer in this regard is a series of neurons [2]:

FIGURE II.

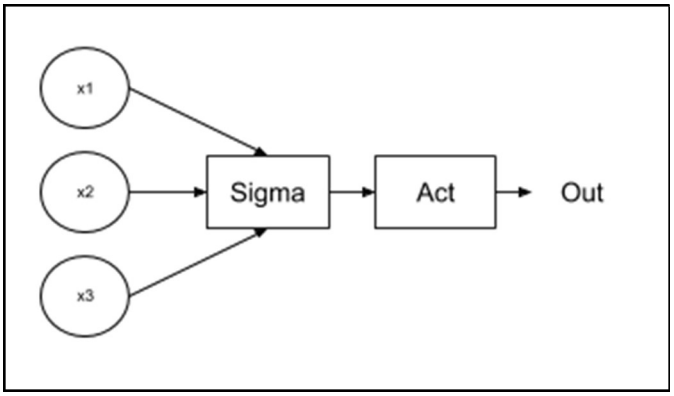


Fig. II. An example neuron.

In the above figure, the neuron comprises of several components, from left-to-right: the neural inputs, the summation function, the activation function, and the neural output. Connecting the neural inputs to the activation function are synapses which are weighted.

The general idea of the artificial neuron is that it takes an input, is transferred to the summation function through the synapses, and the summation function sums the input values depending on the weights of its connected synapses. This can be a ratio of all three inputs or all or none of any given input. The number if inputs is not restricted to any number and can be as large as necessary to fit the problem space the neural network attempts to solve.

The summation function then transfers its own output into the activation function: the purpose of this function is to determine if the neuron will "fire", or to what degree it should "fire". This probability is related to whether each neuron's input is relevant for future prediction [2].

FIGURE III.

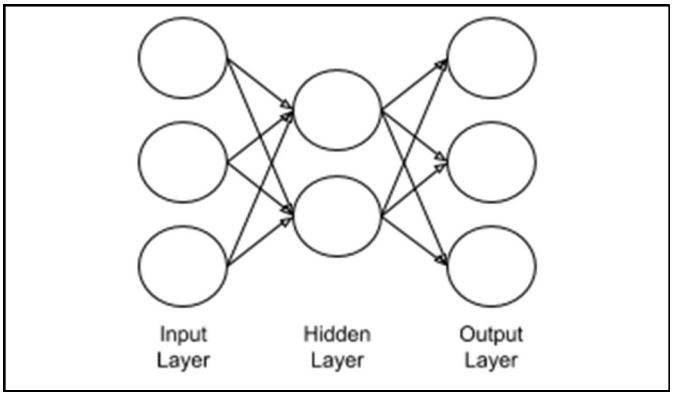


Fig. II. An example network in a 3-2-3 structure.

Artificial neurons are often linked together to form a network. This process of information being transported through the neuron to subsequent neurons is referred to as feedforward.

The structure of a neural network can vary but a feedforward neural network is acyclic and directed; in other words, input information is fed forward through the network but never in the reverse direction. As in the previous figure, this network consists

of three layers: an input layer (where neurons receive information directly from data), a hidden layer (where neurons receive information from the input layer), and an output layer (where neurons receive information from the hidden layer) [2].

An addition to feedforward neural networks is backpropagation. As input information passes through the network, the output values are stored within the neuron. For the first forward pass of the network, each neuron only contains its own outputs and synapse weights. But, when this information reaches the final layer, the output layer, is when backpropagation begins.

In order to improve the performance of a neural network in classification or any other problem, it uses an error function which is a function of what a neuron receives and then sums and activates to output, and from what it is expected to output. For data the network is trained upon, the expected output is embedded within the data as a classification of known data [9][11]. Using the error function, a gradient is found.

The concept of gradient descent is intertwined with neural networks. Comparing the actual output of the network against an expected output, a gradient is found [2]: this gradient is used to determine by how much to adjust the weights of each synapse in a backward pass—backpropagation. The error from the output layer is passed to the hidden layer, whereupon the synapses connecting the two alter, and the cycle repeats until the input layer is then reached [10].

At this point, the weights have changed and provided good parameters are used for the network, the weights change in a direction which decreases the error on subsequent pieces of data. The goal here, is to minimize the error function by updating the weights intelligently.

A rough step-by-step for the feedforward network with backpropagation procedure is as follows:

```
procedure train_network:
  generate network:
    initialize each neuron weight as random -1..1
  for each training epoch do:
    for each training example do:
      forward_pass example through network
      back_propagate error through each prior layer
      update_weights

procedure forward_pass:
  for each layer in network do:
    for each neuron in layer do:
      neuron output := activation(summing(weights)

procedure back_propagate:
  for each layer in network do:
    for each neuron in layer do:
      if output layer:
        error := expected – actual output
      else:
        error := function of weight and delta
      delta := derivative(neural output)

procedure update_weights:
    for each layer in network do:
      for each neuron in layer do:
        change := function of example and delta
        weights += learn * change + moment * delta
```

In the forward pass, neural outputs are calculated as a function of the activation and summing functions, whereas in the backward pass, the delta (change needed) is measured by the derivative of the activation function. Two common activation functions are the logistic sigmoid and hyperbolic tangent functions.

Finally, the weights are updated as a function of the current weight, the example, and the delta as previously found. In the update weights procedure, there exist two parameters: learning rate and momentum.

The learning rate, as a user-defined parameter which needs to be empirically found, determines how much the neuron should learn about the currently iterated example. A low learning rate means, for example, that little information about the example is learned, whereas a high learning rate considers a larger portion of the example [11].

The momentum rate needs to consider temporal delta, or the prior delta of the neuron. It is the "inertia" that prior examples have on the current iteration; in other words, some of the previous example's impact is still felt even during subsequent iterations.

It is useful to think of network training as scaling a hill with bumps and crevices. The learning rate will determine how large the steps are in scaling this figurative hill whereas the momentum rate will determine how much inertia movement has, potentially allowing the error gradient to leave local minima [9]. The goal of the network is to reach a global minimum rather than a local one.

Once these steps have finished, the cycle repeats with further training examples. Over time, and provided the parameters specified are appropriate, the network should converge to a solution and find the global minimum of the problem space. To this end, the network would have been trained and is ready to accept novel input.

III. Methodology (Decision Trees)

The data used to test the decision tree algorithms is the SPECT Heart data set, which is separated into a training and testing portion consisting of 80 and 133 examples respectively. Each tuple in these data sets contains 22 features and one of two classifications.

The ID3 algorithm is as implemented but the C4.5 algorithm is provided by Weka, an open source experimenter for statistics and machine learning.

In order to test the efficacy of both the ID3 and C4.5 algorithms, repeated runs of the algorithms are performed with varying hold-out rates. A hold-out rate is a split on training data whereupon a portion (the hold-out) is taken as training data and the remainder becomes testing data.

TABLE I. HOLDOUT RATES

Holdout Rate
70-30
50-50
30-70
90-10
60-40

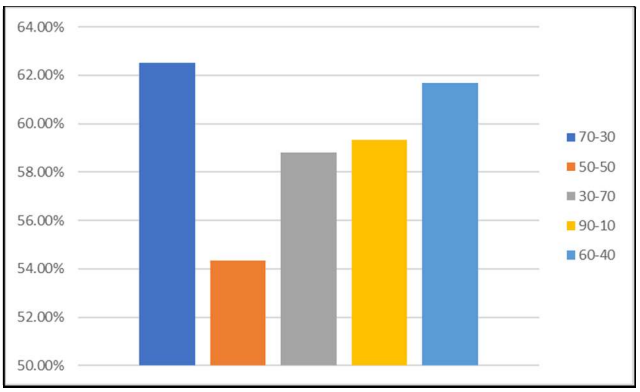
Table I. Variable holdout rates to be tested by ID3 and C4.5.

Using the (larger) testing data as training data, varying the hold-out rate should show an ideal parameter provided enough runs are made. At this point, a direct comparison between the ID3 and C4.5 algorithms can be done. Additionally, the pruning aspect of C4.5 will be examined as well as the execution time of each algorithm.

IV. Analysis (Decision Trees)

On testing various hold-out rates, the ID3 algorithm demonstrates lukewarm performance in classifying the testing portion of the testing hold-out:

Graph I. ID3 Mean Accuracy



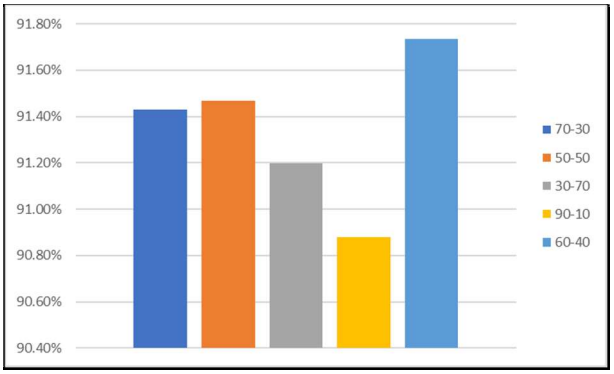
Graph I. ID3 mean accuracy in classifying testing data.

With a mean execution time of 2.43 seconds, the ID3 algorithm as implemented is accurate in the 55-65% range with an optimal hold-out ratio being 70% training and 30% testing. There is no correlation to the size of training data and the performance for these hold-outs tested but the data does suggest more training data is generally greater with outliers. A 60-70% training ratio may prove fruitful as a general guideline but the exact ratio may be problem specific or specific to the number of classifications (recall a binary classification would have 50% accuracy if chosen randomly).

To provide insight to the size of the ID3 generated decision trees, it is found that on average across all hold-out ratios, the tree ranges between 40-70 nodes. The ID3 algorithm has no pruning capability as implemented.

To contrast, the C4.5 algorithm performs much greater:

Graph II. C4.5 Mean Accuracy



Graph II. C4.5 mean accuracy in classifying testing data.

In testing these hold-out ratios, a typical execution time took less than 0.2 seconds and the accuracy is much greater. Comparing the means as shown above, a 60-40 hold-out rate is determined to provide the greatest accuracy of classification.

Interesting is the pruning capability of the C4.5 algorithm. In a majority (95%+) of training instances, the tree was able to be pruned to 1 node. Despite needing to generate a tree as ID3 would, and then recursively prune it, the execution time is monumentally shorter.

V. Conclusion (Decision Trees)

Considering both algorithms, it is appropriate to use C4.5 over ID3 provided the circumstances allowed it: it is faster, able to prune to a high degree, and in general performs better. This is, of course, specific to the SPECTRE Heart data set, so these conclusions may not be indicative of every problem space.

On decision trees as a whole, it can be a speedy method to create a classification system, although like neural networks, relies on adequate parameters (test/train split) and ample examples of training data.

VI. Methodology (Neural Networks)

The data set used is the MNIST Digit data set. It consists of 700 examples of each numeral zero through nine, as a CSV file comprised of 64 features each along with a classification. As a result of this, the input layer is locked to 64 neurons and output layer as 10 neurons.

In order to determine an idealized network, the parameters themselves must be tested alongside the network structure. Each test for ideal parameter will have other parameters set as constants, which will be found sequentially as each test is analyzed. Every test is repeated over fifteen trials to maintain statistical relevance.

For tests, the MSE (Mean Standard Error) is used in lieu of accuracy because it is more granular and simply easier to perform the statistical tests employed in this section. MSE is inversely proportional to accuracy ($r \sim -0.999$), thus MSE is an excellent indicator of system performance.

A. Hidden Layer Structure

Before other parameters can be tested, the network structure itself must be established. However, since the accuracy of the network relies on the other parameters, some will be chosen arbitrarily to give a general idea of how structure affects performance irrespective of other parameters. Below are the structure types to be analyzed:

Table I. Hidden Layer Variables

Hidden Neurons	Learning Rate	Momentum
1	0.001 – 1.000	0.000
5	0.001 – 1.000	0.000
10	0.001 – 1.000	0.000
15	0.001 – 1.000	0.000
20	0.001 – 1.000	0.000
25	0.001 – 1.000	0.000
30	0.001 – 1.000	0.000
35	0.001 – 1.000	0.000

Table I. Hidden Layer structures to be tested.

The learning rate is variable as without an appropriate learning rate parameter chosen, accuracy reported for the hidden structure may not indicate anything of significance, as several permutations of hidden neurons and learning rates may perform adequate. To this end, a learning rate of 0.001, 0.005, 0.010, 0.050, 0.100, 0.500, and 1.000 are averaged to test the hidden structure size.

The momentum rate, however, is kept at a constant 0.000. This test is repeated for both the logistic sigmoid activation function and the hyperbolic tangent activation function to determine if there exists a difference in idealized hidden layer structure between each function.

To determine the relevance of the hidden structure in total system performance, a Kruskal-Wallis test is performed and the means of each value for hidden structure will be found to determine which structure should be used for successive tests.

A Kruskal-Wallis test will determine whether the means of several samples (in this case, 8 samples derived from the mean results of 120 sets of data collected) are significantly different. A significant difference in means would suggest that the hidden layer structure has a large impact on system performance whereas an insignificant test statistic would suggest there is loose impact on performance.

To perform this test, however, presupposes each sample is homogenous and does not follow a normal distribution (else ANOVA would be used). Homogeneity means that the variances between samples is not significantly different. This is done using Levene's Test for homogeneity and Shapiro-Wilk test for normalcy. In short, what these tests provide is a measure of how much variance is within a sample as well as how divergent it is from a normal distribution.

For successive subsections for methodology, the hidden structure will be defined as *n*, as it is currently an unknown value. As it will be established in the results section for the hidden layer structure tests, for now the number of hidden neurons is a placeholder. Were the ANOVA test to suggest that there is no significance between samples, an arbitrary hidden layer size will be chosen.

The anticipated outcome of this test, as suggested by literature [5] is that there is a minimum number of hidden neurons for there to be a viable system, but also a maximum where continued increase in layer size will provide diminishing returns and eventually no significant advantage in performance. For the specific problem of digit recognition using the MNIST Digits data set, this report will attempt to empirically determine an idealized structure of the hidden layer.

B. Learning Rate

Once the hidden layer structure has been established, the learning rate can be varied to determine its import on system performance. Like with the hidden layer size, further parameters rely on adequate learning rate and therefore will be kept as a placeholder for future sections. Below are the learning rate variables to be tested:

TABLE II. LEARNING RATE VARIABLES

Hidden Neurons	Learning Rate	Momentum
n	0.001	0.000
n	0.005	0.000
n	0.010	0.000
n	0.050	0.000
n	0.100	0.000
n	0.500	0.000
n	1.000	0.000

Table II. Learning rate parameters to be tested.

As previously mentioned, the hidden neuron count is not yet defined, and the momentum rate is constant. This test is meant specifically to compare the various learning rates and determine which best fits the system.

Once results are gathered another Kruskal-Wallis test is performed with its prerequisite normalcy and homogeneity tests. Not unlike with the test for hidden layer size, this test is intended to determine if there exists a statistical significance to the learning rate.

Like before, successive tests in methodology will require there to be concrete decision for learning rate. Instead, a placeholder will again be used, *m*, to denote idealized learning rate. This test is repeated for both activation functions and were the test not prove fruitful as with the hidden layer size test, an arbitrary learning rate will be used for successive tests.

The anticipated outcome of this test is that there should be a minute learning rate although there may be some slight success with a larger one. A terribly inappropriate learning rate may not cripple the system, but there is a narrow window on which the learning rate is ideal or within the range of ideal [6]. An exorbitantly high learning rate, however, may prevent the system from learning at all.

C. Momentum Rate

The momentum rate test is essentially a repeat of the learning rate test, although at this point both the hidden layer size and learning rate which produced the greatest accuracy would have been found. Because of this fact, there are no other unknown constants and instead the rest of the system is known. Below are the momentum rate variables to be tested:

TABLE III. MOMENTUM RATE VARIABLES

Hidden Neurons	Learning Rate	Momentum
n	m	0.001
n	m	0.005
n	m	0.010
n	m	0.050
n	m	0.100
n	m	0.500
n	m	1.000

Table III. Momentum rate parameters to be tested.

And as previously mentioned, were the Kruskall-Wallis test be an inappropriate statistical test for this parameter, ANOVA will instead be performed provided the samples are not normal and homogenous. The significance of this test is to determine if there is any statistical relevance in momentum rate affecting the

system performance. Unlike before, however, once an idealized momentum rate is known, it is probable the best parameters for the MNIST Digit recognition problem are known.

Failing this, however, were the test to show little statistical significance in momentum rate chosen, as an exercise in completeness, conditions where the momentum rate would have a larger impact will be demonstrated.

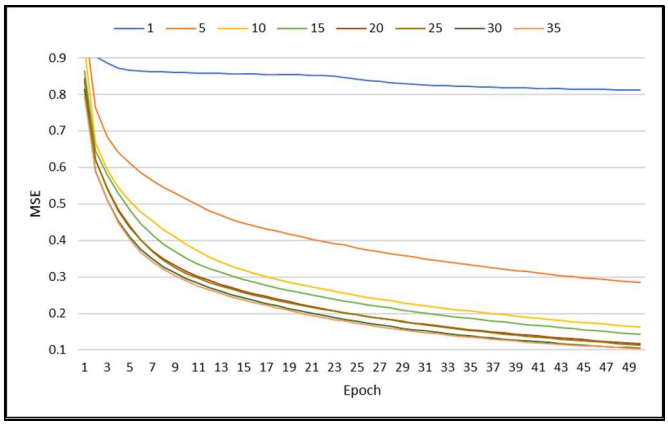
Supposing the hidden layer structure and learning rate parameters are ideal, it is hypothesized that the momentum rate parameters is of little consequence to system performance. In other words, any choice in momentum may have negligible impact or even possibly a negative impact on system performance.

VII. Analysis (Neural Networks)

A. Hidden Layer Structure

After collection of data for each hidden layer structure, the below graph is generated displaying the system error over the training epochs for the logistic sigmoid activation function:

Graph III. Log Hidden Layer Data



Graph III. Collected data comparing hidden layers, log.

Visually, it is apparent there is a window in which the hidden layer size produces similar results along with a few outlier structures provided worse results.

To verify this, however, first a test for homogeneity is needed: using Levene's test for each pair of samples, the mean variance is found to be 0.0954 alongside an f-ratio of 0.05082. The p-value from this test is 0.82211, dramatically higher than the null hypothesis of $p < 0.05$, therefore the requirements for homogeneity are met.

Secondly, a test for normalcy is performed: using Shapiro-Wilks test, it is found that no sample is normally distributed, with an average W of 0.77 (significance at 0.95). Each p-value is infinitesimally small as well, which is much smaller than the null hypothesis of $p < 0.05$. For the hidden layer test, each sample is neither normal nor homogenous which fulfills the conditions for a Kruskal-Wallis test.

Comparing the samples using the Kruskal-Wallis test, it suggests that there is significant variance between means between each sample with a p-value < 0.01. To enumerate the means and variances:

Table IV. Mean and Variance of HL Test

Hidden Neurons	Mean	Variance
1	0.8443	0.0013
5	0.4196	0.0192
10	0.2961	0.0220
15	0.2733	0.0204
20	0.2413	0.0198
25	0.2391	0.0202
30	0.2226	0.0189
35	0.2181	0.0185

Table IV. Mean and variance per hidden layer, log.

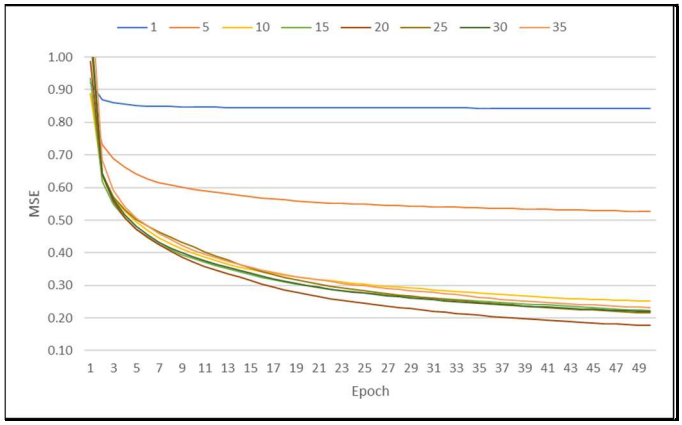
However, outliers can be identified by finding the interquartile range for the set of means. For these sample means and variances, the interquartile range is found:

$$IQR = 0.388 - 0.226 = 0.162$$

Finding the interquartile range allows the dispersion of the means to be identified and identify outliers, of which there are three hidden layer sizes: 1, 5, and 35. Re-examining the remaining samples using the Kruskal-Wallis test describes a different result than before: with a p-value of $0.024 > 0.01$, there is little significance found by the test, and any of the values for this parameter may be used with minimal loss to performance. To this end, a hidden layer size of 20 is chosen for the logistic sigmoid activation function.

This process is repeated for the hyperbolic tangent activation function: data is collected, and a graph is constructed:

Graph IV. Tanh Hidden Layer Data



Graph IV. Collected data comparing hidden layers, tanh.

Again, testing for homogeneity using Levene's test, there is statistical significance with a mean p-value of 0.85495 whereas a p-value < 0.05 would suggest non-homogeneity. Therefore, the requirements of homogeneity are met. Using Shapiro-Wilks test, normalcy is determined: a mean W of 0.65 (significance at 0.95) indicates the samples are not normal, as before.

Comparing the samples using the Kruskal-Wallis test, it suggests there is significant variance between means between each sample with a p-value of < 0.01. To enumerate the means and variances:

TABLE V. MEAN AND VARIANCE OF HL TEST

Hidden Neurons	Mean	Variance
1	0.8472	0.0001
5	0.5706	0.0039
10	0.3423	0.0135
15	0.3207	0.0155
20	0.2925	0.0209
25	0.3319	0.0212
30	0.3242	0.0217
35	0.3461	0.0253

Table V. Mean and variance per hidden layer, tanh.

And once again outliers can be identified using the same method as before:

$$IGR = 0.514 - 0.322 = 0.192$$

In this, there are three outlier hidden layer sizes: 1, 5, and 20. Re-examining the remaining samples using Kruskal-Wallis test provides a p-value of 0.11 which is above the significant p <0.01, therefore there is little significance in the difference of means judging by the test results. A hidden layer size is chosen of 15 to represent the hyperbolic tangent function.

Upon inspection of the means and variances for both chosen structures, there is a clear minimum at which the performance of the system becomes adequate.

In these cases, the logistic sigmoid function requires 10-30 hidden neurons with outsiders to this range either not meeting a performance threshold or providing diminishing returns. For the hyperbolic tangent function, similar results are found: the system requires at minimum 10 hidden neurons to minimize error.

It can be inferred that there is a range wherein the hidden layer size has ideal effect on system performance and that using Pearson's Correlation Coefficient test, the two activation functions can be directly compared: the means as found earlier are used as data for the calculation:

$$r = \sum((X - My)(Y - Mx)) / \sqrt{((SSx)(SSy))}$$

Here, the correlation coefficient is found to be 0.9752 with a p-value < 0.00001, suggesting very high correlation of statistical significance. This only affirms the hypothesis that from within the range tested for hidden layer size (1-35), both activation functions lead to similar results.

At this point, this information is known:

TABLE VI. KNOWN PARAMETERS

	Hidden Neurons	Learning Rate	Momentum Rate
log	20	?	?
tanh	15	?	?

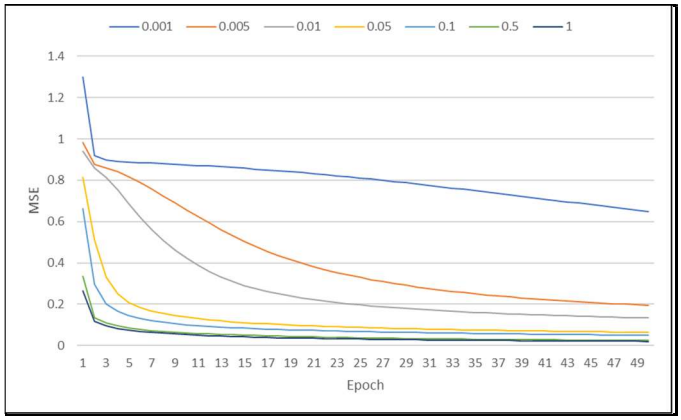
Table VI. Idealized Parameters, log and tanh.

B. Learning Rate

Since the hidden layer size has been established as 20 for the logistic sigmoid activation function and 15 for the hyperbolic tangent activation function, and since that test comprised of testing each learning rate and then averaging them, now the learning rates can be directly examined.

First by using the logistic sigmoid function, data is collected for the various learning rates and depicted in a graph:

GRAPH V. LOG LEARNING RATE DATA



Graph V. Collected data comparing learning rates, log.

As before, Levene's test for homogeneity is performed and it is found that the f-ratio is 13.1 with a p-value < 0.00001, thus the null hypothesis is rejected, and the data is demonstrably homogenous. For each sample, a Shapiro Wilks test is performed with an average W = 62, therefore none of the samples are normally distributed.

Once homogeneity and non-normalcy are established, the Kruskal-Wallis test can be performed: the f statistic is found to be 13.09 with a p-value < 0.0001, well below the null hypothesis p < 0.01, and as before, the means and variances are found:

TABLE VII. MEAN AND VARIANCE OF LR TEST

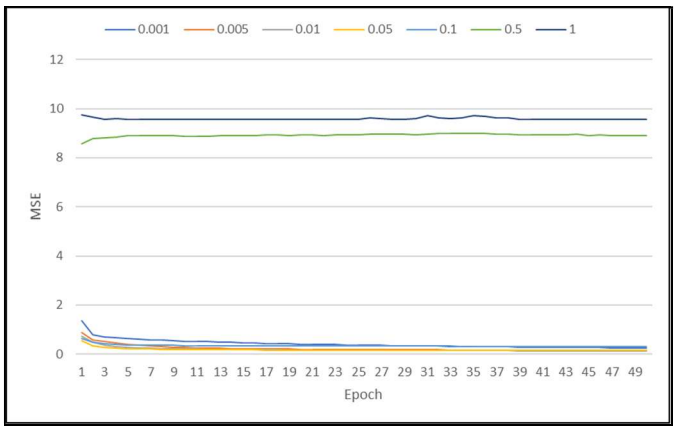
Learning Rate	Mean	Variance
0.001	0.8032	0.3235
0.005	0.4170	0.4741
0.010	0.2903	0.4592
0.050	0.1257	0.3541
0.100	0.0928	0.3047
0.500	0.0472	0.2172
1.000	0.0380	0.1949

Table VII. Mean and variance per learning rate, log.

Finding the outliers using the interquartile range 0.4170 – 0.0472 = 0.3698 yields four outliers: a learning rate of 0.001, 0.005, 0.5, and 1. Observing the remaining learning rates of 0.01, 0.05 and 0.1 through another Kruskal-Wallis test suggests that these three learning rates still have mean sample data which suggests variance. In this instance, the smallest mean is chosen as parameter: a learning rate of 0.5 for the logistic sigmoid function.

For the hyperbolic tangent activation function, the same tests as performed for the logistic function are done. This includes a preliminary test for homogeneity and normalcy followed by a Kruskall-Wallis test. First, the data is collected and can be visualized below:

GRAPH VI. TANH LEARNING RATE DATA



Graph VI. Collected data comparing learning rates, tanh.

Visually, the graph suggests that there is a little more leeway in the learning rate for this activation function, but it still has a functional maximum. This can be verified by using the Kruskall-Wallis test.

Using Levene's test for homogeneity, it is found that the f-ratio is 7.2 with a p-value < 0.00001, thus the null hypothesis is rejected, and the data demonstrated homogeneity. Using a Shapiro-Wilk test to test each sample, none of them exhibit normalcy and combined they have a mean W = 62.

Performing the Kruskall-Wallis test, the test statistic is exorbitantly high at 88,083 and each pair shows significant difference in means. This is with a p-value < 0.00001, thus there is demonstrable variance in means. To verify, the means and variances are tabulated:

TABLE VIII. MEAN AND VARIANCE OF LR TEST

Learning Rate	Mean	Variance
0.001	0.4209	0.4351
0.005	0.2336	0.3615
0.010	0.1908	0.3170
0.050	0.0641	0.2532
0.100	0.0514	0.2267
0.500	0.0657	0.2564
1.000	0.0483	0.2199

Table VIII. Mean and variance per learning rate, tanh.

Finding the dispersion of the means, a range of 0.2336 – 0.0514 = 0.1822 is found. This means a learning rate of 0.001, 0.500 and 1.000 are outliers and can be discarded. Comparing the remaining options using another Kruskall-Wallis test suggests there is still ample variance in the mean to decide. As with the logistic function, the smallest mean is chosen: a learning rate of 0.100.

Finding the correlation coefficient by using Pearson's test again, an r = 0.9926 is found, suggesting very strong correlation between learning rates between both activation functions. Compounding this, the p-value is found to be < 0.00001, suggesting extremely strong significance.

At this point, both the structure and learning rates of networks using either activation functions have been found:

TABLE IX. KNOWN PARAMETERS

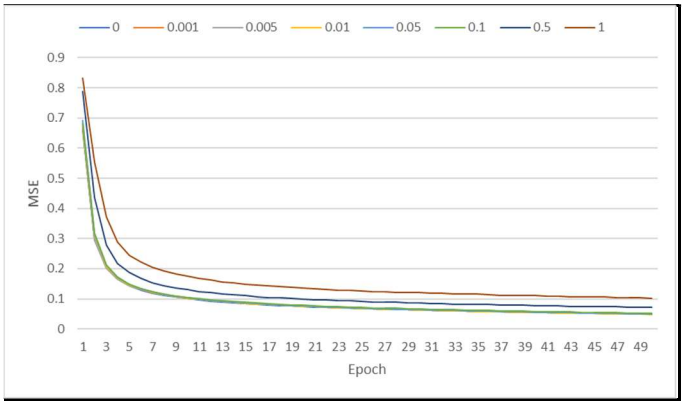
	Hidden Neurons	Learning Rate	Momentum Rate
log	20	0.500	?
tanh	15	0.100	?

Table IX. Idealized Parameters, log and tanh.

C. Momentum Rate

As the hidden layer structure and the learning rate are now known for both activation functions, the momentum rate can now be focused upon. As before data is collected and can be visualized in a graph:

GRAPH VII. LOG MOMENTUM RATE DATA



Graph VII. Collected data comparing momentum rates, log.

Analysis of this data using Levene's test suggests heterogeneity and there is a high equality of variance with a p-value = 0.993. Using the Shapiro-Wilks test, no samples are considered normal with a mean W = 45. While the samples are heterogenous and non-normal, the Kruskall-Wallis test cannot be performed and rather an ANOVA test is instead done.

Using the ANOVA test, an f-statistic of 3.12 is found, along with a p-value of 0.003, still below the null hypothesis of p < 0.05. This suggests there is still some variance between means in each sample, which are enumerated below:

TABLE X. MEAN AND VARIANCE OF MR TEST

Momentum Rate	Mean	Variance
0.000	0.0928	0.3047
0.001	0.0941	0.3071
0.005	0.0935	0.3068
0.010	0.0940	0.3094
0.050	0.0956	0.3129
0.100	0.0976	0.3101
0.500	0.1240	0.3372
1.000	0.1642	0.3513

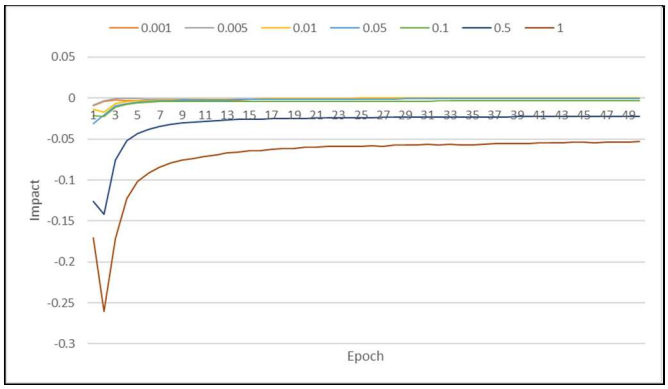
Table X. Mean and variance per momentum rate, log.

Finding the dispersion of the sample means, 0.1174 – 0.0936 = 0.0238, two obvious outliers are found: a momentum rate of 0.500 and 1.000. Once removed and the ANOVA test is again performed, the null hypothesis is accepted with an f-statistic of 0.016 with a p-value of 0.99. This demonstrates that there is very little variance within means of the samples and therefore affect

on performance is negligible if not non-existent. In other words, momentum had no positive impact on the system.

To understand how the outliers affected the system, however, the differences between momentum rates and no momentum can be illustrated:

GRAPH VIII. DIFFERENCES OF LOG MR



Graph VIII. Differences between momentum and none, log.

If not visually, the means of each can be tabulated and used to illustrate the impact on performance:

TABLE XI. MEAN DIFFERENCES OF LOG MR

Momentum Rate	Mean Impact
0.001	-0.00129
0.005	-0.00068
0.010	-0.00125
0.050	-0.00276
0.100	-0.00478
0.500	-0.03122
1.000	-0.07140

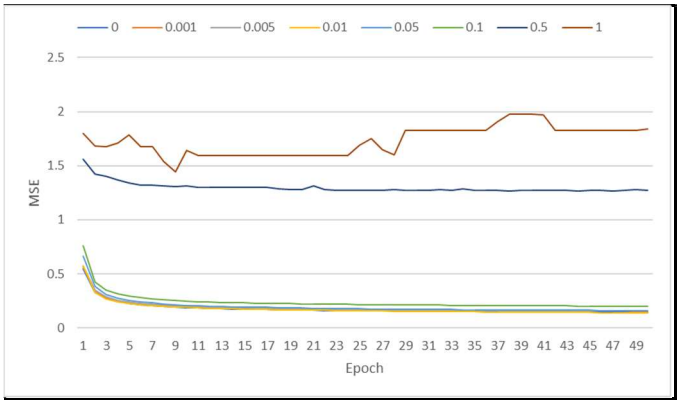
Table XI. Mean differences between momentum and none, log.

In this, the values themselves matter little, it is only the sign of the mean impact which matters in every instance, the impact is negative, suggesting adding momentum to this system hinders performance rather than improves it.

For the logistic sigmoid activation function, this system does not need momentum to perform adequately; contrarily, adding moment causes the system to perform worse. For this reason, a momentum rate of 0.000 is considered ideal.

Repeating the previous process for the hyperbolic tangent activation function, the data is collected:

GRAPH IX. TANH MOMENTUM RATE DATA



Graph IX. Collected data comparing momentum rates, tanh.

Using Levene's test does not show homogeneity as before with a p-value = 0.96 showing confidence in that regard. However, using Shapiro-Wilks, each sample is normal with a mean W = 47. As before, the ANOVA test is employed to analyze significance of mean variance.

The ANOVA test produces an f-statistic of 3029.062, abnormally large considering prior data tested, and has a p-value < 0.0001, suggesting there does exist variance in the means of each sample taken. Like before, the means are enumerated:

TABLE XII. MEAN AND VARIANCE OF MR TEST

Momentum Rate	Mean	Variance
0.000	0.1780	0.2532
0.001	0.1883	0.2556
0.005	0.1899	0.2575
0.010	0.1776	0.2601
0.050	0.2005	0.2816
0.100	0.2410	0.2922
0.500	1.2969	0.2244
1.000	1.7296	0.3635

Table XII. Mean and variance per momentum rate, tanh.

The dispersion range is found to be 1.033 – 0.181 = 0.852. Two outliers exist, like before, momentum rates of 0.500 and 1.000. Removing these outlier cases and redoing the ANOVA test brings the null hypothesis to fruition with an f-statistic of 0.96 and p-value = 0.43, above the threshold needed of $p < 0.05$. Therefore the means of each sample can be considered equal.

Not dissimilar to the logistic activation function, each possible momentum rate performs worse than no momentum rate. Therefore, as before, the idealized parameter for a system using the parameters thus far is 0.000.

Finally, all the parameters have been found as their ideal versions:

TABLE XIII. KNOWN PARAMETERS

	Hidden Neurons	Learning Rate	Momentum Rate
log	20	0.500	0.000
tanh	15	0.100	0.000

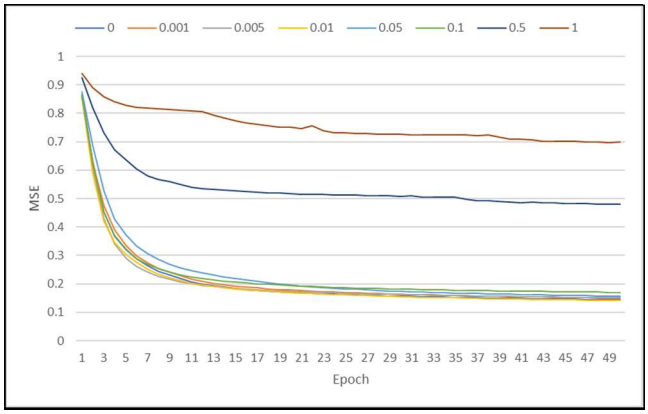
Table XIII. Idealized Parameters, log and tanh.

While anticlimactic, it is the successive retrieval of ideal parameters which contributes to the momentum rate being zero. Incidentally, the size of the hidden layer and the learning rate do not complement a momentum rate, which suggests the system as-is is able to leave local minima—if it even encounters any.

Curious, however, is this does not demonstrate the applicability of momentum rate. Were the other parameters different, momentum may have been applicable and a viable solution to improve system performance.

To take an example, consider a system which uses the logistic activation function and a learning rate of 0.500, but only 5 hidden neurons:

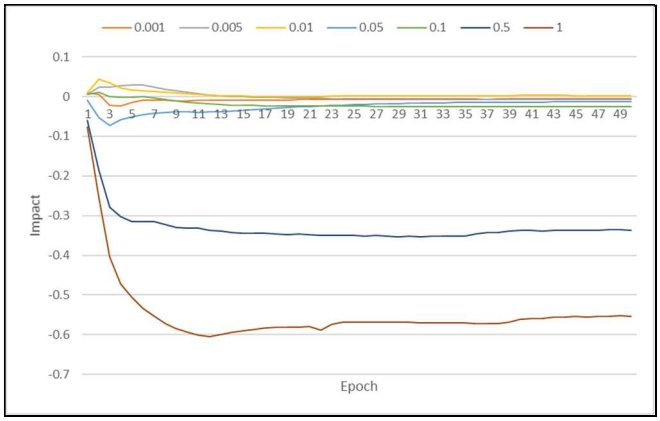
GRAPH X. LOG MOMENTUM RATE DATA



Graph X. Collected data comparing momentum rates, log.

This graph does not look much different from prior momentum rate graphs, however upon inspecting the differences of momentum rates versus none, a positive impact can be seen:

GRAPH XI. DIFFERENCES OF LOG MR



Graph XI. Differences between momentum and none, log

In the above graph, the mean difference between momentum rate choices illustrate that a small momentum rate is ideal for this system:

TABLE XIV. MEAN DIFFERENCES OF LOG MR

Momentum Rate	Mean Impact
0.001	-0.00761
0.005	+0.00051
0.010	+0.00513
0.050	-0.02549
0.100	-0.01991
0.500	-0.33053
1.000	-0.54843

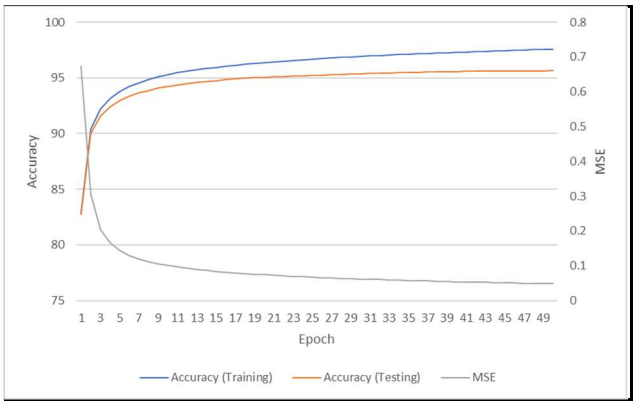
Table XIV. Mean differences between momentum and none, log.

While the impact on MSE may seem small, this corresponds to a 1-3% increase in accuracy between both training and testing data. To this end, momentum should be carefully used to fine tune a system rather than be a primary parameter. However, this is under the assumption that the problem space is similar.

D. Final System Performance

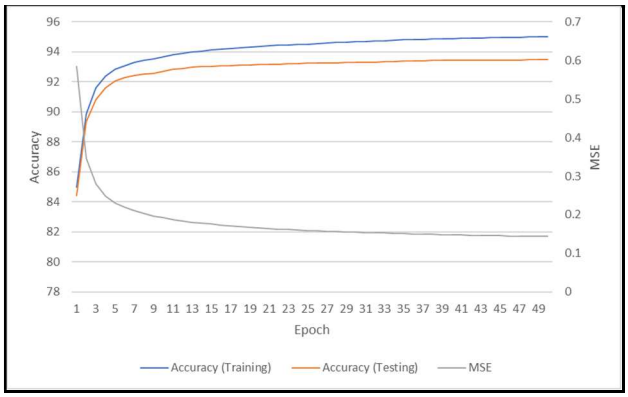
Since the idealized system parameters have been found for both activation functions, the final system performance can be gauged. Rather than 15 runs of training and testing, 30 are used to increase statistical relevance.

GRAPH XII. IDEALIZED PARAMETER DATA (LOG)



Graph XII. Collected data using idealized parameters, log.

GRAPH XIII. IDEALIZED PARAMETER DATA (TANH)



Graph XIII. Collected data using idealized parameters, tanh.

In both graphs, both the MSE and system accuracy is reported, with accuracy being split into training and testing data. A summary of findings is below:

TABLE XV. MAX, MIN, MEAN OF IDEAL PARAMETERS

		Accuracy (Training)	Accuracy (Testing)	MSE
log	Max	97.569%	94.596%	0.67
	Min	82.858%	82.776%	0.05
	Mean	95.976%	94.596%	0.09
tanh	Max	95.013%	93.476%	0.58
	Min	84.961%	84.429%	0.14
	Mean	94.055%	92.839%	0.18

Table XV. Summary of idealized parameters, log and tanh.

VIII. CONCLUSION (NEURAL NETWORKS)

Upon testing the structure of the neural network, a range upon which an adequate number of hidden neurons make a viable system were found. For the specific problem of MNIST Digit recognition, a minimum of 10 hidden neurons was necessary to achieve reasonable accuracy while a maximum of 35 hidden neurons was found to be the upper limit before diminishing returns are encountered. Below this 10-35 range leads to poor performance while above this range leads to artificially lengthened training and testing times.

Once a suitable hidden layer structure was found, the learning rate was then examined and was found to also have a variable range whereupon it makes a viable system. Using the logistic sigmoid activation function, a higher learning rate made the system learn quicker and often achieve higher accuracy, whereas with the hyperbolic tangent function, a higher learning rate, when in the extremes, lead to no learning and instead the system may as well have been guessing at random. However, both functions exhibited sensitivity in the lower ranges suggesting a lower learning rate should be used but should be fine-tuned empirically.

Finally, the momentum rate was examined and was found to have a negligible if not negative impact on system performance provided the other two parameters are ideal. However, upon testing momentum with a smaller network (where performance suffered as a cause of the smaller hidden layer), it was able to increase performance slightly but only when it was minuscule.

The ideal system was found to have a 64-20-10, LR 0.500, MR 0.00 structure using the logistic sigmoid activation function and a 64-15-10, LR 0.100, MR 0.000 structure using the hyperbolic tangent activation function.

While both functions performed well, the logistic sigmoid function was found to be a better activation function for this specific problem space. However, the network was smaller using the hyperbolic tangent function, instead, and thus would train marginally faster and be able to be tested marginally faster.

What can be gathered from these tests is that a given problem may need specific parameters and that perhaps for any system implementation, the parameters used may not be identical. It is important to discover parameters empirically while finding suitable ranges to analyze through literature or other experiments. For the problem space of the MNIST Digit data set, specific parameters were found but this may differ were the data used be different in any regard, such as how large the input layer is or how large the output layer is.

IX. FINAL REMARKS

Both decision trees and neural networks serve similar purposes: as a method of supervised learning, both are able to be used as classification algorithms which can learn from example data and classify novel data. The differences, however, are in precise application: a decision tree may be more appropriate for some problems than neural networks and vice-versa, as well as the problem size may play a factor in the decision to choose one over another.

Neural networks are relatively complex and are not human readable whereas decision trees, at a glance, can be broken down into a series of questions about data which can produce an answer. However, neural networks have shown to be more lenient of parameter choices and structure. Given the numerous permutations of hidden layer structures and learning and momentum rates, for the MNIST Digit problem space at least, it is relatively easy to find adequate parameters for a 95+% accuracy system.

This report demonstrates that many of the neural network parameters need to be found empirically through experimentation and there is no specific rule. There are too many factors at play which affect how the network structure should look that it is impossible to proclaim that one set of parameters categorically works for all instances.

REFERENCES

[1] "What is machine learning?," IBM. [Online]. Available: https://www.ibm.com/topics/machine-learning. [Accessed: 06-Mar-2020].

[2] T. M. Mitchell, Machine learning. New York: McGraw-Hill, 1997.

[3] "Decision Tree Learning," Brock University [Lecture]. B. Ombuki-Berman, 23-Jan-2020.

[4] J. R. Quinlan, "Improved Use of Continuous Attributes in C4.5," Journal of Artificial Intelligence Research, vol. 4, pp. 77–90, 1996.

[5] K. Hornik, "Approximation capabilities of multilayer feedforward networks," Neural Networks, vol. 4, no. 2, pp. 251–257, 1991.

[6] L. Liu, Y. Wu, W. Wei, W. Cao, S. Sahin, and Q. Zhang, "Benchmarking Deep Learning Frameworks: Design Considerations, Metrics and Beyond," in2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), July 2018, pp. 1258–1269.

[7] N. Qian, "On the momentum term in gradient descent learning algorithms," Neural Networks, vol. 12, no. 1, pp.145–151, 1999. [Online].

[8] K. Mehrotra, et al, Elements of Artificial Neural Networks

[9] "D. Kriesel," http://www.dkriesel.com. [Online]. Available: http://www.dkriesel.com/en/science/neural_networks. [Accessed: 01-Mar-2020].

[10] S. Ruder, "An overview of gradient descent optimization algorithms,"arXiv e-prints, p. arXiv:1609.04747, Sep 2016.

[11] Patrick K. Simpson. 1989. Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations. Elsevier Science Inc., USA.