COMPARISON OF PERFORMANCE OF WARD SYSTEMS' NEURAL NETWORKS APPLIED TO A MULTIFONT RECOGNITION PROBLEM

STUDENT PAPER

Rick Niess, Lenny Scardino, William Douglas, Marcin Paprzycki
Department of Computer Science and Statistics
University of Southern Mississippi
Hattiesburg, MS 39406-5106
Rick.Niess@usm.edu

ABSTRACT

Results of performance comparison of the three Ward Systems proprietary neural network architectures and a simple, three-layer, backpropagation neural network applied to a simplified pattern recognition problem are presented and discussed. The varying parameters are the size of the input data and the number of nodes in the hidden layer slabs.

INTRODUCTION

In spite of their relative youth (relative to other areas of computer science), neural networks (NNs) are quickly becoming popular tools for solving problems not easily solved in an algorithmic manner (see Rumelhardt et. al. 1989, Looney 1997 and the references collected there). NNs themselves are based on the biological network structure found in the human brains. The human brain contains billions of neurons, each interconnected with those around it in many different ways. It is this massively parallel arrangement which allows us to recognize and reason. However, current NNs are considerably less complex than this. Most of them only utilize a few thousand neurons or less. However they turn out to be very useful as decision making tools when dealing with problems in which the input data has some known or unknown correlation to the desired output.

In this paper, we focus on only one application: a simplified version of the multifont recognition problem. In particular, we will consider the character recognition of the twenty-size uppercase letters of the Latin alphabet represented in a number of unique fonts (see DATA GENERATION below). In the experimental setup it is assumed that one letter is to be recognized at a time and that each image presented does indeed represent a letter. The aim of or work is to report on an attempt to compare the performance

characteristics of the three Ward Systems' NN architectures available in the NeuroShell 2.0 package (Ward 1998) when applied to a pattern recognition model-problem. This software has been used in earlier work (Bowers and Paprzycki 1999, Bowers et. al., 1999a, Bowers et. al. 1999b, Bowers et. al. 1999c, Paprzycki et. al. 1999). Results reported there were obtained for a relatively small input data set (limited primarily by the available computing power).

The remaining part of this paper is organized as follows. In the next section we briefly introduce the three Ward NN architectures used in our experiments, (details for all architectures should be found in the literature cited). Since the basic three-layer backpropagation NN is a very well known architecture, we leave out its description. We follow with a description of how the data was generated. Finally, we describe the experiments performed and discuss their results.

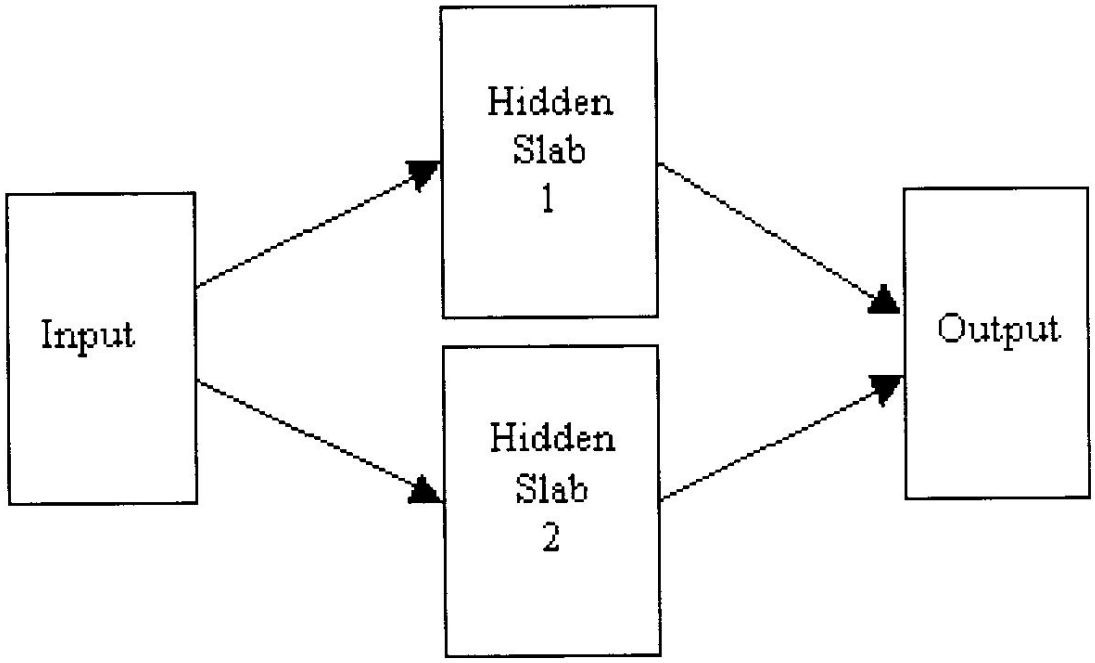

Figure : A simple WNN

WARD SYSTEMS NEURAL NETWORK ARCHITECTURES

As a part of the NeuroShell package Ward Systems included three proprietary NN architectures named Ward Networks (WNNs). While they use the general backpropagation-type learning algorithm, their uniqueness is in the fact that they contain multiple parallel groups (slabs) of neurons in the hidden layer, each of which uses a different activation function. The idea behind this approach is that each separate slab (due to the different activation function) will "detect" different characteristics of the input data. These different "views" on the input are then combined in the output layer thus improving the overall pattern recognition capabilities of the network. There are three variants of this architecture available in the NeuroShell package. All three are three-layer architectures (input, output, and one hidden layer) but with differing numbers of parallel slabs in the

hidden layer and with different arrangements of connections. For lack of proper names, we refer to them hereafter as simple, moderate, and advanced corresponding to their placing (left, middle, and right respectively) in the NeuroShell network selection window. For all architectures, we used (20*20 = 400) input nodes (each node corresponding to one element of the input vector, see next section) with the default *linear [-1,1]* activation function and 26 output nodes (corresponding to the 26 letters of the alphabet) with the default *logistic* activation function.

The simple architecture (see Figure 1) uses two parallel slabs in the hidden layer.

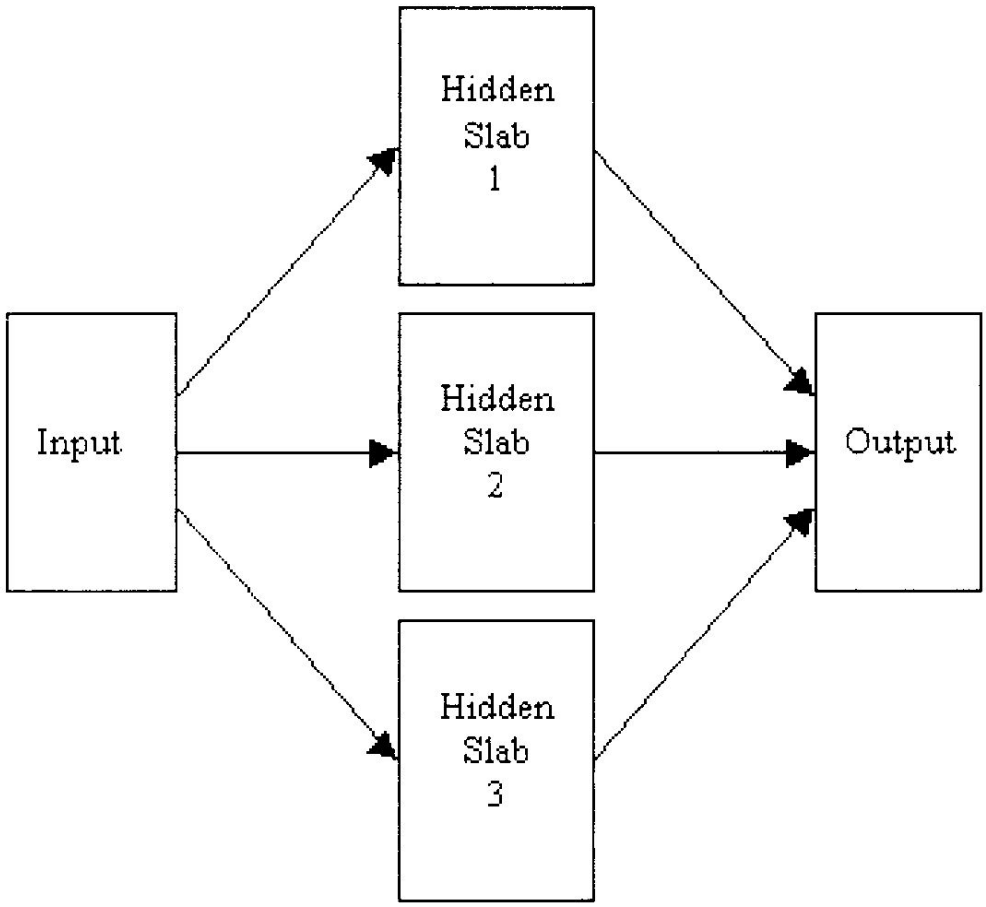

Figure : A moderate WNN

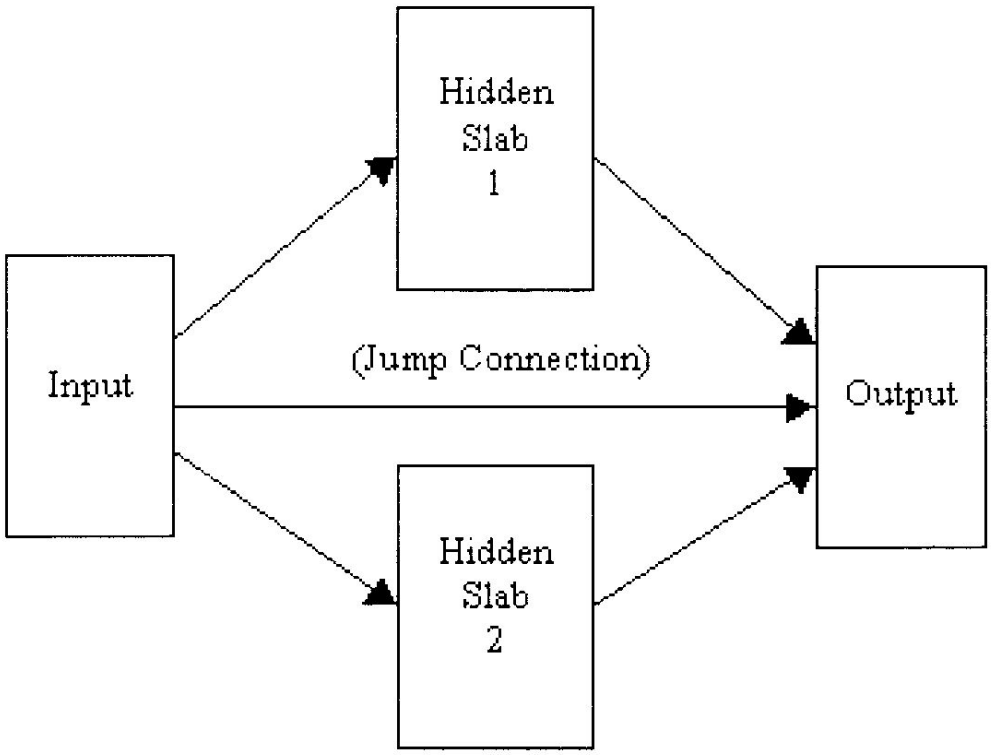

Figure : An advanced WNN

Both receive input directly from the input layer and connect directly to the output layer. One hidden slab uses a *Gaussian* activation function and the other *Gaussian complement* function.

The moderate architecture (see Figure 2) uses three parallel slabs in the hidden layer that receive input from the input layer and connect to the output layer. Two of the slabs use *Gaussian* and *Gaussian compliment* as in the simple architecture, while a third slab uses a *tanh* activation function.

The advanced architecture (see Figure 3) uses only two parallel slabs that each receive input from the input layer and each connect to the output layer. As with the simple architecture, *Gaussian* and *Gaussian compliment* activation functions are used in these slabs. However, there is also a "jump connection" directly from the input slab to the output slab.

DATA GENERATION

As noted in the recent paper (Paprzycki et. al. 1999) we developed a digitizer based on FreeType 1.2, a TrueType font rendering library (FreeType 1999), to create our data. Using this digitizer, letters were centered in a 20x20 "bit-map" allowing a one-pixel border around the map (so the effective letter image was at most 18x18). We began with a very large number of fonts (initially 2450). We manually compared these fonts to remove identical and similar fonts that exist under different names. Only the basic forms were allowed (no italic or script fonts have been used). After a third screening, we ended up with 710 unique fonts for use in this experiment. We then processed the fonts with our digitizer to obtain data in the format required by the NeuroShell environment (each letter represented as a sequence of 400 digits 0 and 1, followed by a 26 digit vector representing the true output). The resulting data was divided into 6 groups of fonts numbering 100, 200, ..., and 600 which were used as training sets. The remaining 110 fonts were used as a testing set (this set was not shown to the network during training). We divided the data into groups alphabetically (based on the font filenames). Since there is no relationship between the name of the font and its shape, this approach did not have any effect on the results.

EXPERIMENTAL RESULTS

Performance Comparison of Three WNN Architectures

We experimented with all three forms of Ward Nets available in the NeuroShell environment (see above). In each case the control variables were the amount of input data (100, 200, ..., 600 fonts) and the total number of nodes in the hidden layer. According to the NeuroShell documentation, the total number of neurons in **all** slabs in the hidden layer combined should correspond to the number of neurons in a single hidden layer in the simple backpropagation architecture. This value is calculated according to the formula "# of neurons in the hidden layer = 1/2 (Inputs + Outputs) + square root of the number of patterns in the Training [set]" (NeuroShell online documentation). Thus we have divided

the total number of neurons corresponding to 100, 200, 300 and 400 neurons in the single hidden layer evenly between the slabs. This resulted in using 50, 100, 150, and 200 nodes per slab in the simple and advanced WNNs and 34, 68, 102, and 136 nodes per slab in the hidden layer for the moderate WNN. In addition, for comparison we also used the NeuroShell-recommended default values for the number of neurons in the hidden layer as calculated using the above formula (see the Tables). We used the default values for all the remaining parameters (learning rate, stopping criteria, etc). Tables 1 through 6 summarize the results for the simple (Tables 1 and 2), moderate (Tables 3 and 4), and advanced (Tables 5 and 6) WNN architectures. In each pair of tables we report the percent of correct answers obtained first, when the training data was shown to the trained network and, second, when the test set was shown to it.

Table 1: Correct answers; simple WNN; training set recognition in %.

Fonts	**50 nodes/ slab**	**100 nodes/ slab**	**150 nodes/ slab**	**200 nodes/ slab**	**default nodes/ slab**	**100**
1.69	**5.69**	**5.88**	**7.65**	**4.88**	**(130 nodes)**	**200**
1.88	**2.5**	**7.83**	**6.31**	**4.38**	**(138 nodes)**	**300**
1.44	**4.73**	**6.53**	**5.64**	**4.41**	**(146 nodes)**	**400**
1.67	**4.59**	**3.22**	**6.63**	**2.39**	**(152 nodes)**	**500**
2.25	**6.05**	**4.34**	**5.26**	**6.41**	**(158 nodes)**	
600	**1.47**	**6.44**	**5.33**	**3.84**	**7.14**	**(162 nodes)**

Table 2: Correct answers; simple WNN; test set recognition in %.

Fonts	**50 nodes/ slab**	**100 nodes/ slab**	**150 nodes/ slab**	**200 nodes/ slab**	**default nodes/ slab**	**100**
1.22	**4.26**	**4.76**	**5.49**	**4.65**	**(130 nodes)**	**200**
1.64	**2.55**	**7.31**	**5.8**	**3.74**	**(138 nodes)**	**300**

1.26	4.16	6.92	5.14	4.37	(146 nodes)	400
1.75	3.85	2.52	6.43	2.13	(152 nodes)	500
2.17	6.43	3.84	4.97	5.66	(158 nodes)	600
1.19	6.12	5.52	3.53	7.06	(162 nodes)	

It is easy to notice that the performance of the simple architecture is very poor regardless of the amount of training data and number of nodes in the hidden layer. Using larger data sets seems to improve recognition of both the training and test sets slightly as does increasing the number of nodes in the hidden layers. We do, however, notice that for training sets of size 400 fonts and smaller, using 200 nodes in the hidden layer notably increases recognition over those for the suggested default numbers of nodes in the hidden layer. Similarly, for the test set, the suggested default value is clearly not the best possible.

Table 3: Correct answers; moderate WNN; training set recognition in %.

Fonts	34 nodes/ slab	68 nodes/ slab	102 nodes/ slab	136 nodes/ slab	default nodes/ slab	100
90.23	95.31	94.92	96.04	95.38	(86 nodes)	200
85.65	91.67	92.19	94.23	93.28	(92 nodes)	300
87.21	91.71	92.83	93.79	90.01	(97 nodes)	400
86.35	93.12	92.57	94.05	94.15	(101 nodes)	500
86.7	93.68	94.14	94.28	93.68	(105 nodes)	
600	85.97	91.56	91.26	92.7	92.82	(108 nodes)

Table 4: Correct answers; moderate WNN; test set recognition in %.

Fonts	34 nodes/ slab	68 nodes/ slab	102 nodes/ slab	136 nodes/ slab	default nodes/ slab	100
78.29	81.08	80.38	80.94	80.14	(86 nodes)	200
80.42	84.3	84.09	85	85.31	(92 nodes)	300

81.5	85.52	86.6	86.47	84.58	(97 nodes)	400
81.47	85.84	86.92	87.13	87.13	(101 nodes)	500
82.73	87.52	87.52	88.7	86.95	(105 nodes)	600
82.69	86.22	86.85	86.82	88.22	(108 nodes)	

Comparing all results presented here, the moderate architecture appears to be the best of the three. We note that using larger training sets seems to degrade the network's ability to recognize its training set while enhancing its ability to recognize the test set. We interpret this behavior as an increase of the network ability to generalize. We also note that the increase in accuracy plateaus somewhere near using 68 nodes per slab in the hidden layer. It can be observed, again, that the proposed default value of neurons per slab, while resulting in a relatively good architecture, does not necessarily lead to the best performance.

Table 5: Correct answers; advanced WNN; training set recognition in %.

Fonts	50 nodes/slab	100 nodes/slab	150 nodes/slab	200 nodes/slab	Default nodes/slab	100
68.54	57.96	64.46	65.38	65.38	(130 nodes)	200
63.15	57.48	58	55.63	51.88	(138 nodes)	300
60.58	66.01	52.49	65.14	55.54	(146 nodes)	400
63.3	65.56	64.26	62.01	59.55	(152 nodes)	500
66.76	61.27	63.76	54.28	54.34	(158 nodes)	
600	58.9	58.72	60.04	60.45	62.71	(162 nodes)

Table 6: Correct answers; advanced WNN; test recognition in %.

Fonts	50 nodes/slab	100 nodes/slab	150 nodes/slab	200 nodes/slab	Default nodes/slab	100
57.45	57.41	50.42	57.55	54.79	(130 nodes)	200
57.97	53.39	54.05	51.71	46.68	(138 nodes)	300
57.97	61.75	50.17	61.26	51.57	(146 nodes)	400
60.94	61.85	60.35	58.29	55.87	(152 nodes)	500
63.04	58.18	61.85	51.75	51.26	(158 nodes)	
600	56.29	56.33	56.69	59.13	60.8	(162 nodes)

Comparing the results of the last two experiments reported earlier with these, it could be easily observed that the advanced architecture, while significantly better than the simple architecture, falls behind the moderate architecture. Increasing the number of nodes in the hidden layers appears raise network's competency slightly with the exception of a drop in accuracy when using 150 nodes per slab in the hidden layer. We will investigate this phenomenon in further experiments.

Performance Comparison Between WNNs and Simple Backpropagation Neural Network (BPNN)

In order to gain perspective on the performance of the three WNNs we have chosen to compare their performance to that of the simple three-layer BPNN. The results are of interest as the BPNN is one of the simplest and oldest NN architectures. It is also the architecture that the WNNs are based on and intended to improve upon. In our experiments with BPNNs, we have used the same methodology as above. The only factors taken into consideration were input data size and number of neurons in the hidden layer.

Table 7: Correct answers; simple BPNN; training set recognition in %.

Fonts	100 nodes/slab	200 nodes/slab	300 nodes/slab	400 nodes/slab	Default nodes/slab	100
96.19	96.69	96.81	96.77	96.42	(259 nodes)	200
94.48	95.81	96.1	96.69	96.31	(277 nodes)	300
95.72	94.56	96.36	95.96	96.71	(292 nodes)	400
94.31	95.92	96.4	97.12	96.23	(304 nodes)	500
95.67	95.51	95.81	96	95.68	(315 nodes)	
600	92.74	93.87	95.49	93.84	95.33	(325 nodes)

Table 8: Correct answers; simple BPNN; test set recognition in %.

Fonts	100 nodes/slab	200 nodes/slab	300 nodes/slab	400 nodes/slab	default nodes/slab	100
82.55	82.45	82.62	82.66	82.62	(259 nodes)	200
85.91	86.15	86.68	87.24	86.75	(277 nodes)	300
88.01	87.38	88.18	87.55	88.5	(292 nodes)	400
88.53	89.34	89.48	89.48	89.37	(304 nodes)	
500	89.67	89.23	89.68	89.69	89.62	(315 nodes)
600	88.46	88.99	90.45	89.37	90.24	(325 nodes)

Since the results of the experiments with the BPNN turn out to be relatively good, we will compare them only with the best WNN (moderate architecture). Before we proceed, it should be noted that, since the simple BPNN has only one slab in the hidden layer, the number of its nodes appears to be larger. This is, however, not the case as the total number of neurons in the hidden layer (when all slabs are taken into account) is the same (it may be off by one or two when the division is not exact). We can see that, across the board, the BPNN outperforms the WNN while following nearly identical improvement patterns. Interestingly, BNN seems to be somewhat "more stable" and less dependent on both the size of the input and the number of nodes in the hidden layer. Moreover, the performance difference between the default number of neurons and the "arbitrarily" selected numbers is almost negligible.

CONCLUDING REMARKS

We have experimented with the three Ward Systems proprietary neural network architectures as applied to the pattern recognition model-problem. We have found that one of them (moderate) is highly competent in solving our simplified problem, while the simple architecture does not perform well at all. We plan to investigate the reason for such a poor performance of the simple architecture further.

We have also compared the performance of WNNs with that of a simple, three-layer backpropagation network. We have found that the backpropagation architecture is slightly less dependent on the size of the input and the number of nodes in the hidden layer and across the board slightly outperforms the Ward Networks. While our experiments do not allow us to make serious generalizations, we can state that Ward Systems attempt to improve the backpropagation architecture did not result in an obvious success for our model problem.

REFERENCES

Looney, C.G. 1997. *Pattern Recognition Using Neural Networks*. New York: Oxford University Press.

Bowers, S., Costeines, A. and Paprzycki, M. 1999a. Applying Probabilistic Neural Networks to the Multifont Recognition Problem with Large Training Set, in: Kumar, A.N. et. al. (eds.) *Proceedings of the Twelfth International Florida AI Research Society Conference*, 336-339, Menlo Park: AAAI Press.

Bowers, S., Costeines, A. and Paprzycki, M. 1999b. Evaluating the Performance of Three Neural Network Architectures Applied to the Pattern Recognition Problem, *Proceedings of the 15th Annual Conference on Applied Mathematics*, 29-36, Edmond: University of Central Oklahoma.

Bowers, S., Morrison, J. and Paprzycki, M. 1999c. Comparing Performance of Neural Networks Recognizing Machine Generated Characters, *Proceedings of the First Southern Symposium on Computing*, to appear, Hattiesburg: University of Southern Mississippi.

Bowers, S. and Paprzycki, M. 1999. Applying PNN's to the Multifont Pattern Recognition Problem, in: Bainov, D., (ed.) *Proceedings of the 9th Colloquium on Difference Equations*, 311-318, Utrecht: VSP.

Burger, C., Traver, R., 1999. Applying Neural Networks to Risk Assessment. http://www.audit.uic.edu/DEPT/IAART5.HTM

FreeType 1999. http://www.freetype.org/

Paprzycki, M. Niess, R., Thomas, J., Scardino L., and Douglass, W. 1999. Comparing Performance of Neural Networks Applied to a Simplified Recognition Problem, *Proceedings of the Thirteens International Florida AI Research Society Conference*, to appear.

Rumelhardt, D.E., McClelland D.L., and the PDCP Research Group. 1986. *Parallel Distributed Processing*. Cambridge: MIT Press.

Ward. 1998. http://www.wardsystems.com