5-2017

A Study of Activation Functions for Neural Networks

Meenakshi Manavazhahan
University of Arkansas, Fayetteville

A Study of Activation Functions for Neural Networks

An undergraduate thesis

in partial fulfillment of the honors program at

University of Arkansas

College of Engineering

Department of Computer Science and Computer Engineering

by: Meena Mana

14 March, 2017

Abstract:

Artificial neural networks are function-approximating models that can improve themselves with experience. In order to work effectively, they rely on a nonlinearity, or activation function, to transform the values between each layer. One question that remains unanswered is, "Which non-linearity is optimal for learning with a particular dataset?" This thesis seeks to answer this question with the MNIST dataset, a popular dataset of handwritten digits, and vowel dataset, a dataset of vowel sounds. In order to answer this question effectively, it must simultaneously determine near-optimal values for several other meta-parameters, including the network topology, the optimization algorithm, and the number of training epochs necessary for the model to converge to good results.

Intro:

Machine learning is starting to enable a multitude of useful applications. A few things that one could do with machine learning are image and sign recognition, predicting an individual's future shopping trends, and diagnose medical conditions. A neural network is one model of machine learning and has recently been found as being especially good at image recognition. We specifically study activation functions within the neural network model that falls under machine learning.

On a similar note, patterns of digits can be trained by computers to be accurately categorized and sorted. Machines could help recognize the digits, proving useful in many fields, such as sorting mail. Postal addresses and zipcodes are often difficult to read, but with the help of optical character recognition (OCR), they may be easier to decipher and then sort. One can picture the effects that sorting mail by computers would have in today's world…the hope is that the letters would be delivered to the receiver in a more perfect, satisfying fashion.

The MNIST(Mixed National Institude of Standards and Technology) dataset, widely used in the field of machine learning, consists of 70,000 handwritten digits, consisting of 60,000 training images and 10,000 test images. The 60,000 training patterns are inputted into the training model as features, and 10,000 patterns are outputted from the testing model as labels. The models used to train and test labels and features are neural networks. These neural networks consist of equations that are tuned by weights. We want the neural networks that are trained and tested on the MNIST dataset to learn well, so that they are able to accurately recognize patterns and be tuned to give almost perfect results. In this way, the MNIST dataset could be used to accurately sort postal mail.

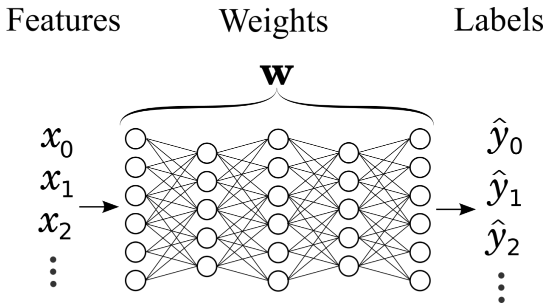
The Vowel dataset contains automatically extracted audio features for recordings from 11 different speakers making a variety of vowel sounds. The features are of MFCC's (Mel Frequency Cepstral Coefficient), as well as certain information about the person making the recordings.

Artificial neural networks are function approximating models. They accept a vector of input values, **x**, and compute a corresponding vector of output values, **y**:

$$\mathbf{y} = f(\mathbf{x}; \mathbf{w})$$

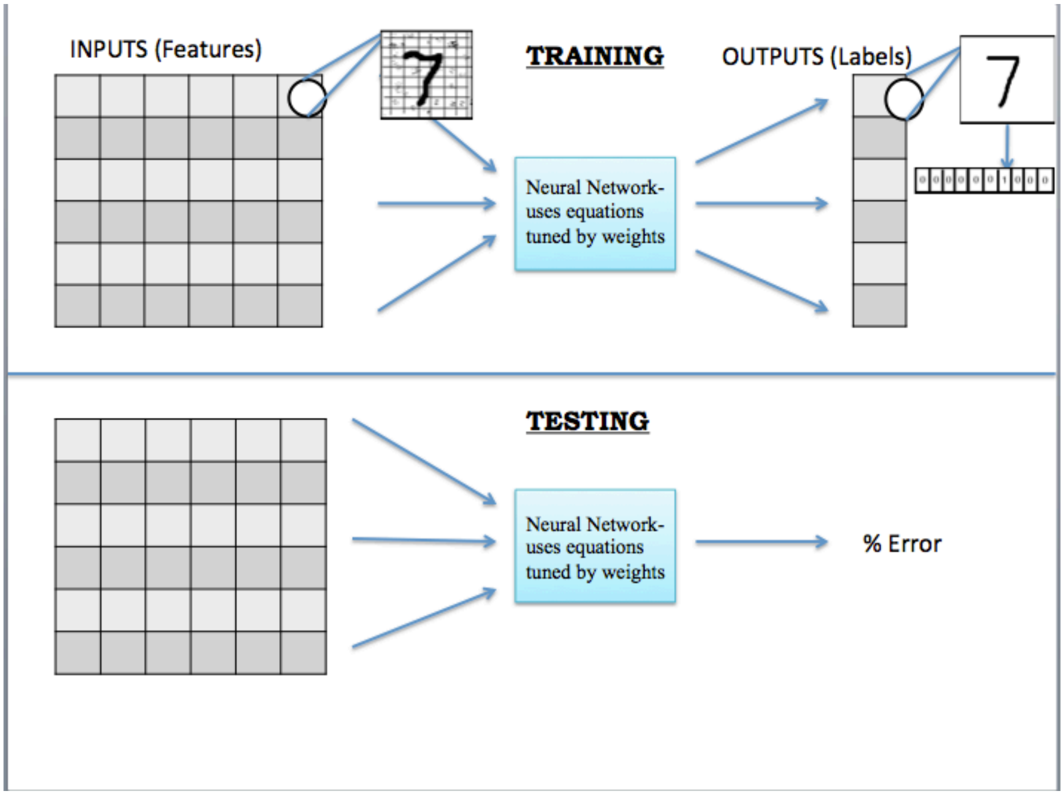
Neural networks rely on an internal set of weights, **w**, that control the function that the neural network represents. The process of adjusting the weights in a neural network to make it approximate a particular function is called “training”.

One of the more common types of neural networks are feed-forward neural networks. In these neural networks, the weights are organized into “layers” that feed into each other as illustrated in the following diagram:



In order to compute interesting functions, a non-linearity, also called an “activation function” or “transfer function” is typically inserted between each layer in the neural network. The activation function significantly increases the power of multi-layered neural networks, enabling them to compute arbitrary functions [3].

The below picture gives a rough idea of how neural networks work.


INPUTS (Features)
7
TRAINING
OUTPUTS (Labels)
7
0 0 0 0 0 0 1 0 0 0
Neural Network- uses equations tuned by weights
TESTING
Neural Network- uses equations tuned by weights
% Error

- 60,000 samples are inputted as features which are trained by the neural network and outputted as 60,000 labels.
- The 60,000 training input sample images have 784 pixels, therefore they are a size of vector 784.
- The 60,000 training output images have 10 encodings each.
- There are 10,000 testing images which are inputted as features, and our experiments give the final percent error, or misclassifications, out of 10,000 test patterns.

Although many activation functions have been studied, identifying the best activation function for a particular task still remains an open question. This thesis seeks to make progress toward understanding which activation functions are the best by systematically testing a collection of activation functions with the MNIST dataset and vowel dataset.

Of course, testing activation functions under sub-optimal conditions would not be very meaningful. Therefore, in order to identify the best activation function, we propose to sweep across a range of values for several other meta-parameters as well. These meta-parameters include the number of artificial neurons in each layer, the optimization method used to train the weights, and the number of training epochs performed to optimize the weights. By testing a full set of combinations of values in these meta-parameters, we seek to find the best activation function to use with this dataset under the best possible conditions of other parameters.

Ultimately, we found the best results for the MNIST dataset using the Leaky Rectifier non-linearity when the neural network contained 316 nodes in each layer and was trained with stochastic gradient descent at 75 training epochs. Our best model achieved a misclassification rate of 1.29%. For the vowel dataset, we found the best results using Gaussian non-linearity when the neural network contained 316 nodes in each layer and was trained with stochastic gradient descent at 154 training epochs. Our best model achieved a misclassification rate of 22.0779%.

Related Works:

Single-layer regression models were studied as long ago as the late 1700's by Gauss and Legendre [1]. Unfortunately, these models lacked the power to approximate complex functions. In the 1980's, with the popularization of backpropagation, multi-layered neural networks began to gain wide acceptance [2]. In the late 1980's, Cybenko proved that a neural network with two layers of weights and just one layer of a non-linear activation function formed a model that could approximate any function with arbitrary precision [3]. The non-linearity is critical for the power of neural networks because without it, the other layers of weights reduce to being equivalent to just a single layer.

In 2006, Hinton presented a method for efficiently training deep neural networks, or neural networks with many layers [4]. This started a resurgence of interest in neural networks that has grown to become the buzzword that is now "deep learning" [5]. These days, neural networks are starting to outperform even humans at complex learning tasks, such as image recognition [6].

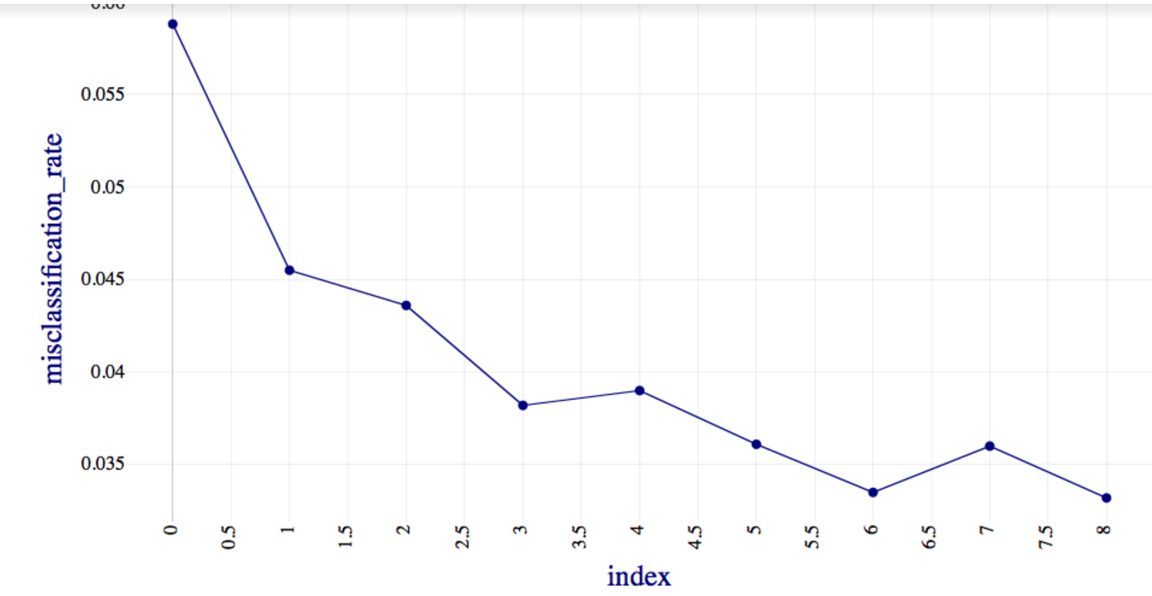
One of the most common datasets for testing deep neural networks is the MNIST (Mixed National Institute of Standards and Technology) dataset. This dataset trains a neural network to perform optical character recognition (OCR). This is a significant task for machine learning because vision has long been considered a task that was difficult for machines [7]. It has 70,000 handwritten digits, consisting of 60,000 training images and 10,000 test images. The 60,000 training patterns are typically used to train the model, and the remaining 10,000 patterns are used to evaluate the trained model. A model that can accurately classify patterns in the MNIST dataset could be used to sort postal mail, for example. However, in machine learning, it is often used simply as a test problem for evaluating a model's ability to learn.

For the experiments in this paper, we used the Waffles machine learning toolkit [8]. This toolkit provides many machine learning algorithms. For our study, however, we restricted our experiments to artificial neural networks. Significantly, this toolkit provides eleven non-linear activation functions, which we tested in our experiments.

Procedure- Technical Approach:

We first installed the latest version of Waffles, available on GitHub [9]. We also obtained a copy of the MNIST dataset in ARFF format [10]. We compiled the code and ran it successfully using these .arff files. Below is the display from the Terminal window:

In preliminary testing with the neural network from the Waffles toolkit, without any parameter tuning, we obtained the results shown below:



In this plot, the horizontal axis represents epochs, or training passes. The vertical axis represents model error. Out of 10000 labels outputted, less than 350 digits were incorrectly recognized, resulting in an approximately 0.034 (3.4%) error, or 97.6% accuracy rate. At epochs 4 and 7, the error rate increased, meaning it got temporarily worse, but overall, the error rate decreased. The model improved in accuracy as it gained experience.

In machine learning, we typically consider accuracy to be more important than training time, because training only needs to be done once. Training can be conducted as many times as needed.

Several factors affect the shape of the curve above. The factors that we focused on are activation functions (coded in non-linearity blocks), topology (coded in layer size), time (measured in epochs), and datasets (MNIST, particularly, and vowel as a stretch goal).

This paper sweeps through the following metaparameters:

• Datasets	MNIST and Vowel
• Topology	5 layer widths of 10, 31, 100, 316, and 1000.
• Activation Function	11 non-linear blocks
• Optimizer	1 optimizers
• Time	200 training epochs

We wrote an automated test suite that ran for 68 days to gather our results. The results led to the conclusion that Leaky Rectifier activation function with a topology of 316 and SGD Optimizer are the most optimal parameters in the MNIST dataset. In the Vowel dataset, the best optimizer is SGD Optimizer with a layer width of 316 using Gaussian nonlinear block. This paper describes the process of arriving at the two conclusions stated.

The datasets used are MNIST and Vowel.

The 11 non-linear blocks are listed below:

Tanh

Sine

Bent Identity

Gaussian

Identity

Leaky Rectifier

Logistic

Rectifier

SoftExp

SoftPlus

SoftRoot

The optimizer used for training and testing are SGD Optimizer.

Each training and testing period for the neural network takes 200 epochs.

Evidence

We wrote code to produce output data that gives the optimal nonlinear block with the best activation and best optimizer. We did experiments with layers of size 10, 31, 100, 316, and 1000.

The results from the first experiment were split into 57 trial runs, each with its own metadata header detailing the below:

a. dataset

b. nonlinearity block

c. training patterns

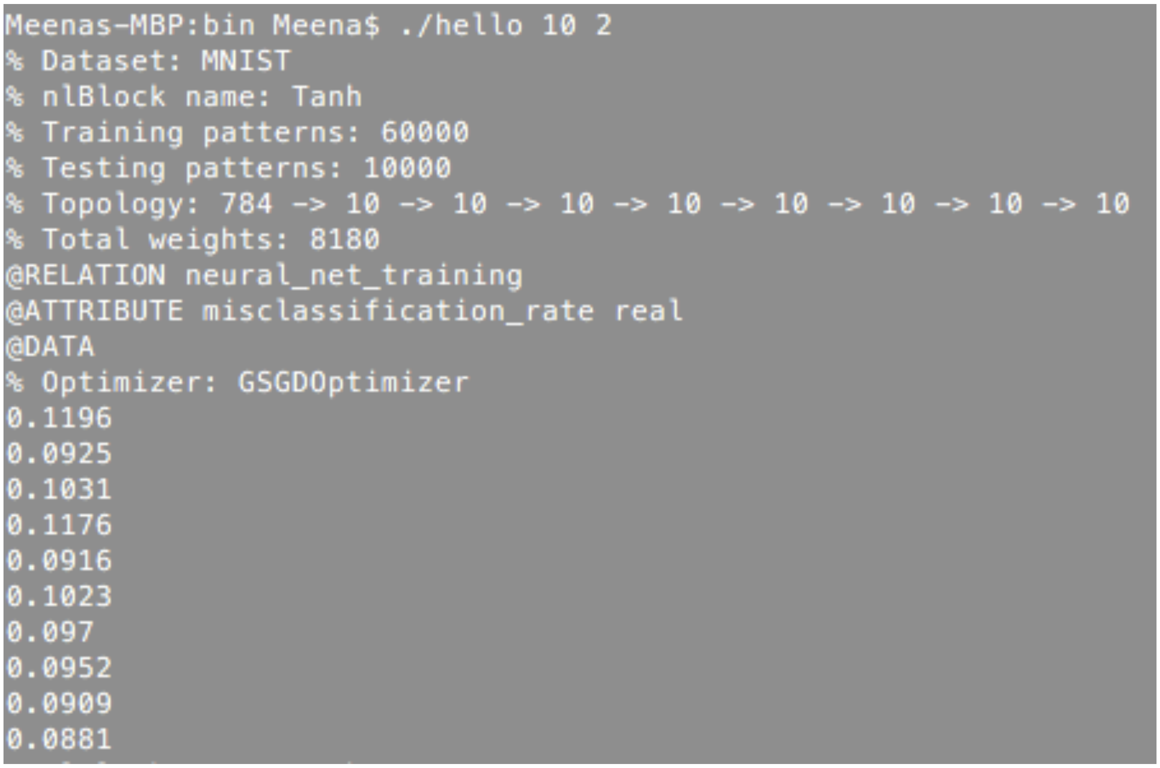
d. test patterns

e. topology with the layer size

f. total weights used to tune the network

g. optimizer used.

Screenshot:



```
Meenas-MBP:bin Meena$ ./hello 10 2
% Dataset: MNIST
% nlBlock name: Tanh
% Training patterns: 60000
% Testing patterns: 10000
% Topology: 784 -> 10 -> 10 -> 10 -> 10 -> 10 -> 10 -> 10 -> 10
% Total weights: 8180
@RELATION neural_net_training
@ATTRIBUTE misclassification_rate real
@DATA
% Optimizer: GSGDOptimizer
0.1196
0.0925
0.1031
0.1176
0.0916
0.1023
0.097
0.0952
0.0909
0.0881
```

After printing the output data in console, it needed to be plotted effectively. We chose Excel to graph the charts containing the output data.

We realized that plotting the results with the percent error would be useless without quantifiable metrics to analyze them with. Therefore, we chose to focus on finding the best nonlinearity block, which optimizer it used, with which topology, and at what epoch number (measuring time). Before identifying the best nonlinearity block, we decided to identify the best optimizer and topology.

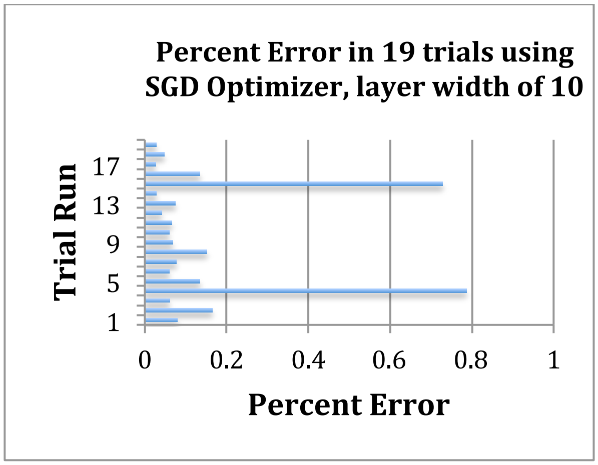
MNIST Dataset

Best Optimizer

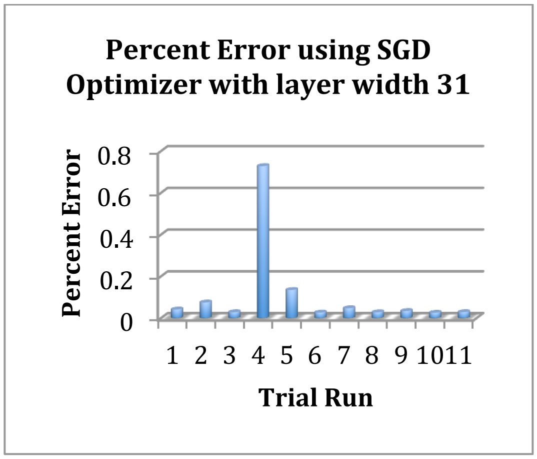
I. Layer Width: 10

We first decided to analyze the results using a layer width of 10 and the 3 different optimizers. We took the results from the 57 trial runs, and wrote a program to print out the percent error using each optimizer.

Our experiments were performed using the stochastic gradient descent optimizer. As shown in the following graphs, it achieved an error rate of less than 1% in the majority of cases.

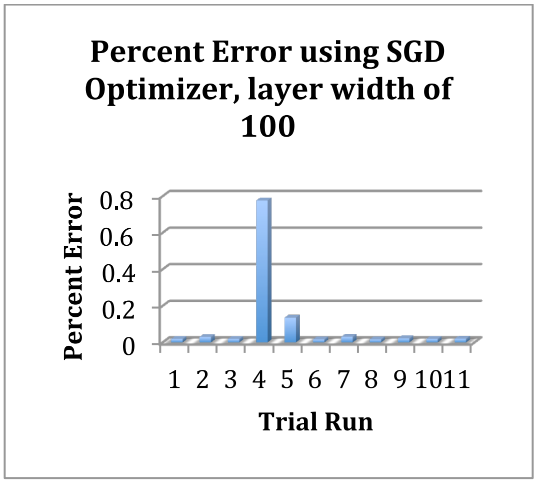


II. Layer Width: 31



The percent error rates for SGD Optimizer using a layer width of 31 are graphed above, showing a consistent low error rate.

III. Layer Width: 100



We narrowed down that SGD Optimizer had a 1.29% percent error at layer width of 316. After finding the lowest percent error rates among all of the layer widths (10, 31, 100, 316, and 1000), we found that SGD Optimizer was the best optimizer in these experiments when training with a layer width of 316.

Vowel Dataset: Best Optimizer

SGD Optimizer seemed to be the best optimizer for the vowel dataset.

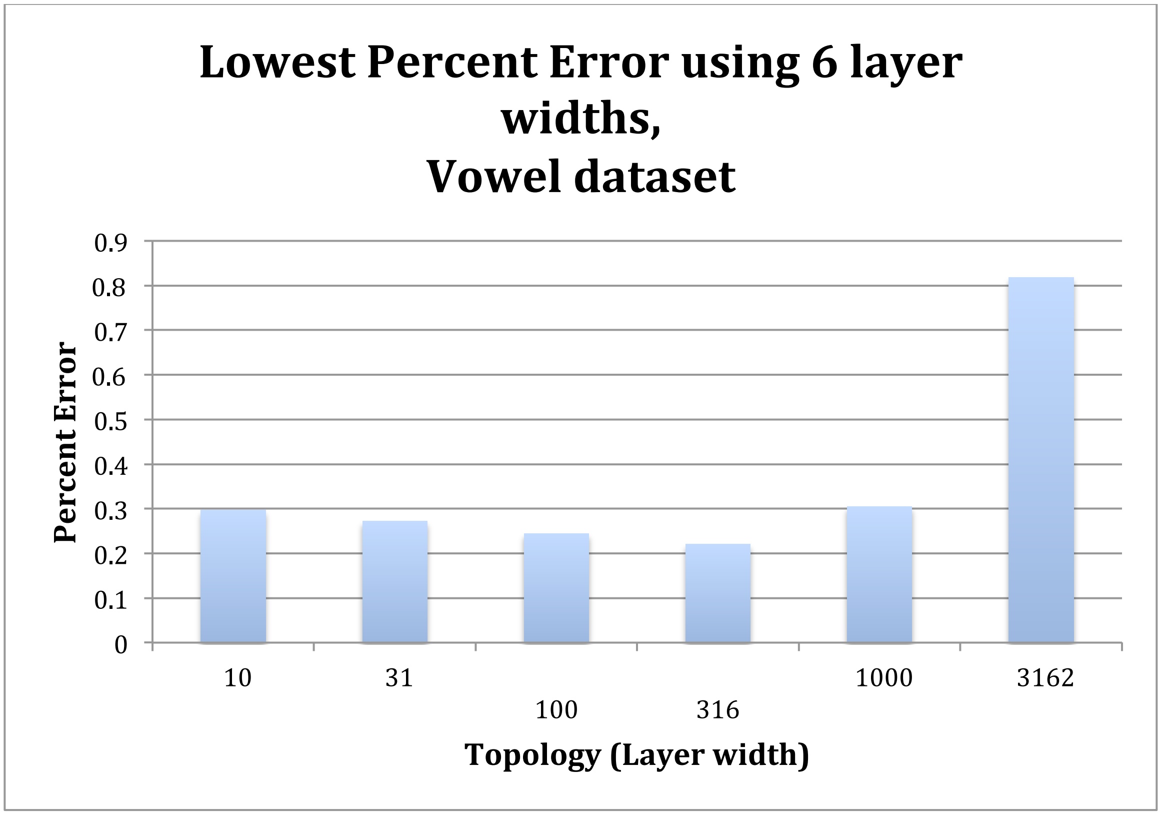
Calculating the lowest percent error for each layer width (10, 31, 100, 316, and 1000), SGD Optimizer gave the lowest percent error, 22.08% error at epoch 154.

MNIST: Best Topology (layer width)

Based on the above graphs calculated to find the best optimizer, the best topology was a layer width of 316, which gives 1.29% error.

Vowel: Best Topology (layer width)

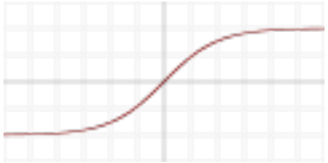
Based on the above graphs calculated to find the best optimizer, the best topology was a layer width of 316, which gives 22.0779% error. The below graph illustrates this result.



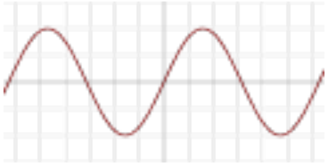
After extensive data analysis, the lowest error was found for each of the 6 layer widths- ranging from 10 to 3162. The optimal layer width for the vowel dataset was 316, giving the lowest overall percent error of 22.08%. The second best layer width was 100, with a percent error of 24.46%. A layer width of 3162 was the worst out of the 6, giving a high percent error of 81.81%.

Before analyzing the results of our experiments, we would like to present the pictures of each of the activation functions and what they look like on a graph [12].

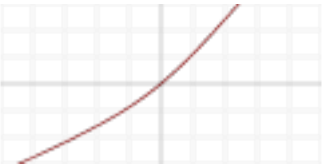
Tanh



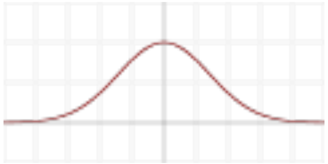
Sinusoid



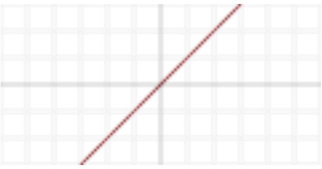
Bent Identity



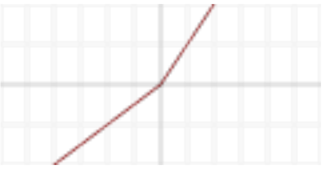
Gaussian



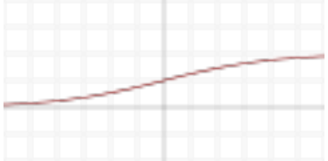
Identity



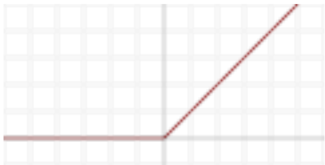
Leaky Rectifier (Rectified Linear)



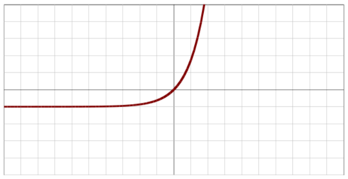
Logistic



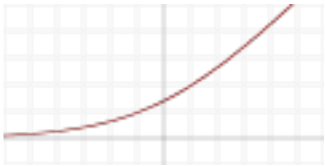
Rectifier



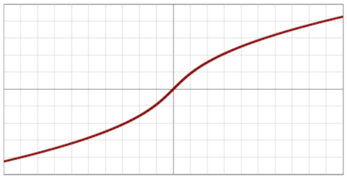
SoftExp



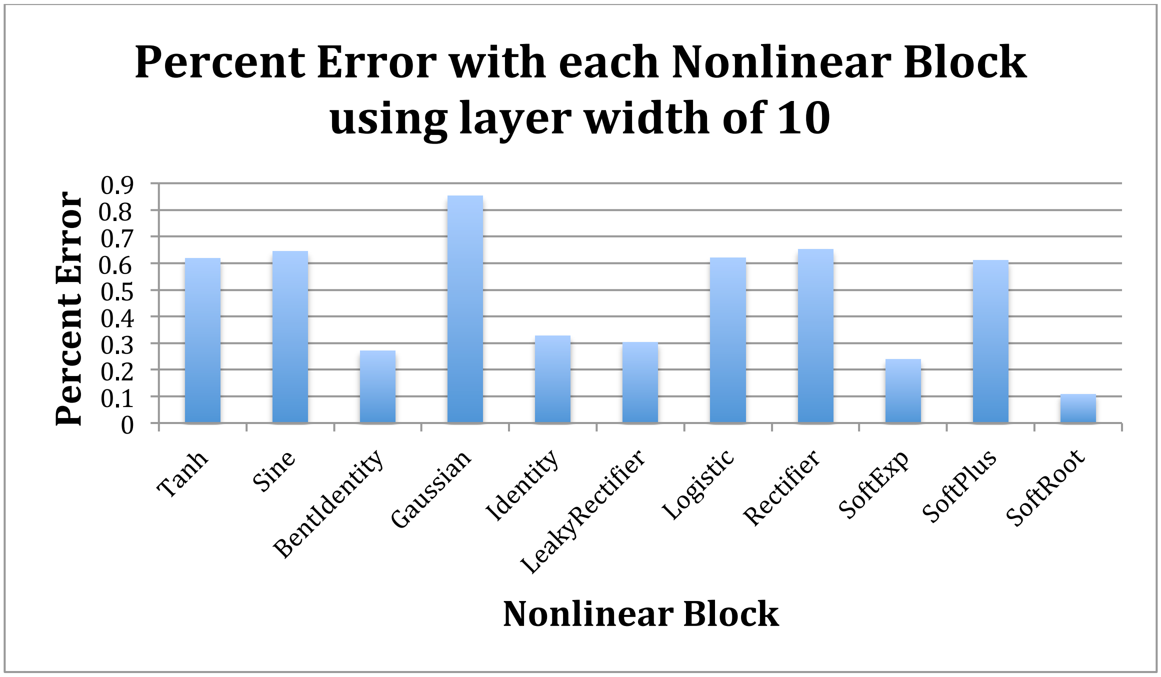
SoftPlus



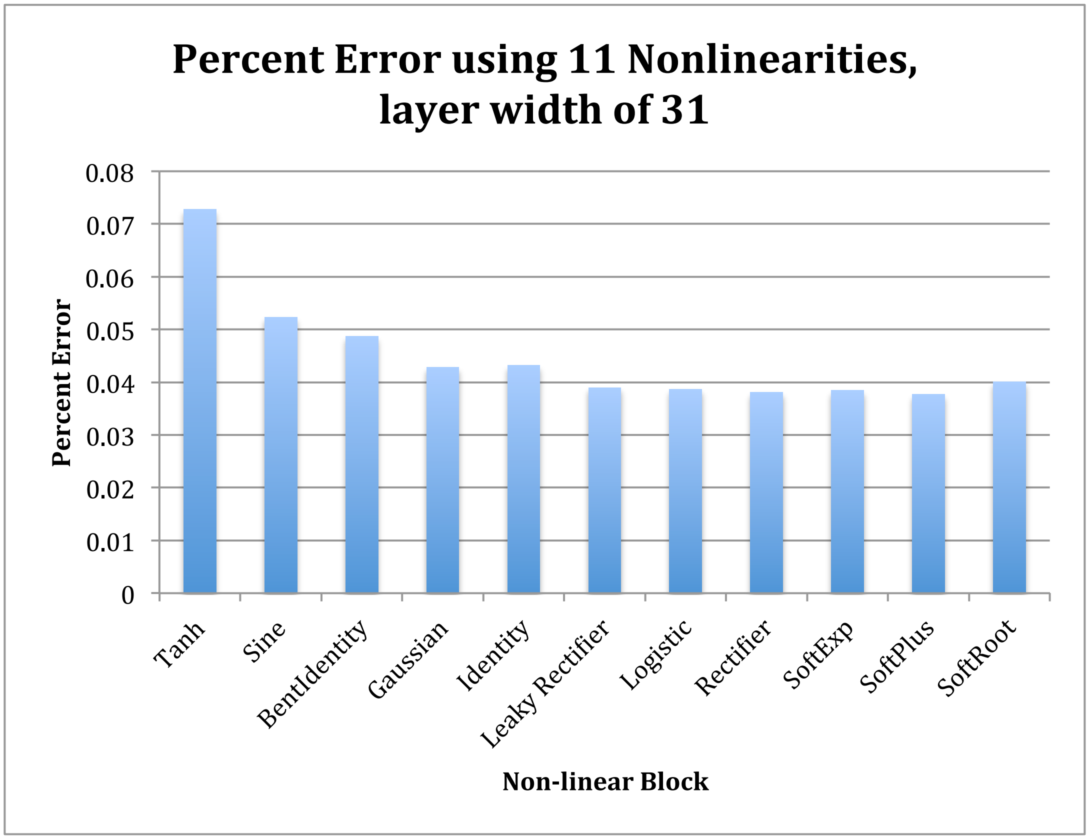
SoftRoot



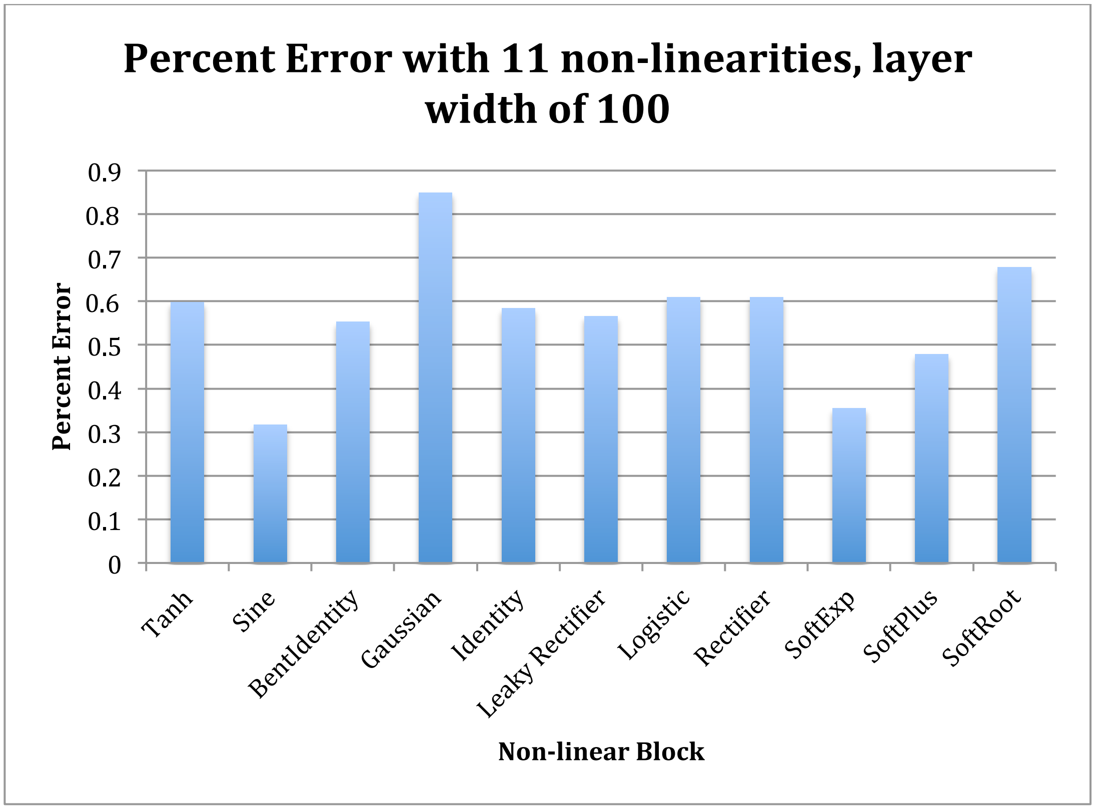
We analyzed the results of the experiments, specifically, the non-linear block. The below graph shows training and testing using a layer width of 10. Initially, it appeared that SoftRoot is the best nonlinear block, with a 10.88% error…



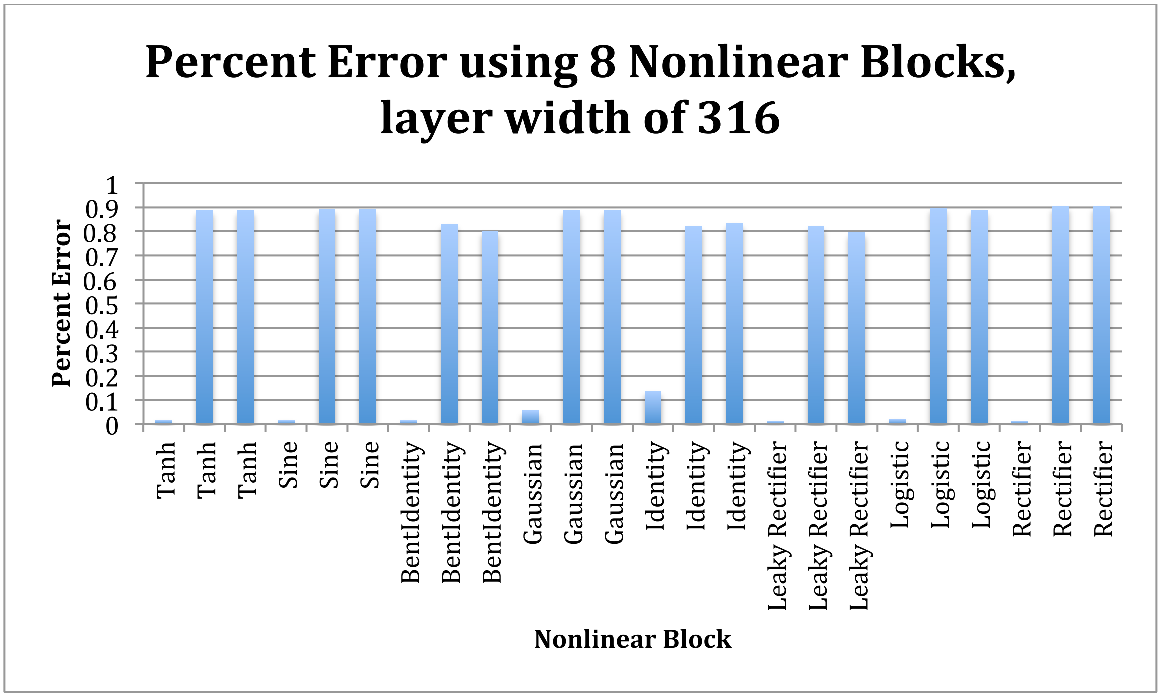
To test the initial hypothesis, we went further and plotted percent errors with deeper layers of 31, 100, 316, and 1000. error.

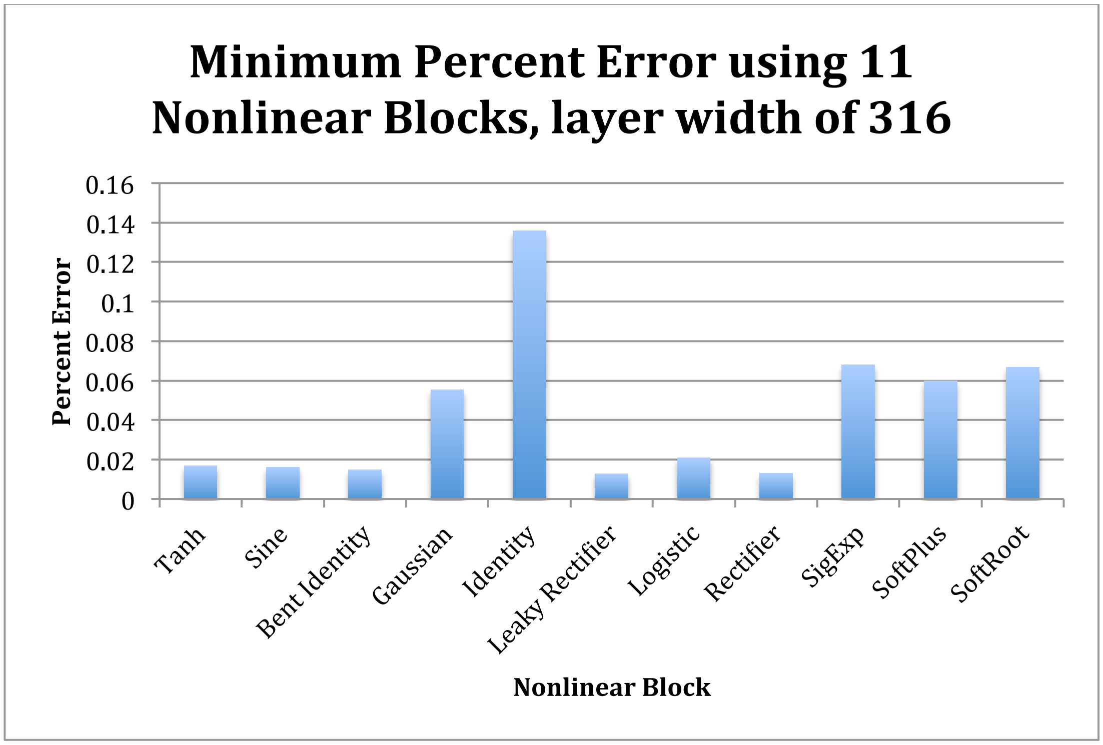


Plotting the average percent error with 11 non-linearities and a layer width of 31, SoftPlus proved to be the best nonlinearity, giving a percent error of 3.77%. Interestingly, tanh proved to be the worst non-linearity block, with a percent error of 7.28%. One noteworthy point is that though it appears that the differences in percent error are great, the range between the best percent error and worst percent error is 3.51%, which is relatively small.



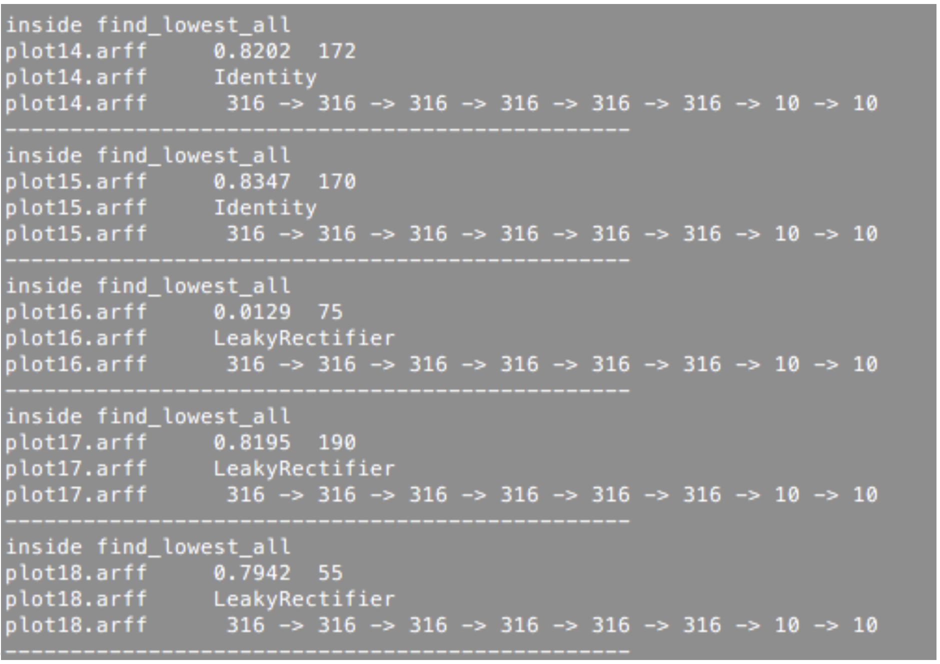
Using a layer width of 100, the best activation function proved to be sine, with a percent error of 31.66%. The worst activation function was Gaussian with a percent error of 84.97%. The other activation functions ranged evenly, approximately 55.00%, between the two.





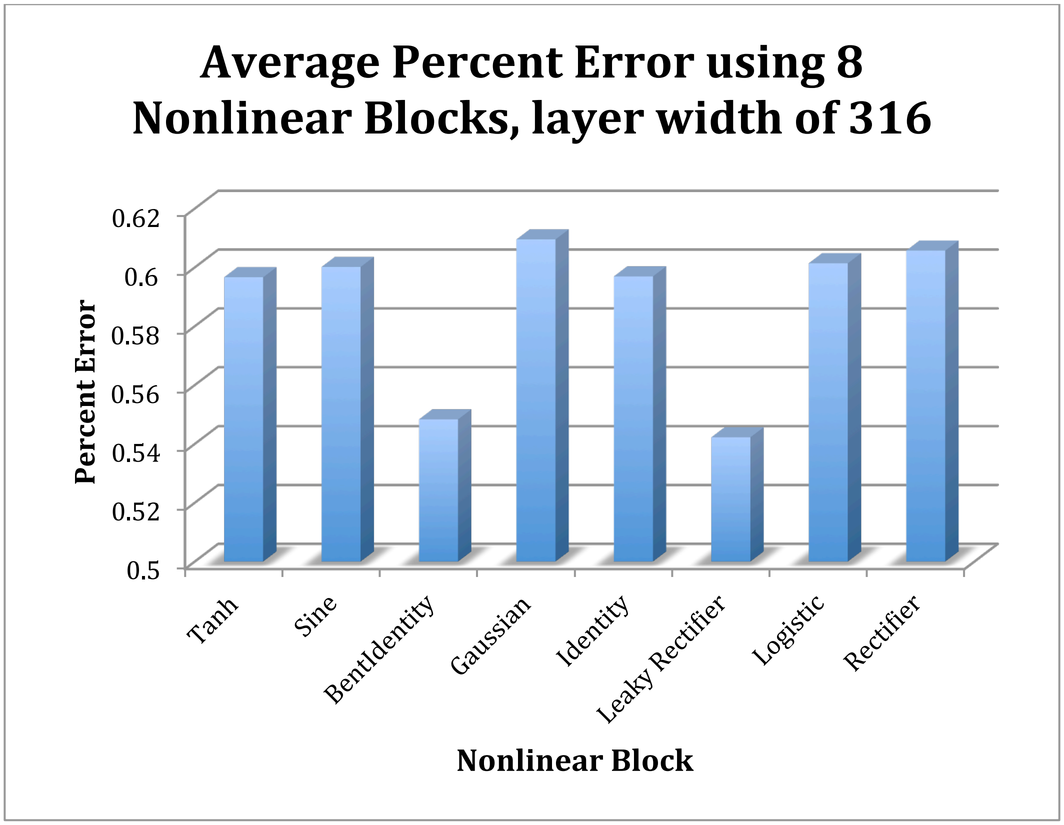
Plotting the results with a layer width of 316, I found that Leaky Rectifier appeared to be the best nonlinear block, with a percent error of 1.29%. It clearly beat the other activation functions tested. Though it is hard to tell, Rectifier was the worst activation function using a layer width of 316, with a percent error of 90.20%.

Below is the screenshot of the results:

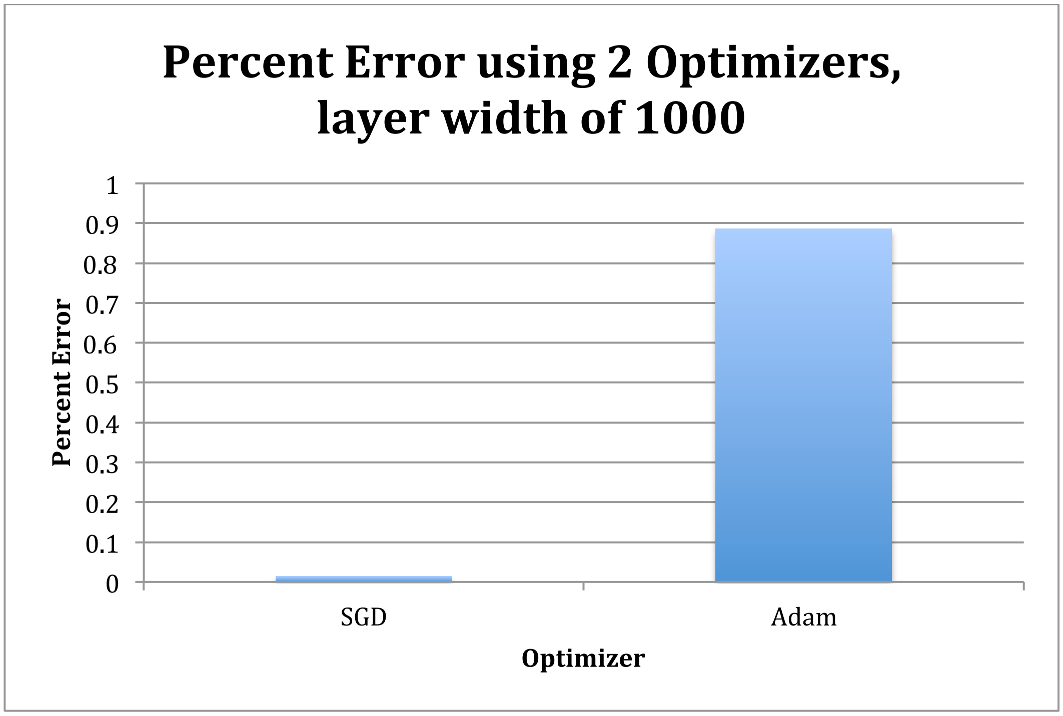


```
inside find_lowest_all
plot14.arff     0.8202  172
plot14.arff     Identity
plot14.arff      316 -> 316 -> 316 -> 316 -> 316 -> 316 -> 10 -> 10
-----------------------------------------------
inside find_lowest_all
plot15.arff     0.8347  170
plot15.arff     Identity
plot15.arff      316 -> 316 -> 316 -> 316 -> 316 -> 316 -> 10 -> 10
-----------------------------------------------
inside find_lowest_all
plot16.arff     0.0129  75
plot16.arff     LeakyRectifier
plot16.arff      316 -> 316 -> 316 -> 316 -> 316 -> 316 -> 10 -> 10
-----------------------------------------------
inside find_lowest_all
plot17.arff     0.8195  190
plot17.arff     LeakyRectifier
plot17.arff      316 -> 316 -> 316 -> 316 -> 316 -> 316 -> 10 -> 10
-----------------------------------------------
inside find_lowest_all
plot18.arff     0.7942  55
plot18.arff     LeakyRectifier
plot18.arff      316 -> 316 -> 316 -> 316 -> 316 -> 316 -> 10 -> 10
-----------------------------------------------
```

4



Further analyzing the results with layer width of 316, I found that Leaky Rectifier proves to be the best activation function, giving 54.22% average percent error. Bent Identity came in as a close second, with an average percent error of 54.83%.



I found that using a layer width of 1000, tanh is the best nonlinear block, with a 1.48% error. However, this considered results with only tanh block, because the experiment takes a considerable amount of time to run through 200 epochs of each nonlinearity block. Given that the experiment ran for about a month, it had time to train and test samples using SGD Optimizer and Adam Optimizer. The program did not yet print results for RMS Prop Optimizer.

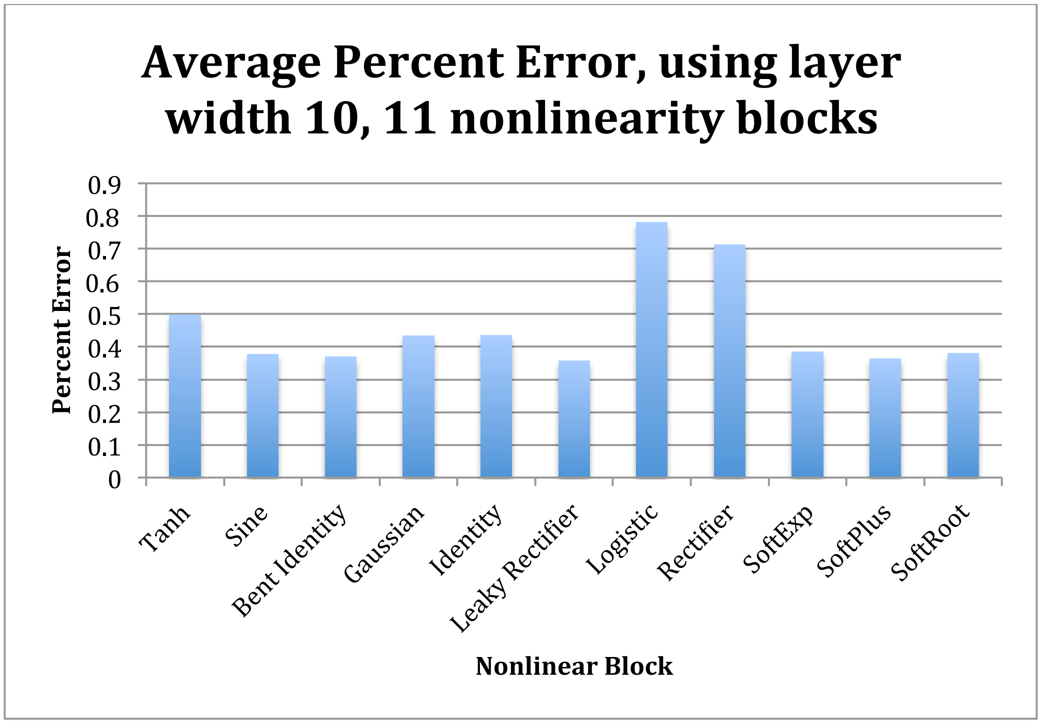
Layer Width	Best Activation Function	Percent Error
10	SoftRoot	10.88%

31	SoftPlus	3.77%
316	Leaky Rectifier	1.29%
1000	Tanh	1.48%

Clearly, in the MNIST dataset, using the neural network of layer width 316, Leaky Rectifier block, and SGD Optimizer gives the lowest percent error at 1.29%.

Vowel Dataset: Best Nonlinear Block

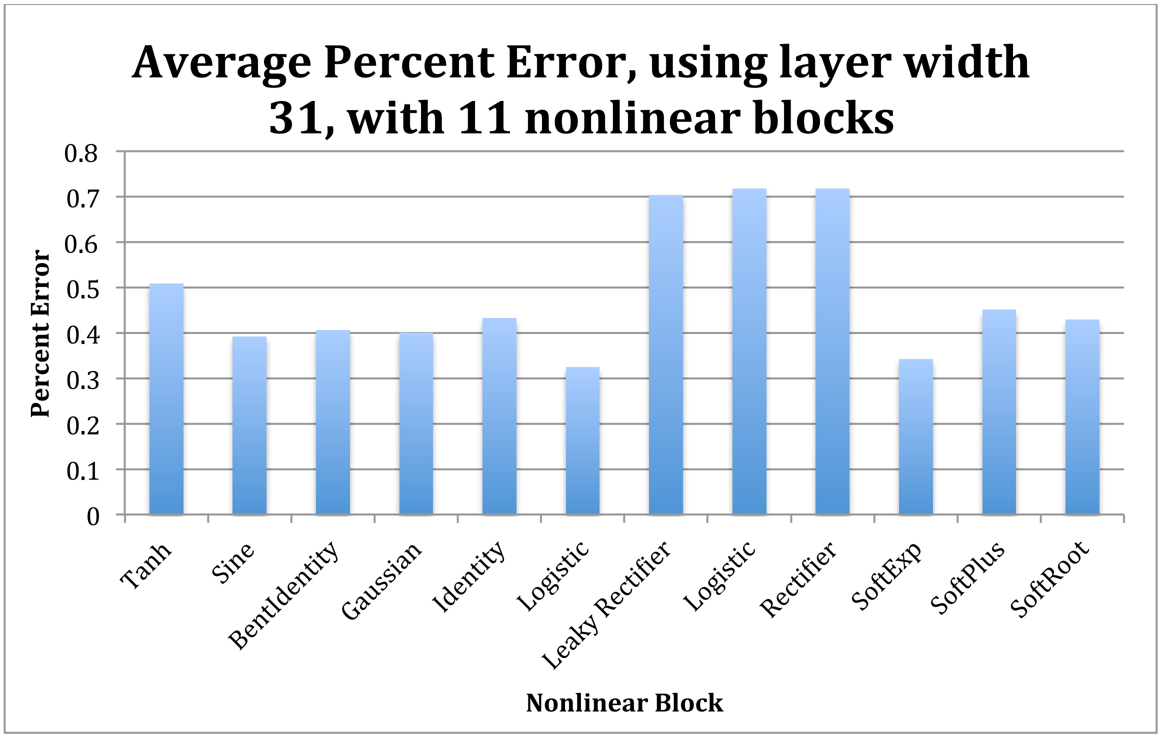
Layer Width: 10



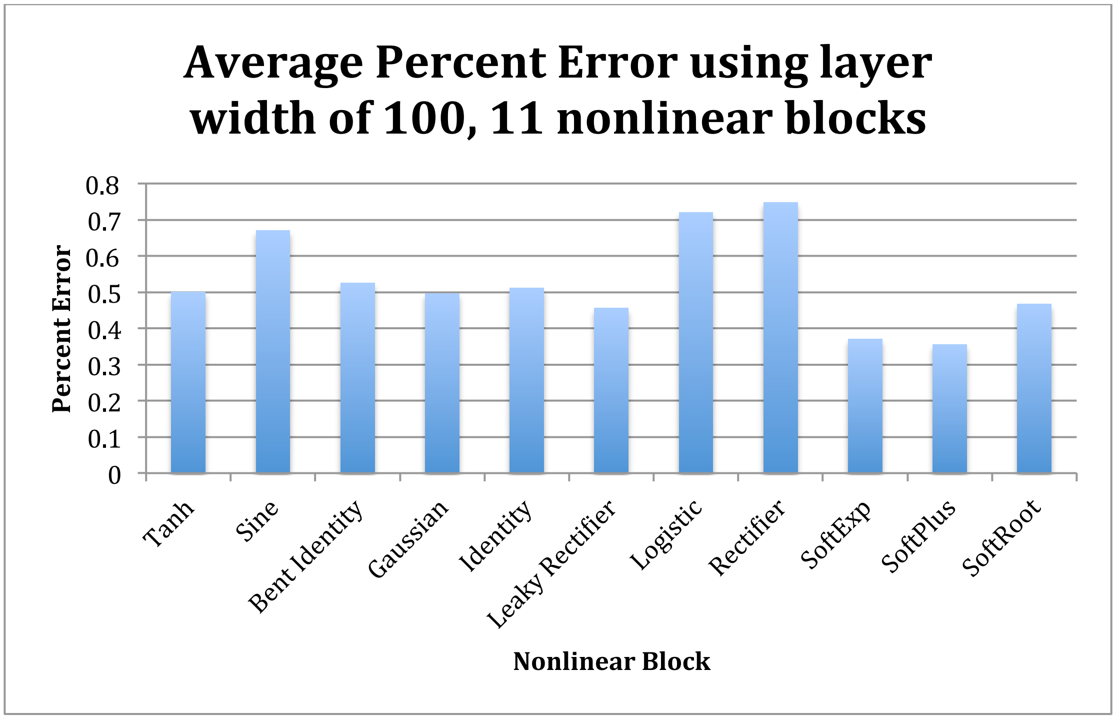
Plotting the 11 nonlinear blocks with a layer width of 10, we found that Leaky Rectifier provided the best percent error, with 35.71% error.

We decided to plot the nonlinear blocks' percent error, similary to MNIST dataset, in each of the layer widths. Below are the results.

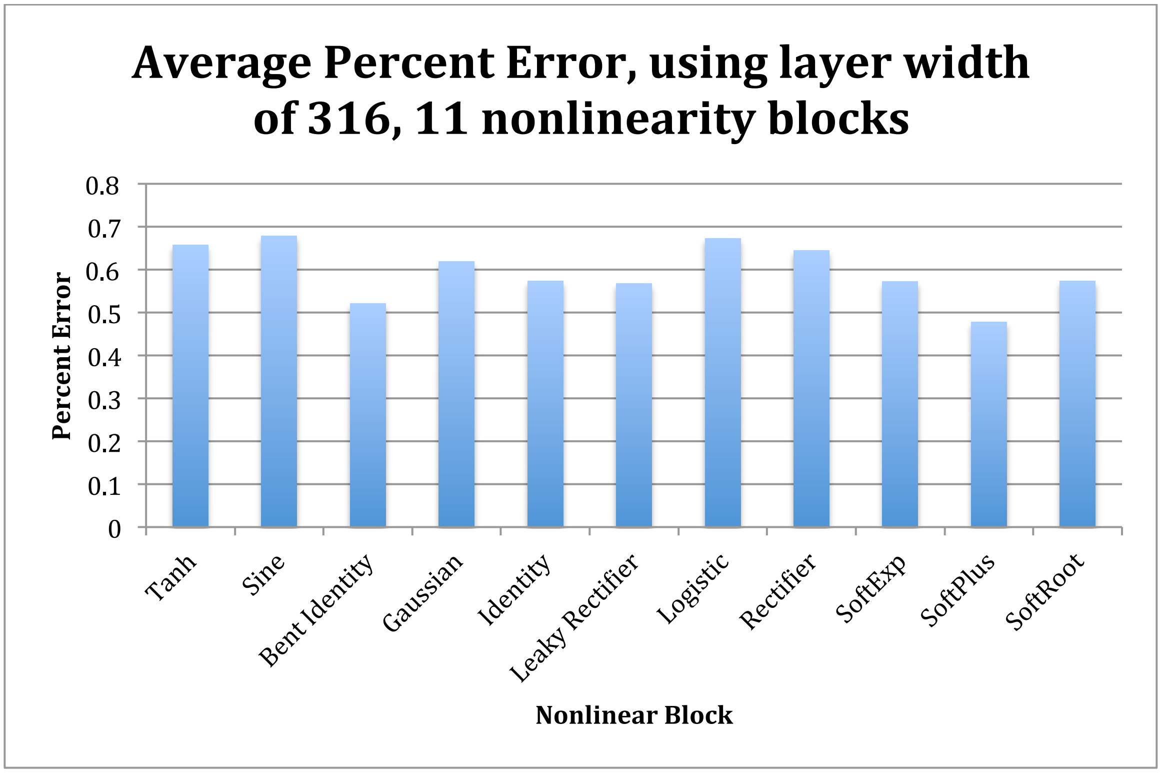
Layer Width- 31



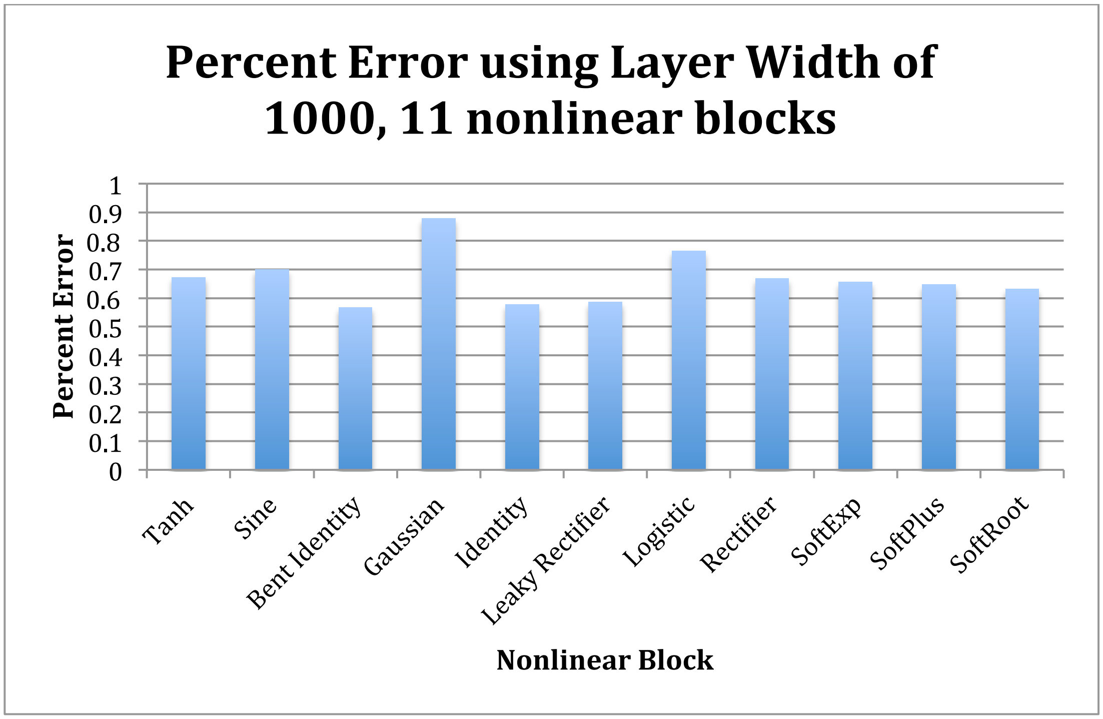
Plotting the 11 nonlinearities with a layer width of 31, we found that Logistic was the best activation function, with a percent error of 32.3954% error.



After plotting the 11 nonlinear blocks using a layer width of 100, SoftPlus was found to be the best nonlinear block, with a percent error of 35.57%.

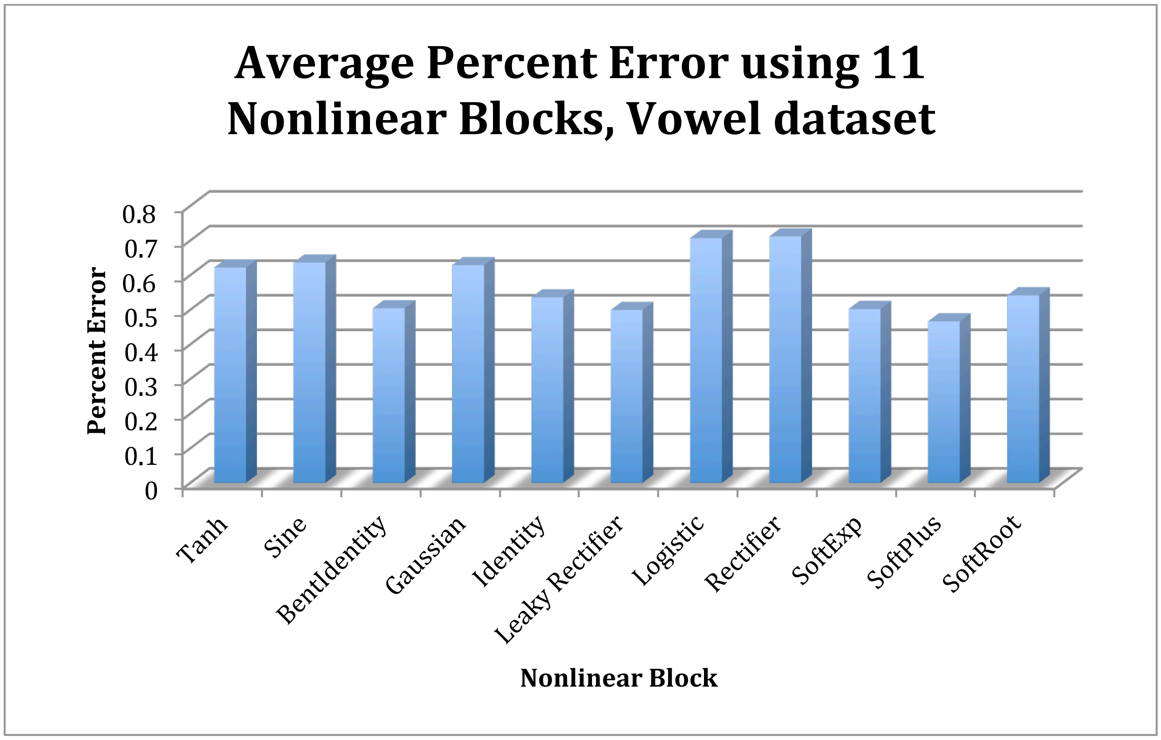


Plotting the average percent error using a layer width of 316, the best activation function proved to be SoftPlus, with a percent error of 47.8355% error.


Percent Error using Layer Width of 1000, 11 nonlinear blocks

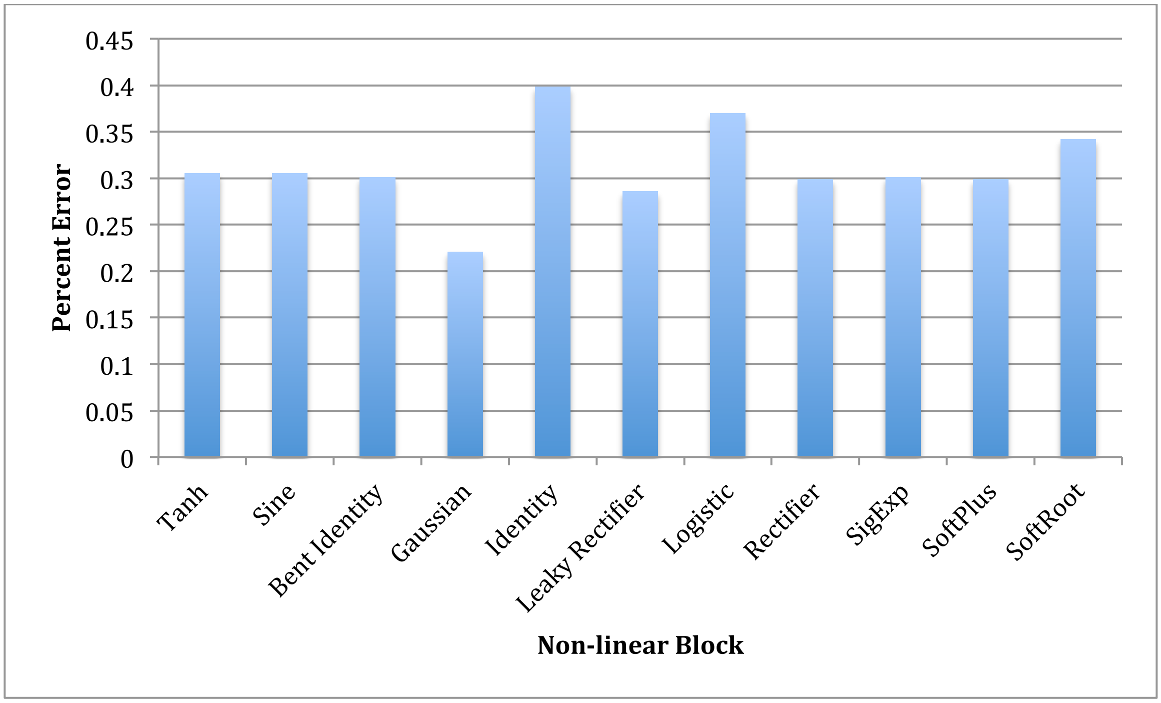
Using a layer width of 1000, we plotted the average percent error of the 11 nonlinear blocks, and found that SoftPlus was the best activation function, with a percent error of 35.57%.

Finally, we decided to plot, using a layer width of 3162, the average percent error using 11 nonlinear blocks. However, as the experiments hadn't completed, we decided not to use the results from these tests, as they only tested tanh nonlinear block.



SoftPlus was the best average nonlinear block for the vowel dataset, with an average percent error of 46.58%. Leaky Rectifier came in second, with an average percent error of 49.86%. On the other hand, Rectifier seemed to be the worst nonlinear block for the vowel dataset, giving an average percent error of 71.13%. The second worst nonlinear block was Logistic, with an average percent error of 70.62%.

Graphing the minimum error of each of the nonlinear blocks, we found the below graph:



Considering the above average percent errors in each layer width, we must consider the best optimizer and topology that gives the lowest percent error. Gaussian nonlinear block was the best nonlinear block for the vowel dataset, with a 22.0779% error.

Conclusion

We analyzed 3 optimizers, 12 non-linearities, 4 topologies, and 2 datasets. With the MNIST dataset, each experiment ran for 200 epochs over 60,000 training patterns and 10,000 test patterns. There were a total of 330 permutations, and these experiments took 68 days to run. With this in-depth analysis of the optimal optimizer, topology, and non-linearities, we found Leaky Rectifier, with Stochastic Gradient Descent Optimizer, using a layer width of 316, at 75 epochs, gives 1.29% error, proving to be the best nonlinear block for the MNIST dataset. For the Vowel dataset, we found that Gaussian nonlinear block, tested using Stochastic Gradient Descent optimizer and a layer width of 316, was found to be the best nonlinear block, giving a percent error of 22.0779%.

References

[1] Stigler, Stephen M. "Gauss and the invention of least squares." *The Annals of Statistics* (1981): 465-474.

[2] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *Cognitive modeling* 5, no. 3 (1988): 1.

[3] Cybenko, George. "Approximation by superpositions of a sigmoidal function." *Mathematics of Control, Signals, and Systems (MCSS)* 2, no. 4 (1989): 303-314.

[4] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation* 18, no. 7 (2006): 1527-1554.

[5] Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." *Neural networks* 61 (2015): 85-117.

[6] Cireşan, Dan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. "A committee of neural networks for traffic sign classification." In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 1918-1921. IEEE, 2011.

[7] Pinto, Nicolas, David D. Cox, and James J. DiCarlo. "Why is real-world visual object recognition hard?." *PLoS Comput Biol* 4, no. 1 (2008): e27.

[8] Gashler, Michael. "Waffles: A machine learning toolkit." *Journal of Machine Learning Research* 12, no. Jul (2011): 2383-2387.

[9] https://github.com/mikegashler/waffles

[10] http://uaf46365.ddns.uark.edu/data/mnist/

[11] http://uaf46365.ddns.uark.edu/lab/ml.pdf

[12] https://en.wikipedia.org/wiki/Activation_function