Pattern Recognition Letters 13 (1992) 387-397
North-Holland

May 1992

# Neural network, self-organization and object extraction

Ashish Ghosh and Sankar K. Pal

*Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035, India*

Received 6 February 1990
Revised 13 November 1990

*Abstract*

Ghosh, A., and S.K. Pal, Neural network, self-organization and object extraction, Pattern Recognition Letters 13 (1992) 387-397.

Algorithms for object extraction nsing a nenral network are proposed. A single nenron (processor) is assigned here to every pixel for its operation in order to implement the concept of self-organized featnre mapping. Both global and local information have been used as input feature. Statistieal criteria for obtaining the optimal output are snggested. Theoretical proof for the convergence of the algorithms is also given. The algorithms are found to work well even foı noisy input.

*Keywords.* Neural network, self-organization, object extraction, image segmentation, image processing.

## 1. Introduction

Neural net (NN) models are specified by the net topology, node characteristics and learning rules. The rules specify an initial set of weights and indicate how weights should be adapted during use to improve performance. A number of NN models is proposed by several authors [1-9]. Some of the important net models are that of Hopfield [6], Kohonen [8,9], Carpenter and Grossberg [16], single layer and multilayer perceptrons [3,4], and Hamming net [4].

Neural net based information processing is a new and promising field of research work. There have been some attempts [3, 4, 7, 9, 16] to use neural nets for pattern recognition, natural language processing, speech recognition and image processing problems in order to have output in real time.

The present work is an attempt to use Kohonen's concept of self-organized feature mapping for object extraction from an image. The term 'self-organization' refers to the ability to learn the structure of the input data even when there is no information about it. Self-organization is a basic principle of sensory paths of the human visual system. The feature mapping algorithm considers a set of neurons representing a class and the same input is provided to all the neurons. The present algorithm, on the other hand, considers the number of neurons equal to the number of pixels in an input image and the input differs from neuron to neuron.

*Correspondence to*: S.K. Pal, Electronics and Communication Sciencs Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035, India.

It is to be mentioned here that the work of Chua and Yang [10] also considers different input for different neurons, but the working principle (weight adjustment/learning) and the output functions are different. The principle used in the present algorithm is more general as compared to that in [10], because they assumed the output state of the neurons as $-1$ or $+1$, whereas in the present case the output may be continuous.

## 2. Kohonen's model of self-organized feature mapping

Kohonen's self-organizing network consists of a single layer of neurons as shown in Fignre 1. The neurons, however, are highly interconnected within the layer as well as the outside world. The above mentioned network functions as a vector quantizer by adjusting weights depending on the input. Let the signal

$$U = \{u_1, u_2, \ldots, u_n\}' \in \mathbb{R}^n$$

be connected as input to all the processing units. Let the length of the input vector be fixed (to some constant length). Let the unit (processor/neuron) $i$ have the weights

$$W_i = \{w_{i1}, w_{i2}, \ldots, w_{in}\}'$$

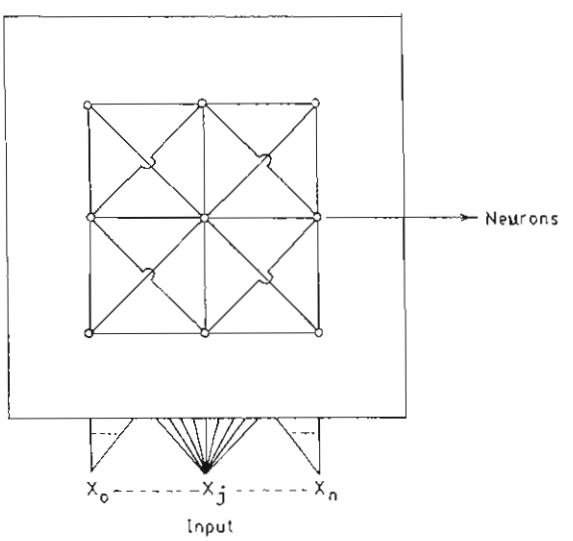


Figure 1. Kohonen's model of two-dimensional neural network used to form self-organizing feature map [4].

where $w_{ij}$ is the weight of the $i$th unit from the $j$th component of the input. The $i$th unit then forms the discriminant function

$$\eta_i = \sum_{j=1}^{n} w_{ij} u_j = W_i \cdot U \tag{1}$$

where $\eta_i$ is the output of that unit. A discrimination mechanism will then detect the maximum of $\eta_i$. Let

$$\eta_k = \max_i \{\eta_i\}. \tag{2}$$

For the $k$th unit and all of its neighbours (within a specified radius of $r$, say) the following adaptation rule

$$W_k(t+1) = \frac{W_k(t) + \alpha \cdot U(t)}{\|W_k(t) + \alpha \cdot U(t)\|} \tag{3}$$

is applied. Here the variables have been labelled by a discrete time index $t$, $\alpha$ $(0 < \alpha < 1)$ is a gain parameter that decreases with time and the denominator is the length of the weight vector (which is used only for normalization of the weight vector). Note that the process does not change the length of $W_k$, rather rotates $W_k$ towards $U$.

The concept can be written in algorithmic form as follows [4].

*Step 1. Initialize weights*
Initialize weights of the neurons to random values.
*Step 2. Present new input*
*Step 3. Compute distance to all nodes*
Compute distance $d_i$ between the input and each output node $i$ using

$$d_i = \sum_j (u_j(t) - w_{ij}(t))^2.$$

*Step 4. Select output node with minimum distance*
Select node $i^*$ as the output node with minimum $d_i$.
*Step 5. Update weights to node $i^*$ and neighbours*
using equation (3).
*Step 6. Repeat by going to Step 2.*

The basic points to be noted here are as follows:
(i) Continuous valued inputs are presented.
(ii) After enough input vectors have been presented, weights will specify clusters.

(iii) The weights will organize in such a fashion that the topologically close nodes/neurons/processors will be sensitive to inputs that are physically close.

(iv) The weight vectors are normalized to have constant length.

(v) A set of neurons (may differ from cluster to cluster and from network to network) is allocated for a cluster.

The algorithm for feature map formation requires a neighbourhood to be defined around the neurons. The size of the neighbourhood gradually decreases with time as shown in Figure 2. It is to be noted that, since the maximum length of the input and weight vectors are fixed, the distance minimization and the dot product maximization criteria are basically the same.

## 3. Self-organization and object extraction

Extraction of an object from a given image can be done either by gray level thresholding or by pixel classification (region growing). There exist a number of such algorithms both classical and fuzzy set theoretic [11–13]. In this section object extraction algorithms will be formulated using a neural network (Kohonen's model).

From the previous section it is seen that the self-organization and feature mapping characteristic can provide the structural information of the input data set, without having any a priori knowledge on it. In the context of pattern recognition and image processing, this can be viewed as clustering and/or segmenting an image space into meaningful regions, say object and background.

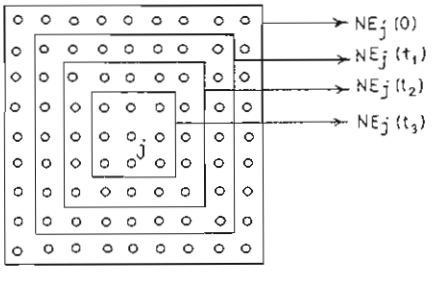


Figure 2. Topological neighbourhoods at different times for the neuron $j$ $(0<t_1<t_2<t_3)$ [4].

In the following subsections, we will explain the way how the concept of self-organization can be exploited for object extraction from a gray tone image. Before doing this, let us represent the gray level image in terms of a neural network.

### 3.1. *Image processing in the notion of neural networks*

An $L$ level $M\times N$-dimensional image $X(M\times N)$ can be represented by a neural net $Y(M\times N)$, having $M\times N$ neurons arranged in $M$ rows and $N$ columns (Figure 1). Let the neuron in the $i$th row and $j$th column be represented by $y(i, j)$ and let it take input from the global/local informatiou of the $(i, j)$th pixel $x(i, j)$ having gray level intensity $x$.

It has already been mentioned that in the feature mapping algorithm, the maximum length of the input and weight vectors is fixed. Let the maximum length for each component of the input vector be unity. To keep the value of each component of the input $\leqslant 1$, let us apply a mapping

$$f: [l_{\min}, l_{\max}] \to [0, 1].$$

Here $l_{\max}$ and $l_{\min}$ are the highest and the lowest gray levels available in the image. The components of the weight vectors are chosen as random numbers in $[0, 1]$.

The output (activation level) of a neuron $y(i, j)$ is a function of both input and the weight vector. The function may be linear or nonlinear depending on the problem at hand. The output status of neuron $y(i, j)$ then governs the processed image output. For example, consider the case of extracting edges (or regions, i.e., segmentation). Here, the output value of a neuron determines whether the pixel $x(i, j)$ is a member of the edge pixel subset (or a region) or not. Similarly, for image enhancement, it reflects the enhanced value of pixel intensity.

### 3.2. *Formulation of object extraction algorithm*

Let us consider the case of object-backgronnd classification in an image. Here the input to a neuron $y(i, j)$ is considered to be an $N$-dimensional vector

$$U(i,j) = [u_1(i,j), u_2(i,j), \ldots, u_n(i,j)]'.$$

The weight (to a neuron) is also a vector quantity

$$W(i,j) = [w_1(i,j), w_2(i,j), \ldots, w_n(i,j)]'.$$

Then compute the dot product as

$$d(i,j)_t = U(i,j)_t \cdot W(i,j)_t = \sum_{k=1}^{n} u_k(i,j) \cdot w_k(i,j) \quad (4)$$

where the subscript '$t$' is used to denote the time instant.

Only those neurons for which $d(i,j) > T$ ($T$ is assumed to be a threshold) are allowed to modify their weights along with their neighbours (withiu a specified radius). Considetation of a set of neighbonrs enables one to gtow the region by including those which might have been dropped out because of the randomness of weights. The weight updation procedute is the same as in equation (3). The value of $\alpha$ (equation (3)) decreases with time. In the present case, it is chosen to be invetsely related to the iteration number.

The output (just to check convergence) of the network is calculated as

$$O_t = \sum_{\substack{i,j \\ \text{with } d(i,j)_t > T}} d(i,j)_t. \quad (5)$$

The procedure of updation of the weights is continued until

$$|O_{t+1} - O_t| < \delta \quad (6)$$

where $\delta$ is a preassigned very small positive quantity. At this stage the network is said to have converged (settled). After the network has converged, the pixels $x(i,j)$ (along with their neighbours) with the constraint $d(i,j) > T$, may be considered to constitute a region, say, object.

The network converges for any valne of $T$. The proof is as follows.

*Proof of convergence*

To prove

$$|O_{t+1} - O_t| < \delta$$

where

$$O_t = \sum_{i,j} d(i,j)_t \quad \text{with } d(i,j)_t > T,$$

$T$ is a threshold, and $\delta$ is a preassigned small positive quantity. Here

$$d(i,j)_t = U(i,j)_t \cdot W(i,j)_t.$$

**Proof.** From equation (3)

$$W_{t+1} = \frac{W_t + \alpha U_t}{\|W_t + \alpha U_t\|}.$$

From above it is evident that $\|W_{t+1}\| = 1$, $\forall t$. It is also given that $\alpha$ decreases as $t$ increases and for the present case $U$ is constant with tespect to time. So,

as $t \to \infty$, $\alpha \to 0$.

Hence

$$\lim_{t \to \infty} W_{t+1} = \frac{W_t}{\|W_t\|} = W_t, \quad \text{as } \|W_t\| = 1.$$

So, with time thc updating of the weights converges. Since $U$ is not a function of time,

$$d_t = W_t \cdot U$$

also converges as $t \to \infty$. So

$$O_t = \sum_{i,j} d(i,j)_t$$

will become constant as $t \to \infty$.

So $|O_{t+1} - O_t|$ will becoine less than $\delta$ (any preassigned small positive quantity). Hence the proof. □

### 3.3. Selection of optimal T

In the above section, a thteshold $T$ in the $W \cdot U$ plane was assumed to decide whether to update (or not to do so) the weights of a neuron; i.e., the threshold $T$ divides the neuron set into two gtoups (thereby the image space into two regions, say object and background). The weights of only one group are updated. For a particulat threshold the updation procedure continues until it converges. The threshold is then varied. Any one of the following two statistical criteria can be used for selecting the optimal threshold.

#### 3.3.1. Least square error (LSE) method

As the network is converged, the image space is divided into two groups, one group having $d$

$(=U \cdot W) > T$ (the object pixels), and the other with $d < T$ (the background pixels). Let $m_o$ and $m_b$ be the mean values of the activation levels (input values) of the object and background regions, respectively. Since we are concerned with object/background situation only, we can assnme that $m_o$ and $m_b$ are the ideal (constant) activation levels of the object and background pixels, respectively. Let $N_o$ be the number of pixels in the object region, $N_b$ that in the background region and $U_i$ be the activation level of the $i$tb element in the stable state. Then the total square error is

$$\varepsilon(T) = \sum_{i=1}^{N_o} (U_i - m_o)^2 + \sum_{i=1}^{N_b} (U_i - m_b)^2. \tag{7}$$

The threshold ($T$) for which this error is minimum can be taken as the optimal threshold. This is optimal in the sense that the deviation of the segmented image from ideal scene is minimum.

### 3.3.2. *Correlation maximization method*

In this case the correlation coefficient between the input sequence (image) and the output sequence is maximized to get the optimal output. As the threshold is varied, the output sequence is changed thereby altering the value of the correlation coefficient. When the threshold corresponds to the actual boundary, the initial grouping is close to the expected output and thus the correlation coefficient is maximum for this threshold.

Let $x_i$ aud $y_i$ $(i = 1, 2, \ldots, n)$ be two seqnences with mean values $\hat{x}$ and $\hat{y}$, respectively. Then the correlation coefficient ($r_{xy}$) between these two sequences is defined as

$$r_{xy} = \frac{\mathrm{Cov}(x, y)}{\sigma_x \cdot \sigma_y} \tag{8}$$

where

$$\mathrm{Cov}(x, y) = \frac{1}{n} \sum_i (x_i - \hat{x})(y_i - \hat{y})$$

and

$$\sigma_x^2 = \frac{1}{n} \sum_i (x_i - \hat{x})^2, \qquad \sigma_y^2 = \frac{1}{n} \sum_i (y_i - \hat{y})^2.$$

In the present case, one sequence is the input values of all the neurons. The second sequence contains values either $m_o$ (for all elements belonging to objects) or $m_b$ (for all elements belonging to the background), where $m_o$ and $m_b$ have the same meaning as in the previous case. Let $N_o$ be the number of pixels belonging to the object region and $N_b$ be that to the background. It can easily be seen that the means of the above two sequences are the same. Let the mean of the sequences be $m$. Then the sigma values of the input and the output sequences will be

$$\sigma_{\mathrm{inp}}^2 = \frac{1}{n} \sum_i (U_i - m)^2, \tag{9}$$

$$\sigma_{\mathrm{out}}^2(T) = \frac{1}{n} \left\{ \sum_{i=1}^{N_o} (m_o - m)^2 + \sum_{i=1}^{N_b} (m_b - m)^2 \right\}, \tag{10}$$

$$\mathrm{Cov}_T(\mathrm{inp}, \mathrm{out}) = \frac{1}{n} \left\{ \sum_{i=1}^{N_o} (m_o - m)(U_i - m) + \sum_{i=1}^{N_b} (m_b - m)(U_i - m) \right\} \tag{11}$$

and

$$r(T) = \frac{\mathrm{Cov}_T(\mathrm{inp}, \mathrm{out})}{\sigma_{\mathrm{inp}} \cdot \sigma_{\mathrm{out}}(T)}. \tag{12}$$

### 3.4. *Distinguishing features*

Kohonen demonstrated a self-organized feature mapping criterion [8] for developing a self-supervised classification algorithm considering a set of neurons to represent a class. The numbers of ueurons differ from class to class depending on their probability of occurrence. The same input is given to all the neurons for estimation (learning) of their weights corresponding to a class. The present work on unsupervised classification (i.e., clustering or segmentation), on the other hand, considers the number of neurons the same as that of the pixels and the input is different for different neurons.

The mathematical proof for the convergence of the algorithms is given in Section 3.2. It can further be established logically as follows. The weight updating procedure is the same as that of Kohonen (equation (3)). In Kohonen's model the inputs to all the neurons are the same, whereas in the present case the inputs to different neurons are different.

The weight updating procedure is such that it always brings the randomly assigned weights towards the input. So, even if the inputs to the neurons are not the same, the weight updating procedure will converge thereby settling the network.

In the present case of object extraction it is assumed that the image contains compact object regions. The regions are assumed to have more or less uniform intensities, thereby forming spatial clusters to some extent. So it can be said that if a pixel belongs to an object, then the probability of its neighbours to belong to the same region is very high. Thus at the onset, the input valnes to the neurons in a particular region are more or less similar, but rhe weights may be different, as they are assigned randomly. The formation of the clusters can then be described as follows.

If the weights assigned are such that

$$U_i \cdot W_i > T \quad \forall i$$

in a particular region, then in course of time the weight updating will converge and a spatial cluster will be formed in that region.

On the other hand, if the assigned weights are such that

$$U_i \cdot W_i > T$$

for a few $i$'s in a parricular region, then during updating of the weights of those neurons they will update the weights of their neighbours also and thereby bring them too into the cluster.

But a cluster will not be formed if

$$U_i \cdot W_i < T \quad \forall i,$$

in a region. Since the weights are assigned randomly, the probability of occurrence of such a situation in the practical case is nil. So it is clear that the clusters will be formed and extracted in the proposed technique.

Furthermore, instead oI minimizing the difference between the input and weight vectors, their dot product is maximized here. Since the maximum length of the input and weight vectors is fixed, the two crireria are basically the same.

The above mentioned theory developed has then been used to formulate two algorithms considering different versions of input vector. These are explained below.

*Case 1*

Let the input vector have only one component and consequently the weight vector will have a single component.

Let

$$U(i,j) = \frac{x(i,j) - l_{\min}}{l_{\max} - l_{\min}} \tag{13}$$

where $x(i,j)$ is the gray level of the $(i,j)$th pixel, and $l_{\max}, l_{\min}$ are the maximum and minimum available gray levels of the image. Note that the value of $U(i,j)$ lies in the range $[0,1]$. The weights will be assigned as random numbers in $[0,1]$.

**Algorithm 1**

*Step 1.* Assign input $U(i,j)$ to the neuron $y(i,j)$ using equation (13), for all possible $i$ and $j$.

*Step 2.* Initialize the weights of the neurons by random numbers in $[0,1]$.

*Step 3.* Compute the products (for all neurons) as

$$d(i,j) = U(i,j) \cdot W(i,j).$$

*Step 4.* Select those ueurons for which $d(i,j) > T$ ($T$ is a preassigned threshold, $0 < T < 1$) and update (along with the neighbours) the weights using equation (3).

*Step 5.* Repeat Steps 3-4 until convergence is obtained with respect to equation (6).

*Step 6.* Vary $T$, iterate Steps 2-5 and find the optimum one.

*Case 2 (Algorithm 2)*

In Algorithm 1 the input to a neuron is a scalar quantity, the representative of the gray level information of the pixels. No local information is incorporated. In the present algorithm, the input to each neuron is considered to be a vector quantity having two components, namely, thc gray value of the pixel and the average gray value of its neighbours, mapped onto $[0,1]$. The computational steps of this algorithm are the same as those of Algorithm 1, except that the input has the aforesaid two components.

The self-organizing ability of the network makes it possible to organize the object pixels (ncurons having $d = W \cdot U > T$) together; thereby separating the objects from the background. The local feedback makes the procedure noise insensitive and robust.

## 4. Computer simulation and results

For the present problem, the neighbourhood of a neuron is considered over a $3 \times 3$ window. The value of $\alpha$ (equation (3)) is chosen as

$$\alpha_N = 1/2N,$$

i.e., $\alpha$ at the $N$th iteration is taken as $1/2N$. This ensures $0 < \alpha < 1$ and decrease of $\alpha$ with time. The value of $\delta$ is taken to be 0.001.

The proposed algorithms are implemented and tested on three different images. The images used are that of *Noisy Tank* (Figure 3(a)), *Biplane* (Figure 4(a)) and *Lincoln* (Figure 5(a)). The size of the images is $64 \times 64$.

For the *Noisy Tank* image the optimal outputs were obtained for the thresholds 0.156 and 0.25 by Algorithms 1 and 2, respectively. The corresponding results are shown in Figures 3(b) and 3(c). From the results it is noticed that the object extracted by Algorithm 2 (Figure 3(c)), as expected, is more compact than that by Algorithm 1 (Figure 3(b)). This is because Algorithm 2 takes local information into account which Algorithm 1 does not.

A very recent image segmentation algorithm proposed by Pal and Rosenfeld [14] dealt with the image of the *Noisy Tank*. The algorithm is based on 'fuzzy compactness' [15] minimization. The algorithm takes the geometry of the object into consideration for its extraction. One of the good thresholds obtained by their algorithms was 31. The corresponding segmented version is given in Figure 6 (for comparison). From the segmented

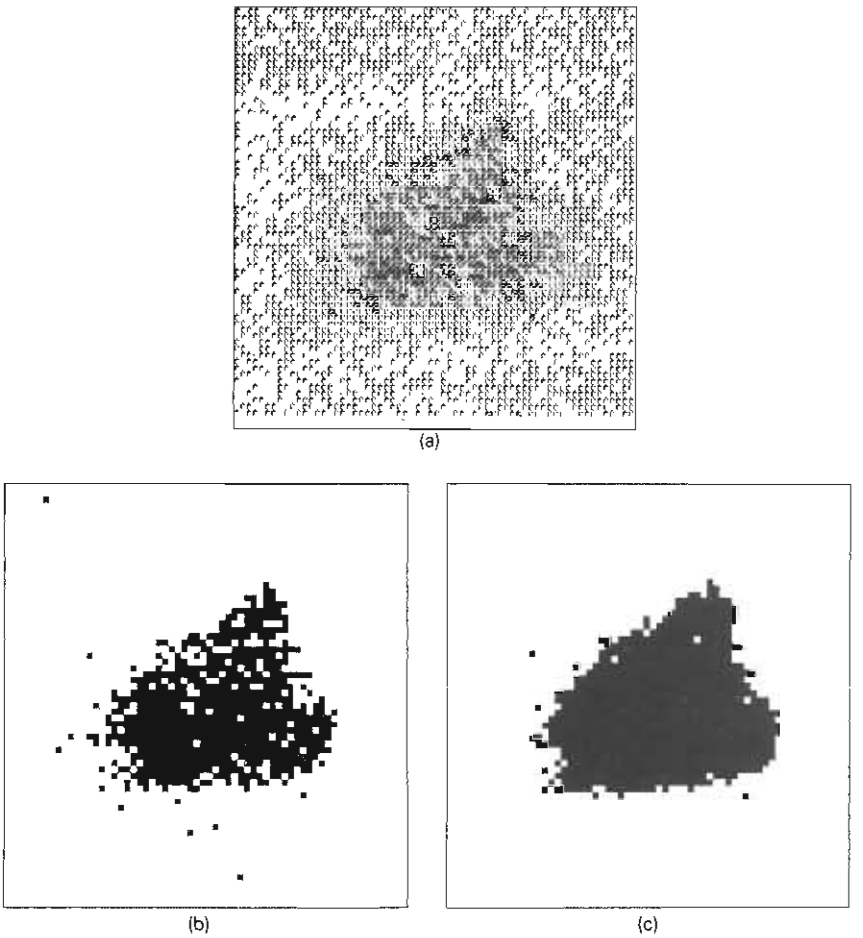


Figure 3. *Noisy Tank* image: (a) input, (b) extracted object by Algorithm 1, (c) extracted object by Algorithm 2.

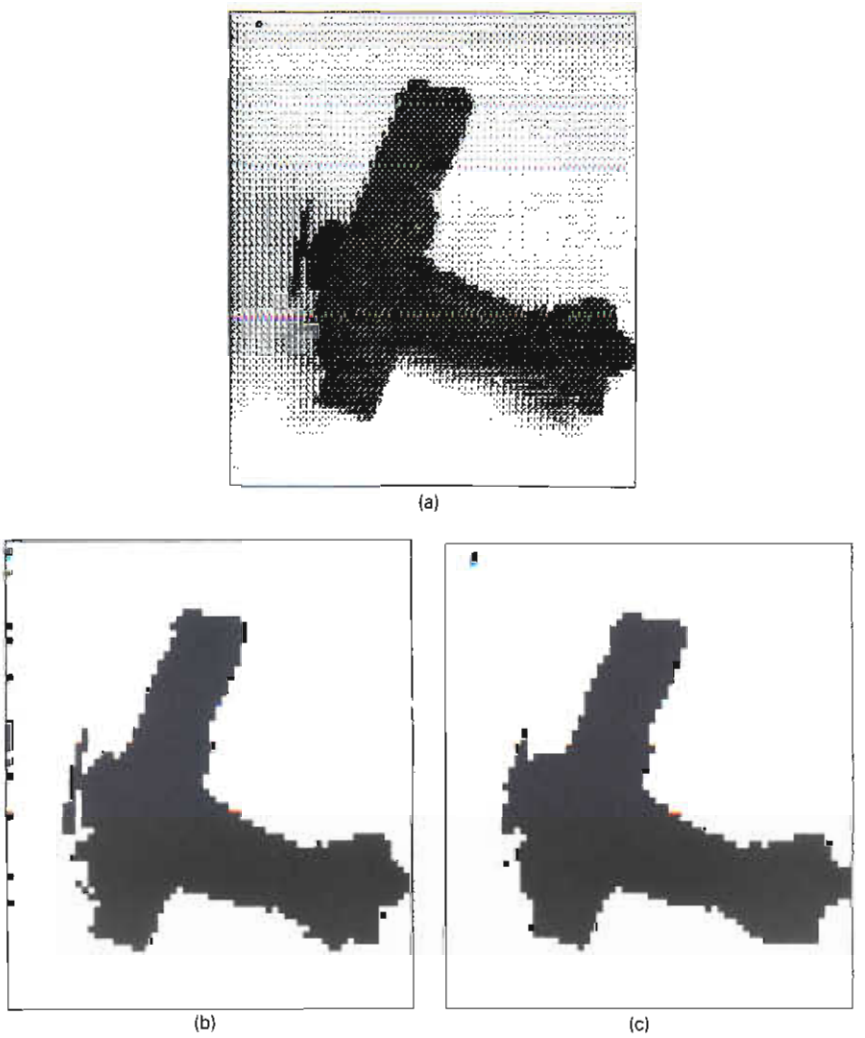


Figure 4. *Biplane* image: (a) input, (b) extracted object by Algorithm 1, (c) extracted object by Algorithm 2.

image (Figure 6) it is very difficult to identify the object as a tank. The results obtained by the proposed algorithms are comparatively better than those of Pal and Rosenfeld [14]. Besides this, the problem of choosing any parameter does not arise in the present case.

The results obtained by Algorithms 1 and 2 for the *Biplane* (Figure 4(a)) image are depicted in Figures 4(b) and 4(c), respectively. In this case the propeller of the plane could not be perfectly extracted by Algorithm 2, which may be due to the incorporation of local information and thereby causing blurring effects. The optimal thresholds for the Algorithms 1 and 2 are 0.25 and 0.50, respectively for this image.

The proposed algorithms are also tried on the image of *Lincoln* (Figure 5(a)). The extracted objects are shown in Figures 5(b) and 5(c) corresponding to Algorithms 1 and 2. The results show that the output obtained by Algorithm 2 preserves continuity much more than that obtained by Algorithm 1. The optimal thresholds are 0.281 and 0.531 for Algorithms 1 and 2, respectively.

From the outputs and the previous discussion it is evident that the outputs of Algorithm 2 are more compact and more noise immune compared to that of Algorithm 1. So, if the noise level of the input is less, one can prefer Algorithm 1, but for more noisy images Algorithm 2 is preferable.

As typical illustration two curves (Figures 7 and 8) showing the variations of the two objective criteria (equations (7) and (12), respectively) with change of threshold (for the *Noisy Tank* image and Algorithm 1) are given. It is to be mentioned here

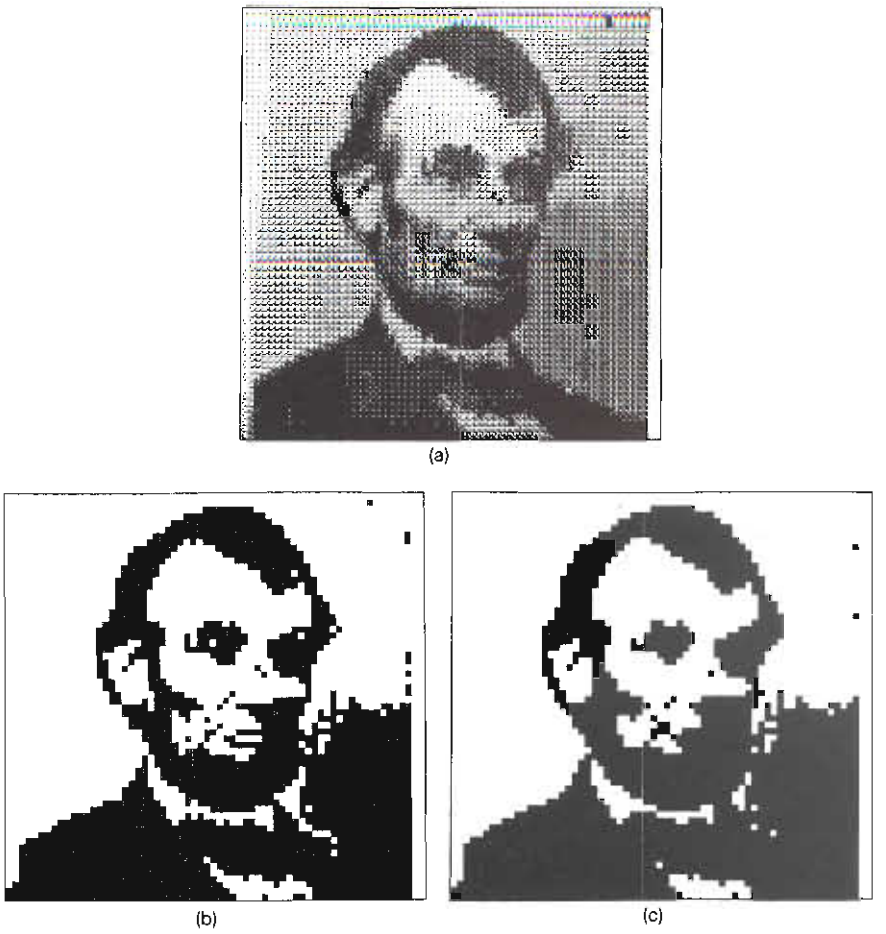


Figure 5. *Lincoln* image: (a) input, (b) extracted object by Algorithm 1, (c) extracted object by Algorithm 2.

that for the images used in the present case, incidentally the optimal thresholds for both the optimality criteria are the same.

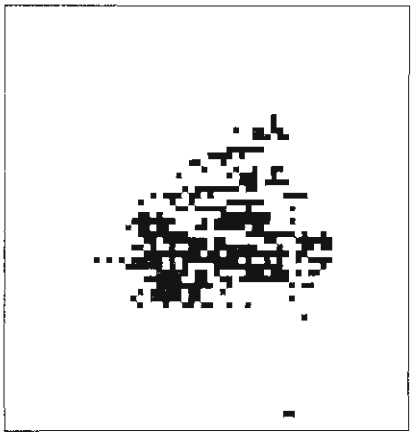

Figure 6. *Noisy Tank* image, thresholded at 31.

## 5. Conclusions and discussion

An attempt has been made here to demonstrate an application of neural net in image processing problems. The concept of 'self-organization' of neural network proposed by Kohonen [2, 8, 9] has been exploited here to extract objects from a scene. An unsupervised classification (clustering) algorithm has been formulated by adopting some modifications in the structure of the net. Unlike assigning a set of neurons to a class (depending on the a priori probability of the class), one neuron is assigned to each pixel. Instead of giving the same input to all the neurons, inputs given to different neurons are different. The convergence of the algorithms has been proved mathematically. Statistical criteria for choosing the optimal output are also suggested.

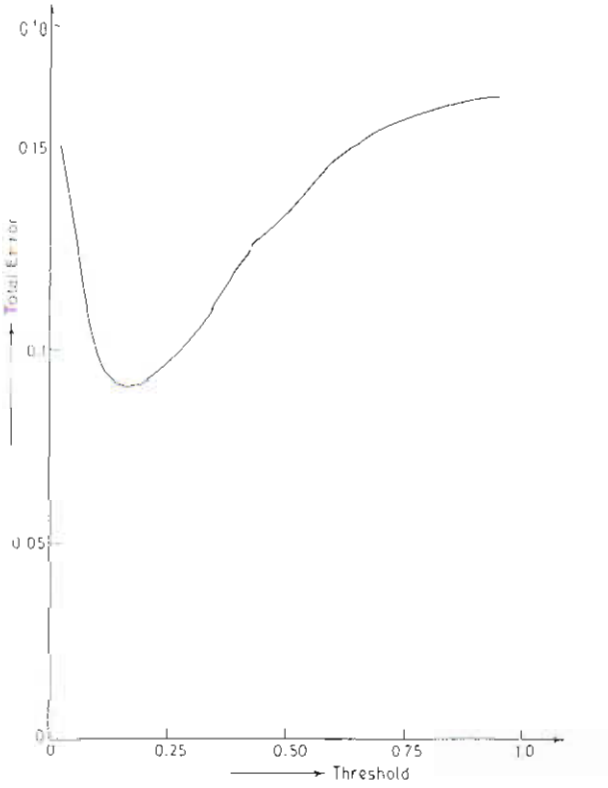


Figure 7. Variation of total square error with threshold.

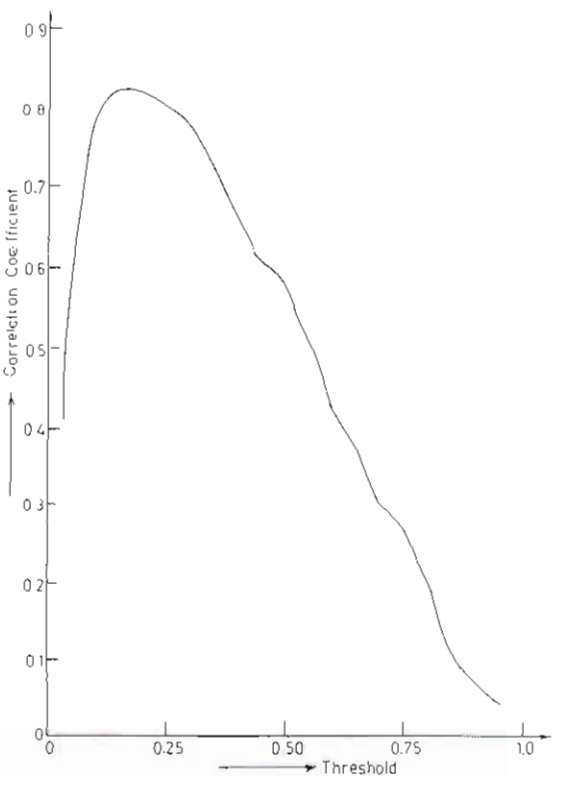


Figure 8. Variation of correlation coefficient with threshold.

The proposed algorithms are parallel in nature and hence are quite fast, and the processed output can be obtained in real time. The results obtained by the proposed algorithms are seen to be quite satisfactory. Their superiority over another algorithm, very recently developed, has also been shown.

In passing it can be mentioned that the algorithm basically is of a relaxation type. The differences between the proposed technique and relaxation labelling are described below.

In relaxation labelling, one compatibility function is defined depending on the a priori information about the membership of the elements in classes. A prior assignment of the elements into different classes is also necessary. The number of probability values (membership in different classes) to be calculated is the same as the number of classes. Given an initial stage, the corresponding converged stage is taken as the final output. There is no provision for choosing the optimal output.

In rhe present algorithm there is no necessity of calculating the compatibility function. As far as class assignment is concerned, only one membership value is sufficient. An optimal output selection criterion is specified in the present technique.

## Acknowledgement

The authors gratefully acknowledge Prof. D. Dutta Majumder for his interest in this work and Mr. S. Chakraborty for drawing the diagrams. Provision of the *Noisy Tank* image data by Prof. A. Rosenfeld is also gratefully acknowledged. Thanks are due to Dr. N.R. Pal for valuable discussions while revising the manuscript. Thanks are also due to the anonymous referees for critical examination and valuable comments.

## References

[1] Fahlman, S.E. and G.E. Hinton (1989). Connectionist architectures for artificial intelligence. *IEEE Trans. Comput.*, 100–109.

[2] Kohonen, T. (1988). An introduction to neural networks. *Neural Networks* 1, 3–16.

[3] Rumelhart, D.E., J. McClelland and PDP Research Group

(1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. I.* MIT Press, Cambridge, MA.

[4] Lippmann, R.P. (1987). An introdnction to compnting with nenral nets. *IEEE ASSP Magazine*, 3–22.

[5] Linsker, R. (1988). Self organisation in perceptual network. *IEEE Trans. Comput. Magazine*, 105–117.

[6] Hopfield, J.J. (1982). Nenral networks and physical systems with emergent computational abilities. *Proc. Nat. Acad. Sci. USA* 79, 2554–2558.

[7] Chna, L.O. and L. Yang (1988). Cellnlar neural networks: applications. *IEEE Trans. Circuits and Systems* 35(10), 1273–1290.

[8] Kohonen, T. (1982). Self organised formation of topologically correct feature maps. *Biol. Cybernet.* 43, 59–69.

[9] Kohonen, T. (1989). *Self Organisation and Associative Memory*. Springer, Berlin, 3rd edition.

[10] Chua, L.O. and L. Yang (1988). Cellular neural networks: theory. *IEEE Trans. Circuits and Systems* 35(10), 1257–1272.

[11] Pal, S.K. and D. Dntta Majnmder (1986). *Fuzzy Mathematical Approach to Pattern Recognition*. Wiley (Halsted Press), New York.

[12] Pal, N.R. and S.K. Pal (1989). Entropic thresholding. *Signal Processing* 16, 97–108.

[13] Rosenfeld, A. and A.C. Kak (1982). *Digital Picture Processing*. Academic Press, New York.

[14] Pal, S.K. and A. Rosenfeld (1989). Image enhancement and thresholding by optimization of fuzzy compactness. *Pattern Recognition Letters* 7, 77–86.

[15] Rosenfeld, A. (1984). Fuzzy geometry of image subsets. *Pattern Recognition Letters* 2, 311–317.

[16] Carpenter, G.A. and S. Grossberg (1987). A massively parallel architecture for a self-organising neural pattern recoguition machine. *Computer Vision, Graphics, and Image Processing* 37, 54–115.