Comparative Analysis of Basic Self-Organizing Map and Neocognitron for Handwritten Character Recognition

Ivo R. Draganov[1] and Antoaneta A. Popova[2]

Abstract – **In this paper we present a comparative analysis of the performance both as accuracy and time consumption of two neural network classifiers for handwritten character recognition – a basic self organizing map and neocognitron proposed by Kohonen and Fukushima respectively. The results of our study are found useful for characters and pseudo-characters recognition as a certain stage of processing in a whole handwriting recognition system.**

Keywords – **character, pseudo-character, word, handwriting recognition, self-organizing map, neocognitron.**

I. Introduction

A possible concept used in some systems for handwriting word recognition is splitting words in single characters and pseudo-characters [1]. Thus the separated elements of a word should be recognized at a later stage in these systems using a number of preliminary chosen rules based on common handwriting and language characteristics to form word hypotheses. After confirming or rejecting these hypotheses applying lexicon verification along with language statistics and other techniques we are able to recognize handwritten text word by word.

Considering the large number of methods developed for handwritten character recognition [2,6] and their capability to recognize pseudo-characters (parts of one or more characters) we decide to directly compare a self-organizing map [3] and a neocognitron [4] proposed by Kohonen and Fukushima respectively. These both self-learning neural networks will be compared in their basic form considering recognition accuracy with high quality images of characters free from noise, shifts, etc. Our main goal is to find out to what degree they differ one from another as for their recognition capability based on the different architecture and working principles they have when high quality test data is passed to them.

In the next part we present a description of the structure, preprocessing, learning and recognition algorithms used for both networks in our study. The third part contains the experimental results for different parameters defined in part two. In the last part a conclusion is made which of these classifiers is more appropriate to be included in a complete handwriting recognition system.

II. Compared Classifiers Description

The architecture of the self-organizing feature map (SOFM) used in our experiments is given in Fig.1.

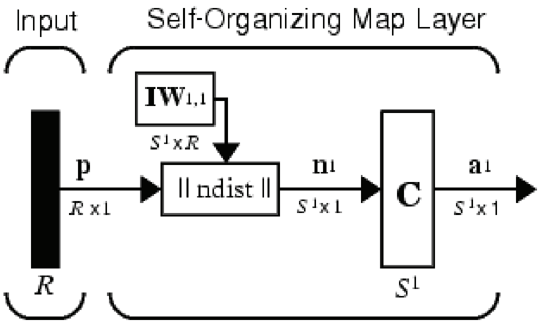

Fig.1. Self-organizing feature map architecture

The input receives consecutively vectors ***p*** with dimensions *Rx1*, where *R=256*, each ***p*** represents a character (or pseudo-character). At first a grayscale images for the separate characters are used which we binarize using Otsu algorithm. Afterwards we resize the binary symbol to *256x256* pixels with nearest neighbour interpolation. Then thin or thicken "the width of the pen" used to *12* pixels using erosion or dilation *l* times:

$$l = \frac{|t-12|}{2}, \tag{1}$$

where *t* is the initial "width of the pen" found as a maximal value in a histogram representing all "the widths of the pen" (number of ones closed between zeros for each line in the image if an inverted one is used) in the resized binary image. This preprocessing step is needed to make characters as invariant as possible for further forming the learning extract. Finally we divide the result image to *16x16* pixels sized blocks. The number of pixels (ones for example) corresponding to character in each block is calculated and the normalized value (in range *[0,1]*) forms a respective component for the input vector ***p***. The blocks are passed from left to right and from top to bottom.

We use the original Kohonen learning rule defined as:

$$\begin{aligned} w_i(q) &= w_i(q-1) + \alpha[p(q) - w_i(q-1)] = \\ &= (1-\alpha)w_i(q-1) + \alpha p(q) \end{aligned}, \tag{2}$$

[1]Ivo R. Draganov is with the Faculty of Communications and Communications Technologies, Technical University, Kliment Ohridski 8, 1000 Sofia, Bulgaria, E-mail: idraganov@abv.bg

[2]Antoaneta A. Popova is with the Faculty of Communications and Communications Technologies, Technical University, Kliment Ohridski 8, 1000 Sofia, Bulgaria, E-mail: antoaneta.p@komero.net

where *i* denotes a single neuron; *w* – the neuron weight; *q* – current step; *α* – the learning rate; *N_i(d)* is the neighbourhood around a wining neuron from which weights of all neurons *j* should be updated. Here *d* is the radius of the neighbourhood:

$$N_i(d) = \{ j, d_{ij} \le d \} . \tag{3}$$

$IW^{1,1}$ from Fig.1 is a weight matrix for all the *S* input neurons (equal to the number of classes/subclasses for all the characters) with *R* weights each. $\mathbf{n}_i^1$ is the result from finding the distance from ***p*** to *i*-th column of ***$IW^{1,1}$***:

$$\mathbf{n}_i^{\,1} = -\left\| \mathbf{IW}_i^{\,1,1} - \mathbf{p} \right\| , \tag{4}$$

which is passed to competitive layer ***C***, the output from which $\mathbf{a}^1$ – a *Sx1* vector containing one component equal to *1*, indicating the recognized character and all the other components equal to *0*.

Given the preprocessing steps from above it is clear that our SOFM operates in 256-dimensional space, for each dimension of which a range of *0* to *1* is set. We use midpoint initialization for all the neurons' weights (0.5). The learning rate *α* and neighbourhood distance *d* are altered through the learning procedure. It lasts for a given number of steps *Q*. The neighbourhood distance starts as the maximum distance d_{max} (calculated at first step) between two neurons, and decreases to preliminary defined end neighbourhood distance d_{min}. Similarly the learning rate starts at some initial value α_{init} and decreases until it reaches some smaller one α_{end} – both are preliminary set. As *d* and *α* decrease, the neurons of the network typically order themselves in the input space with the same topology in which they are ordered physically.

The neocognitron performs classification of input through a succession of functionally equivalent stages. Each stage extracts appropriate features from the output of the preceding stage and then forms a compressed representation of those extracted features. Fig.2 shows the structure of the neocognitron for a case of recognizing *10* objects (e.g. digits).

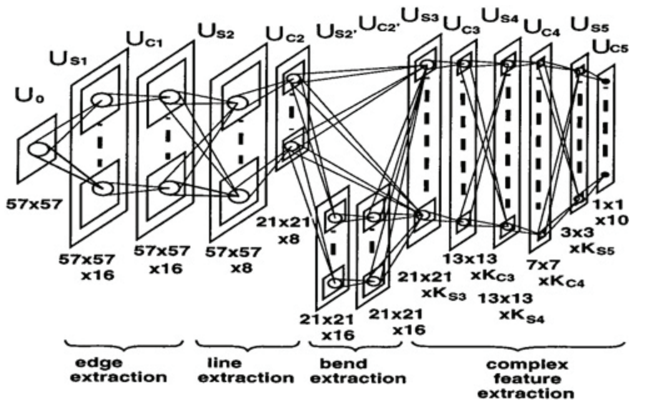

Fig.2. Structure of a neocognitron for *10* objects

The feature extraction is performed by arrays of *S*-cells that have been trained to respond to certain features that characterize the input patterns U_0 as it is seen from Fig.2. After training the weight vector of an *S*-cell is equal to the sum of the inputs that have appeared within its receptive field. Each *S*-cell also receives an inhibitory signal proportional to the *root mean square* (*rms*) activity present in its receptive field. The behaviour of an S-cell can be mathematically formulated as a function *φ(.)* of a cell's activation:

$$o(\mathbf{x}) = r_l \varphi(a(\mathbf{x})) , \tag{5}$$

$$a(\mathbf{x}) = \frac{1 + \mathbf{x}^T \mathbf{w}}{1 + \frac{r_l}{1 + r_l} . b_l . rms(\mathbf{x})} - 1 , \tag{6}$$

where ***x*** is the vector of activities present at the receptive field input, ***w*** is the vector of weights learned by the *S*-cell and r_l is the selectivity parameter found by a separate closed-form training algorithm which we will not discuss. The *rms* activity of the input to the *S*-cell is defined by:

$$rms(x) = \sqrt{\sum_{i=1}^{N} \mathbf{c}_i \mathbf{x}_i^2} , \tag{7}$$

where the vector $\mathbf{c}=[c_1,\ldots,c_N]^T$ describes a Gaussian kernel that serves to accentuate inputs towards the centre of the cell's receptive field, as well as implementing the arithmetic mean of the inputs. b_l is a factor set by the learning rule to maximize the cell's response to any training feature.

Interesting here is the *S*-cell transfer function considered influencing the final recognition accuracy [5]. Originally Fukushima used a threshold linear function of the form:

$$\varphi_{ThreshLin}(a) = \begin{cases} 0, a < 0 \\ a, 0 \le a \end{cases} , \tag{8}$$

but here we will use another two transfer functions in our experiments – the more simple threshold function:

$$\varphi_{Thresh}(a) = \begin{cases} 0, a < 0 \\ 1, 0 \le a \end{cases} , \tag{9}$$

and the sigmoid transfer function:

$$\varphi_{Sig}(a) = \frac{1}{1 + e^{-\beta a}} . \tag{10}$$

Fig.3 shows the structure of an *S*-cell along with the described transfer functions.

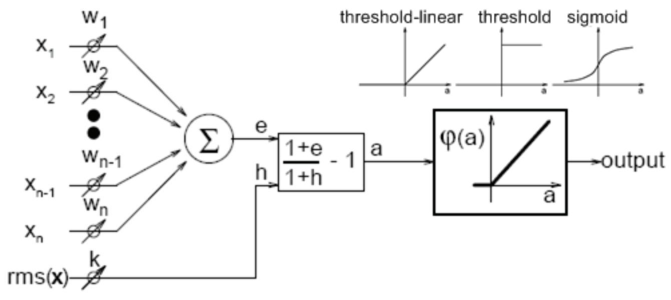

Fig.3. *S*-cell structure with different transfer functions

C-cells compress the representation – the input to a *C*-cell from its receptive field is a subsampling of the activity in the preceding *S*-plane. By susbsampling this activity, a compressed representation of the *S*-plane output is obtained. *C*-cell also blur the activations of the preceding *S*-planes by performing a weighted sum of inputs, this time using fixed weights that describe a Gaussian kernel. If we denote the subsampled input as a vector ***x*** and the Gaussian kernel as ***w*** then the activation of a *C*-cell in Fukushima's original description of the neocognitron can be written as:

$$a(x) = \mathbf{w}^T \mathbf{x}. \quad (11)$$

This weighted mean is then passed through a transfer function that limits the output of the *C*-cell to *[0,1)*:

$$\psi_{Mean}(a) = \frac{a}{1+a}. \quad (12)$$

Here is the second difference from the very original Fukushima's neocognitron. Again in [5] is noted that blurring of *S*-plane activity by the *C*-cells is important in allowing the neocognitron to be tolerant of a considerable degree of input distortion. Thus a ranked order filter is incorporated in the very structure of the *C*-cell. The output of the modified *C*-cell is given by:

$$\psi_{Max}(x) = \max_i x_i w_i, \quad (13)$$

where $\boldsymbol{x}=[x_1,\ldots,x_N]^T$ is again the subsampled input vector and $\mathbf{w}=[w_1,\ldots,w_N]^T$ is the Gaussian kernel. We refer to Eq. (13) as a weighted max operation.

III. Experimental Results

All the experiments are implemented on an IBM® compatible PC® with Pentium®4 Processor working at 1.5 GHz, 256 MB RAM. The operating system is MS® Windows® XP® with SP2 and the working environment is Matlab® 7.0.1 SP1.

Our database collected from different authors contains images of 14502 lower and upper case Latin characters and digits (8-bpp grayscale, 150 dpi). From them we use 8152 preprocessed (filtered by median and range filters) images of lower case characters to train the networks and afterwards 600 additional images which form the test set (for the recognition phase) although a set of 400 is considered to be enough [5].

We use the following parameters for the accuracy comparison:

- relative number of correctly recognized characters:

$$r_c = \frac{m_c}{m}.100, \%, \quad (14)$$

where *m* is the number of passed characters for recognition (*m=600*); m_c – the number of correctly recognized ones;

- relative number of wrong classified characters:

$$r_w = \frac{m_w}{m}.100, \%, \quad (15)$$

where m_w represents the absolute number of wrongly recognized characters;

- relative number of non-classified characters:

$$r_n = \frac{m_n}{m}.100, \%, \quad (16)$$

where m_n represents the absolute number of non-classified (rejected) characters.

Our first experiment includes finding the optimal value for the minimum learning rate α_{min} of the SOFM. We set $\alpha_{max}=0.9$ and d_{min} = 0.001 – a midrange value for the typical working interval [0.000001,1] for the particular case. Before recognition phase we train the network for 10 epochs with 26 classes with 5 subclasses for each present. The so called 'gridtop' topology is used along with Euclidean distance. Rejection criterion is $d>10^{-6}$. The results are shown in Table I.

TABLE I
FINDING THE OPTIMAL MINIMAL LEARNING RATE FOR SOFM

α_{min} / Recognition	0.1	0.3	0.6	0.8
r_c, %	26.09	47.83	60.87	56.52
r_w, %	56.52	34.78	21.74	26.09
r_n, %	17.39	17.39	17.39	17.39

The graphical representation of the results from Table I is given in Fig.4.

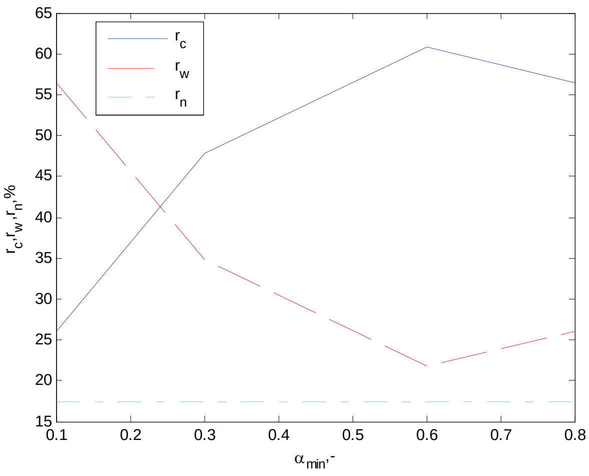

Fig.4. r_c, r_w, r_n as functions of α_{min} for SOFM

Given the above results we choose to use $\alpha_{min}=0.6$ as an optimal value because the number of correctly recognized characters is maximal for it. Thus the next experiment concerns the radius of the neighborhood for fixed α_{min} – we decrease d_{min} from *1* to 10^{-5} by a factor of *10*. Again our SOFM is trained for 10 epochs, 26 classes with 5 subclasses. The results are shown in Table II.

TABLE II
FINDING THE OPTIMAL NEIGHBOURHOOD DISTANCE FOR SOFM

d_{min} / Recognition	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
r_c, %	39.13	56.52	60.87	56.52	52.17	47.82
r_w, %	60.87	26.09	21.74	26.09	30.43	34.78
r_n, %	0.00	17.39	17.39	17.39	17.39	17.39

The graphical representation of the results from Table I is given in Fig.4.

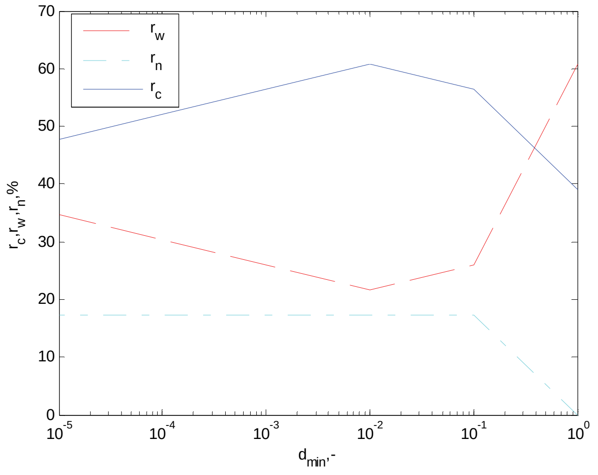

Fig.5. r_c, r_w, r_n as functions of α_{min} for SOFM

Now we have the optimal $\alpha_{min}=0.6$ and $d_{min}=0.01$. How the number of epochs of training the SOFM affects the recognition process, for 26 classes with 5 subclasses, is shown in Table III and graphically in Fig. 6. Training time consumption order is as follows – for 10 epochs – a few minutes, for 100 – about an hour and for 1000 – more than 8 hours.

TABLE III
INFLUENCE OF THE NUMBER OF EPOCHS FOR SOFM

Epochs / Recognition	10	100	1000
r_c, %	60.87	73.91	82.61
r_w, %	21.74	21.74	17.39
r_n, %	17.39	4.35	0.00

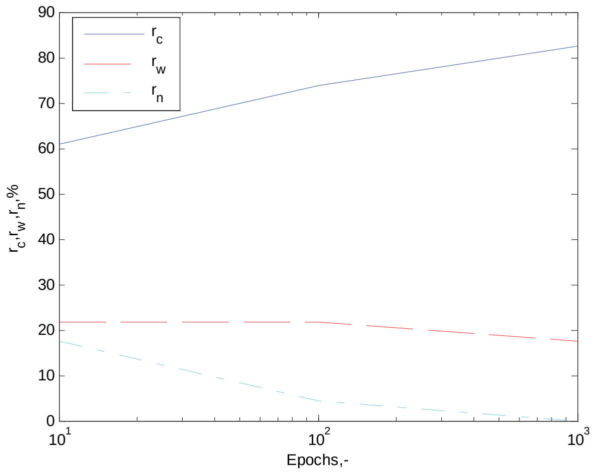

Fig.6. r_c, r_w, r_n as functions of number epochs for SOFM

The results for the neocognitron using the same training and test data are shown in Table IV. The only parameter we change is the *S*-cell transfer function as seen. The values of the other parameters for this experiment are given in [5].

TABLE IV
RECOGNITION ACCURACY FOR THE NEOCOGNITRON

$\varphi(.)$ / Recognition	Thresh	ThreshLin	Sigmoid
r_c, %	34.75	54.50	65.50
r_w, %	23.00	23.75	34.50
r_n, %	42.25	21.75	0.00

IV. CONCLUSION

It is obvious that the SOFM is equivalent to the neocognitron concerning recognition accuracy when a reasonable amount of time is spent for training (*2-10* epochs) and preliminary processed high quality test data is used (small noise levels, shifts, etc.). The last confirms the need and the important role of qualitative preprocessing.

For both - learning rate and neighborhood distance for SOFM should be found appropriate values during training as they highly affect the final recognition accuracy. As neighborhood distance decreases and inclines to *0*, the SOFM behaves itself more like a simple competitive neural network which is explainable by its working principle.

As for the *S*-cell transfer function for the neocognitron it is visible that the sigmoid one is the best option from the three tested ones. In the other two cases the neocognitron drops behind SOFM as for the recognition accuracy.

Depending of the circumstances both of the networks can be applicable in a complete handwriting recognition system.

REFERENCES

[1] R. Bozinovic and S. Srihari, “Off-Line Cursive Script Word Recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 1, pp. 68-83, 1989.
[2] O. Trier, A. Jain and T. Taxt, “Feature Extraction Methods for Character Recognition – A Survey”, Pattern Recognition, vol. 29, no. 4, pp. 641-662, 1996.
[3] T. Kohonen, “The Self-Organizing Map”, Proceedings of IEEE, vol. 78, no. 9, pp. 1467-1480, 1990.
[4] K. Fukushima, “Neocognitron: A Self Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position”, Biological Cybernetics, vol. 36, no. 4, pp. 193-202, 1980.
[5] K. Fukushima and N. Wake, “Handwritten Alphanumeric Character Recognition by the Neocognitron”, IEEE Transactions on Neural Networks, vol. 2, no. 3, pp. 355-365, 1991.
[6] A. Bekiarski and L. Batchishing, “Choice of structure features for recognition of mongolian letters”, Electrotechnics and Electronics, XXXII, no. 1-2, pp. 43-46, 1997.