Proceedings of the First Southern Symposium on Computing
The University of Southern Mississippi, December 4-5, 1998

COMPARING PERFORMANCE OF NEURAL NETWORKS RECOGNIZING MACHINE GENERATED CHARACTERS

SEAN BOWERS[*], JODY MORRISON[*] AND MARCIN PAPRZYCKI[*]

Abstract. Neural networks are a popular tool in the area of pattern recognition. However, since a very large number of neural network architectures exist, it has not been established which one is the most efficient. In this paper we compare the performance of three neural network architectures: Kohonen's self-organizing network, probabilistic neural network, and a modified backpropagation applied to a simplified character recognition problem.

1. Introduction. The pattern recognition problem (as defined in computer science) consists of developing algorithms that allow a computer to correctly identify a given pattern. Such a pattern can be a sound or an image and it has to be represented in a form recognizable by a computer. For example, an image could be a photograph, a fingerprint, or a handwritten text. Such an image has to be digitized before a computer can be asked to recognize its "content." Since the pattern recognition problem is so broad, we restrict our attention to the writing recognition. More precisely, we devote our attention to the recognition of machine generated capital letters of the Latin alphabet.

Neural networks (NN) have been around since mid-1950s when they were introduced 1950s by Rosenblatt [8]. In the late 1950s and early 1960s they became a direct competitor to other approaches to artificial intelligence. At this time NN's were not able to deliver promised results (partially due to the lack of computational power of existing computers). In addition, in 1969, M. Minsky and S. Papert published their famous book "Perceptrons: an Introduction to Computational Geometry" [5] in which they heavily criticized the neural network approach. Very few researchers remained in the field in the 1970s. Only when the standard artificial intelligence research stalled in the late 1970s did neural network research resumed. This effort produced results when, in 1986, D. E. Rumelhardt, J. L. McClelland and the PDCP research group [9], reintroduced NN's.

NN's attempt to model the problem solving processes occurring in a human brain. A typical network consists of a number of input nodes (neurons) which process the incoming data. They are usually referred to as the input layer. These nodes can be connected to a group of output nodes (output layer) or they can be connected to one or more groups of intermediate nodes (hidden layers). Each of the connections between two nodes is characterized by a weight that represents the strength (importance) of a given connection. NN's learn from "experience." Training consists of showing patterns to the network and causing adjustments of the weights. A correctly designed training leads to a network that achieves a high level of generalization and is capable of giving correct responses to the data that has not been previously encountered. In addition, NN's are able to extract essential information from noisy data and often deliver solutions faster than more conventional methods of classification.

Nowadays there exist a very large number of NN architectures. They differ in the number and size of layers and the weight adjustment functions applied during the learning process. In addition, the learning process can proceed with or without supervision. What we find particularly interesting is the fact that, in most cases, when a network is designed its performance is studied "in vacuum," without a comparison with other network architectures. In addition, before the data is shown to the NN it is often pre-processed by a feature extracting software (to reduce the size/domain of the problem). This makes it rather difficult to verify the performance claims and establish the best network architecture. In our earlier work [1] we have compared the performance of the Probabilistic Neural Networks (PNN) and the Ward Nets applied to the letter recognition problem. This comparison was, however, very preliminary and the presented results

[*] Department of Computer Science and Statistics, University of Southern Mississippi, Hattiesburg, MS 39406.
Marcin.Paprzycki@usm.edu.

suffered from the extremely small size of the training sets. Since then we have expanded the size of the training set (increasing both the number of fonts and the size of the digitized letters) and reported the performance of PNNs for this new data set [6]. The aim of this paper is to compare the performance of PNNs with Ward nets and the Kohonen self-organizing networks. To achieve this goal we have used implementations of these three neural network architectures available in the NeuroShell package [13]. We have decided to conduct our study by using an "off the shelf" NN environment as a recognition of the fact that the field has matured enough for the researchers to be able to avoid re-implementing the NN architectures (and potentially introducing inefficiencies and/or errors). In addition, since all architectures are implemented by the same vendor and in the same package, we believe that our comparison is relatively fair. Finally, all of our comparisons are done with the data being shown directly to the network (without pre-processing). The aim of this approach is to study how the networks behave when the size of the input (and thus all layers) is relatively large.

The remaining parts of the paper are organized as follows. In Section 2 we briefly summarize the three NN architectures (more details can be found in [2,4] and the references in the section). Section 3 presents the information about the experimental setup (test data generation and training). In Section 4 the experimental results are presented and discussed.

2. Neural network architectures

2.1. Kohonen's self-organizing networks (KSN). Kohonen's networks were introduced in 1989 [3]. They are extremely simple, with only two layers (input and output), and are based on competitive unsupervised learning. When the training data is shown to the network it tries to separate it into a specified number of categories. The weight adjustment process during learning is based on the Kohonen self-organizing map and attempts at categorizing the data based on the perceived similarity between elements.

2.2. Probabilistic neural networks (PNN). Probabilistic neural networks were introduced in 1989 by D. Specht [10, 11, 12]. They are based on an application of a classification theory based on the minimization of the "expected risk" function (such strategies are typically called "Bayesian" and can be applied to problems with multiple categories as long as the decision boundaries remain relatively smooth). The key to the classification process is the ability to estimate the probability density functions, which, in case of NN, have to be estimated on the basis of the training patterns. Specht suggested that Parzen's probability density estimator should be used [7]. The PNN consists of three layers (one hidden layer). The application of probability density estimation meant that *all* information from the training set had to be stored (in the hidden layer) and used during testing and thus, for large training sets, an enormous amount of memory was required. This was a severe limitation in the early 1990's and it is only now that PNNs become more popular. On the positive side, during learning, each training pattern is shown to the network only once thus substantially reducing the training time.

2.3. Ward networks (WN). The most common types of NN's are backpropagation networks. Typical backpropagation architecture consists of three layers and learns via a supervised learning procedure. An input pattern from the training set is shown to the network and the produced output is confronted with the correct one. The difference between the computed and correct output is propagated back across the network to modify the strength of connections, according to the learning rules, and the next test pair is introduced to the network. This process is repeated until the network stabilizes and produces mostly correct outputs. Ward Systems Group introduced a modified backpropagation network (called Ward Net). WNs come in three variants differing by the number of slabs in the hidden layer (here a slab is defined as a group of neurons sharing certain characteristics and residing in a given layer [13]). WNs use a different activation function for each layer and each slab in the hidden layers (not the case for the standard backpropagation). The different activation functions are designed to detect different features of the data. This approach is supposed to improve the final classification results since the output layer has different "perspectives" on the data.

3. Experimental setup. In the initial paper [1] we presented results based on 5 fonts represented as 10x10 bit-maps (shifted and fuzzy data were also used in the experiments). Here, we expanded the data set used for experimentation. Using a Delphi program, provided by Daniel Hellsson of Unseen Technology, we

have generated a set of bit-maps of size 14x14 for the capital letters from the following fonts (which are standard fonts available on almost any PC running Windows 95):

Arial	Desdemona	School
Bookman	Haetten	Script
Brush	Impact	Tahoma
Colonna	Kino	TNR
Corsiva	Monocorsiva	Verdana
Courier	New Century	

In all cases letters were centered in the bit-map and 0's filled the "borders" of the map (so the effective letter image was at most 12x12).

We have also generated fuzzy representations of each letter of each font. Here, for each bit of the bit-map representation we have generated a random number and this bit was flipped (from 0 to 1 or from 1 to 0) with probability K/100 (where K denotes the level of "fuzzification"). Thus, e.g. for the data of type fuzzy 1 each bit of the map was flipped with probability 1/100 while for data type fuzzy 5 each bit was flipped with probability 5/100.

This data has been preprocessed to match the input/output requirements of the NeuroShell environment. In all networks we used (14*14 = 196) input nodes and 26 output nodes (corresponding to the 26 upper case letters of the Latin alphabet). The number of nodes in the hidden layer varied depending on the network architecture and the size of the input data. For the KSN there is no hidden layer. For the PNN there is one node in the hidden layer for each element in the input data set. In case of Ward nets we used the default 44 nodes in each slab of the hidden layer.

4. Experimental results

4.1. Individual fonts used for training. In the first series of experiments we established the similarity between various fonts as "perceived" by the three neural network architectures used as pattern classifiers. We trained the networks with a given font and for each network used the default settings of the NeuroShell package. We then presented to the trained network all remaining fonts and scored how many letters have been recognized correctly. Tables 1, 2 and 3 summarize our findings for the three network architectures. They contain the best results obtained for various values of the smoothing factor. (The smoothing factor parameter is applied to the trained network and represents the degree of expected generalization when the new data is being presented to it.) In the Tables, for each font, we also represent the total number of correct answers obtained across all fonts when (a) a given font was used to train the network (row-total), and when the given font was recognized by networks trained with other fonts (column-total). Since for both PNNs and WNs the network was able to correctly recognize all letters of the font used to train them it we left the appropriate spaces empty.

First, it can be observed that the Kohonen networks perform very poorly being able to recognize only very few letters from any font. It matches the results presented in [2, pp. 169ff] and suggests that recognition of characters is not the best application for this network architecture. From the displays generated by the NeuroShell during the KSN training we gather that the network does not manage to properly group letters into the required 26 categories. Some categories overflow with letters, while others remain almost empty.

Both PNNs and WNs perform much better. When the capability of a given font of being a "trainer" is considered (row totals), in all cases but for Haetten, PNNs deliver better results. When the capability of "being recognized" by networks trained by other fonts is considered, the overall performance of PNNs is better, however for Corsiva, Desdemona, Monocorsiva and Script WNs deliver slightly better results than PNNs.

	Arial	Bookman	Brush	Colonna	Corsiva	Courier	Desdemona	Haetten	Impact	Kino	Monocorsiva	New Century	School	Script	Tahoma	TNR	Verdana	**TOTALS**
Arial	3	0	1	0	2	1	0	1	0	1	0	1	0	0	2	1	2	15
Bookman	0	0	0	0	2	0	1	1	2	0	1	1	1	0	2	0	1	12
Brush	1	0	3	1	1	0	0	0	2	3	3	1	2	0	0	0	0	17
Colonna	0	1	2	0	3	0	1	2	0	1	1	0	2	2	1	1	2	19
Corsiva	1	0	0	1	2	1	0	0	0	1	1	1	0	2	1	0	1	12
Courier	0	1	1	1	0	0	1	0	0	2	2	0	1	1	2	0	1	13
Desdemona	4	1	1	1	2	2	1	2	2	2	1	0	1	1	1	2	1	25
Haetten	1	1	1	1	0	1	1	0	1	0	1	0	0	3	1	0	1	13
Impact	1	2	1	2	1	3	0	2	2	0	1	1	2	0	2	1	2	23
Kino	0	0	2	1	2	0	1	2	1	2	1	0	0	0	1	0	1	14
Monocorsiva	1	0	1	2	0	2	2	0	1	1	1	1	0	0	1	2	2	17
New Century	1	0	1	2	1	1	0	0	0	1	2	0	2	2	1	0	0	14
School	3	0	0	0	1	0	0	0	0	1	0	1	0	1	0	2	0	9
Script	0	0	1	2	1	1	1	0	0	2	1	1	1	1	1	1	0	14
Tahoma	1	0	1	0	2	0	1	1	0	2	3	0	1	1	0	0	1	14
TNR	2	1	1	1	2	1	0	0	1	2	0	0	1	1	3	1	1	18
Verdana	0	2	1	0	1	3	0	1	1	0	1	1	1	0	3	1	4	20
TOTALS	19	9	18	15	23	16	10	12	13	21	20	9	15	15	22	11	21	

TABLE 1. *Kohonen networks, single font training*

	Arial	Bookman	Brush	Colonna	Corsiva	Courier	Desdemona	Haetten	Impact	Kino	Monocorsiva	New Century	School	Script	Tahoma	TNR	Verdana	**TOTALS**
Arial		8	1	11	2	3	4	4	5	3	0	7	2	3	14	11	9	87
Bookman	9		3	8	2	4	5	6	2	4	1	9	6	0	12	6	13	90
Brush	2	6		5	3	1	2	3	2	3	3	1	3	1	5	4	5	49
Colonna	5	8	8		4	6	2	4	4	2	6	5	3	2	6	8	4	77
Corsiva	1	1	2	3		3	3	0	0	3	11	3	4	1	2	1	3	41
Courier	2	8	2	11	5		6	6	3	6	1	9	8	1	7	2	7	84
Desdemona	4	7	1	5	2	4		3	3	5	1	7	4	1	7	4	10	68
Haetten	5	10	2	4	4	6	6		11	6	1	7	3	0	4	4	6	79
Impact	4	6	4	5	2	1	6	14		6	1	5	3	1	5	6	7	76
Kino	2	3	2	1	3	6	6	4	4		3	4	2	1	5	2	5	53
Monocorsiva	0	0	5	5	19	2	3	1	1	2		3	4	2	0	0	0	47
New Century	8	10	3	6	5	7	8	4	4	3	2		13	2	10	6	20	111
School	2	5	6	5	11	3	5	3	3	1	5	9		4	1	3	6	72
Script	0	0	5	1	2	2	0	0	1	3	2	2	1		3	1	1	24
Tahoma	12	10	3	5	4	4	7	7	5	3	1	9	4	3		4	17	98
TNR	10	8	4	8	0	1	4	5	6	2	2	7	4	0	6		6	73
Verdana	10	11	3	5	5	7	7	6	6	6	1	19	7	3	16	8		120
TOTALS	76	101	54	88	73	60	74	70	60	58	41	106	71	25	103	70	119	

TABLE 2. *Probabilistic networks, single font training*

	Arial	Bookman	Brush	Colonna	Corsiva	Courier	Desdemona	Haetten	Impact	Kino	Monocorsiva	New Century	School	Script	Tahoma	TNR	Verdana	**TOTALS**
Arial		4	2	4	4	2	3	1	2	2	2	4	3	4	6	8	4	55
Bookman	6		1	7	7	1	7	3	3	1	4	1	5	2	9	8	11	76
Brush	1	1		4	3	0	3	0	2	0	3	1	1	3	2	1	2	27
Colonna	5	5	2		7	2	4	2	5	3	4	4	4	2	6	5	6	66
Corsiva	3	1	0	1		3	3	2	3	1	6	3	2	0	1	3	1	33
Courier	3	7	2	8	8		5	4	5	4	2	3	4	1	6	1	6	69
Desdemona	2	2	1	2	5	5		4	5	3	4	2	6	3	1	1	5	51
Haetten	4	9	5	6	4	6	7		10	5	3	5	4	1	4	5	3	81
Impact	2	3	3	3	4	2	7	7		10	10	2	3	4	1	5	3	69
Kino	4	3	3	1	1	3	3	4	6		3	4	3	1	5	2	5	51
Monocorsiva	1	2	2	3	12	1	4	1	1	3		2	3	1	1	1	1	39
New Century	2	1	1	3	2	3	3	3	3	3	1		12	0	1	1	3	42
School	0	2	1	3	7	1	3	0	1	1	1	7		1	1	2	7	38
Script	2	6	2	4	4	1	1	1	2	2	3	2	1		4	2	4	41
Tahoma	7	8	1	4	6	2	11	4	3	3	2	3	4	0		4	13	75
TNR	7	7	1	4	3	1	5	2	3	1	2	0	2	2	2		3	45
Verdana	5	3	2	3	7	5	10	4	5	8	0	1	4	2	14	4		77
TOTALS	54	64	29	60	84	38	79	42	59	50	50	44	61	27	64	53	77	

TABLE 3. *Ward networks, single font training*

It can be observed that some fonts are relatively good as "trainers." In particular, PNNs trained with Verdana and New Century recognize substantially more letters of other fonts (in average 7.5 and 6.93 letters per font respectively) than networks trained with the remaining fonts. Verdana and New Century also seem rather easy to recognize by other fonts as they have the highest scores in columns. At the same time Script and Corsiva are extremely bad choices as training sets (they lead to the recognition of about 1.5 and 2.56 letters per font) while Script and Monocorsiva are the most difficult to recognize. In case of WNs the situation is slightly worse. The best "trainer" is Haetten and it allows correct recognition of an average of 5.06 letters per font. At the same time the worst "trainer" (Brush) is better than the worst PNN "trainer" and allows average recognition of 1.68 letters per font.

4.2. Experiments with fuzzy fonts. In the second series of experiments we have added to each font used for training its fuzzified versions. This was an attempt at adding diversity to the data and thus possibly increasing the generalization capabilities of the networks. To a group of seven individual fonts we have added a total of nine of their fuzzified versions with degrees of fuzzification 1, 2, …, and 9. It should be observed that since our data was represented by a 14x14=196 bit-map and the probability of a single bit being flipped was K/100, than for the fuzzification level 1 there were approximately 2 bits flipped in the bit-map. For the fuzzification level 9 there were approximately 18 bits flipped in each letter. We have combined the standard and fuzzified fonts into one group of 260 letters and used it as the training set. After the network was trained we have shown to it all remaining basic fonts. In all cases the default training setup for all network has been used. The results are summarized in Tables 4, 5 and 6. As previously, for PNNs and WNs, the networks were able to correctly recognize the basic font that was used for training and we left the appropriate spaces empty. For each case the best results (for the optimal smoothing factor) are reported. It should be added that for each network we have also experimented with other fonts as well and results were quite similar (thus only the intersection of fonts used on all architectures is reported).

	Arial	Bookman	Brush	Colonna	Corsiva	Courier	Desdemona	Haetten	Impact	Kino	Monocorsiva	New Century	School	Script	Tahoma	TNR	Verdana	**TOTAL**
Arial + F	1	1	0	0	2	1	0	2	1	0	2	0	0	1	1	2	1	15
Brush + F	0	1	0	0	0	1	0	0	1	0	1	0	2	1	2	1	0	10
Corsiva + F	1	1	0	1	0	0	0	0	0	0	0	1	1	0	1	0	1	7
New Century + F	1	1	1	1	0	0	1	1	2	0	2	3	1	1	1	0	1	17
Script+ F	0	1	0	0	1	1	2	1	1	1	0	1	1	1	1	0	1	13
Tahoma+ F	0	0	1	0	0	1	0	0	0	1	0	1	1	2	0	2	0	9
Verdana+ F	0	0	1	1	0	1	0	0	0	0	1	1	1	0	0	0	0	6

TABLE 4. *Kohonen networks, training with a single font and its fuzzy representations*

	Arial	Bookman	Brush	Colonna	Corsiva	Courier	Desdemona	Haetten	Impact	Kino	Monocorsiva	New Century	School	Script	Tahoma	TNR	Verdana	**TOTAL**
Arial + F		8	1	11	2	3	4	4	5	3	0	7	2	3	14	11	9	87
Brush + F	2	6		5	3	1	2	3	2	3	3	1	3	1	5	4	5	49
Corsiva + F	1	1	2	3		3	3	0	0	3	11	3	4	1	2	1	3	41
New Century + F	8	10	3	6	5	7	8	4	4	3	2		13	2	10	6	20	111
Script+ F	0	0	5	1	2	2	0	0	1	3	2	2	1		3	1	1	24
Tahoma+ F	12	10	3	5	4	4	7	7	5	3	1	9	4	3		4	17	98
Verdana+ F	10	11	3	5	5	7	7	6	6	6	1	19	7	3	16	8		120

TABLE 5. *Probabilistic networks, training with a single font and its fuzzy representations*

	Arial	Bookman	Brush	Colonna	Corsiva	Courier	Desdemona	Haetten	Impact	Kino	Monocorsiva	New Century	School	Script	Tahoma	TNR	Verdana	**TOTAL**
Arial + F		5	0	1	0	2	1	3	2	0	0	1	1	0	6	8	7	37
Brush + F	0	0		1	0	0	1	0	0	0	0	1	0	2	1	0	0	6
Corsiva + F	0	0	1	2		0	1	0	0	0	4	0	2	1	1	0	1	13
New Century + F	0	4	3	2	1	4	2	2	2	3	3		7	2	7	0	18	60
Script+ F	0	1	2	0	0	0	1	0	0	0	0	0	0		2	0	0	6
Tahoma+ F	7	5	1	0	1	3	3	1	1	1	0	3	0	0		2	14	35
Verdana+ F	5	7	2	0	2	4	5	3	3	2	0	7	2	0	13	3		53

TABLE 6. *Ward networks, training with a single font and its fuzzy representations*

The results clearly indicate that adding the fuzzified data does not lead to improvements in the network' performance. In case of Kohonen networks, only for the New Century font a slight improvement was observed, while for most fonts a performance drop resulted. For PNNs there was no difference in performance. WNs behaved similarly to the KSNs as only for the New Century font the results improved.

4.3. Using all fonts but one as a training set. In the last series of experiments we have used all fonts but one to train the network and then use this one missing font to test the pattern recognition capabilities of

the network. The results for all three network architectures are summarized in Table 7. As previously, the default training parameters have been used for all networks.

	Kohonen	Probabilistic	Ward
Arial	0	19	8
Bookman	0	15	10
Brush	1	6	7
Colonna	2	11	10
Corsiva	3	14	6
Courier	0	6	7
Desdemona	0	11	6
Haetten	0	13	6
Impact	0	9	9
Kino	3	7	6
Monocorsiva	0	4	3
New Century	2	12	9
School	1	16	10
Script	1	2	1
Tahoma	2	19	20
TNR	0	13	8
Verdana	1	19	21
TOTALS	16	196	147

TABLE 7. *Performance of the NN for* 15 *fonts (*390 *letters) in the training set.*

The results are rather surprising. In case of the Kohonen networks there is absolutely no performance increase achieved by adding more diversity to the data. The situation changes only slightly for PNNs and WNs. Here there are some fonts for which adding new data represents an improvement over the recognition by any of the single-font networks. However, such improvements are only minimal. Overall, even in the *best* case of the Verdana and Tahoma fonts only 73-81% of the letters are recognized correctly.

5. Concluding remarks. In this paper we have compared the performance of three neural network architectures applied to the restricted problem of recognition of machine generated characters. We have found that the performance of neither is extremely impressive, with the Kohonen network proving that it is not very well suited to solve this problem.

In the near future we plan to further expand the bitmap size and the number of font used in the experiments. We have obtained a CD with a large number of fonts and we plan to use it to prepare our test data. We will then apply to the problem all neural network architectures available in the NeuroShell package.

REFERENCES

[1] A. N. FAIRCHILD AND M. PAPRZYCKI, *Performance Evaluation of Neural Networks Applied to the Pattern Recognition*, in Proceedings of the 14th CAM, University of Central Oklahoma Press, Edmond, OK (1998), pp. 161-172.

[2] L. FAUSETT, *Fundamentals of Neural Networks,* Prentice Hall, Upper Saddle River, NJ (1994).

[3] T. KOHONEN, *Self Organization and Associative Memory*, Springer-Verlag, Berlin (1989).

[4] C. G. LOONEY, *Pattern Recognition Using Neural Networks*, Oxford University Press, New York (1997)

[5] M. MINSKY AND S. PAPERT, *Perceptrons: an Introduction to Computational Geometry*, MIT Press, Cambridge (1969).

[6] M. PAPRZYCKI AND S. BOWERS, *Applying Probabilistic Neural Networks to the Multifont Recognition Problem* (submitted for publication)
[7] E. PARZEN, *On Estimation of a Probability Density Function and Mode*, Ann. Math. Stat., 33 (1962), pp. 1065-1076
[8] F. ROSENBLATT, *Mechanisation of Thought Processes*, Proceedings of a Symposium held at the National Physical Laboratory, London, Her Majesty's Stationary Office, 1 (1958), pp. 449-458
[9] D. E. RUMELHARDT, D. L. MCCLELLAND AND THE PDCP RESEARCH GROUP, *Parallel Distributed Processing*, MIT Press, Cambridge (1986).
[10] D. SPECHT, *Probabilistic Neural Networks for Classification, Mapping, or Associative Memory*, Proceedings IEEE Conference on Neural Networks, IEEE Press, Los Alamitos, CA, vol. 1 (1989), pp. 525-532.
[11] D. SPECHT, *Probabilistic Neural Networks*, Neural Networks, 3 (1990), pp. 109-118.
[12] D. SPECHT, Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification, IEEE Transactions on Neural Networks, 1(1) (1990), pp. 111-121.
[13] WARD SYSTEMS GROUP, INC, *NeuroShell* 2 *User's Manual*, Ward Systems Group, Inc., (1996), (see also `http://www.wardsystems.com`).