Self-Organization for Object Extraction Using a Multilayer Neural Network and Fuzziness Measures

Ashish Ghosh, Nikhil R. Pal, *Member, IEEE*, and Sankar K. Pal, *Fellow, IEEE*

***Abstract*—The feedforward multilayer perceptron (MLP) with back-propagation of error is described. Since use of this network requires a set of labeled input-output, as such it cannot be used for segmentation of images when only one image is available. (However, if images to be processed are of similar nature, one can use a set of known images for learning and then use the network for processing of other images.) A self-organizing multilayer neural network architecture suitable for image processing is proposed. The proposed architecture is also a feedforward one with back-propagation of errors; but like MLP it does not require any supervised learning. Each neuron is connected to the corresponding neuron in the previous layer and the set of neighbors of that neuron. The output status of neurons in the output layer is described as a fuzzy set. A fuzziness measure of this fuzzy set is used as a measure of error in the system (instability of the network). Learning rates for various measures of fuzziness have been theoretically and experimentally studied. An application of the proposed network in object extraction from noisy scenes is also demonstrated.**

***Index Terms*—Image processing, object extraction, fuzzy sets, neural networks, multilayer perceptron.**

I. Introduction

IMAGE segmentation and object extraction play a key role in image analysis and computer vision. Most of the existing techniques, both classical [1], [2] and fuzzy set theoretic [3]–[5] are sequential in nature and the segmented output cannot be obtained in real time. On the other hand, there are some relaxation [2] type algorithms for which parallel implementations are possible, but robustness of these algorithms usually depends on some prior knowledge about the image, which may be difficult to obtain.

With a view to obtaining the output in real time by parallel processing, recent researchers have been trying to develop neural network (NN) [6]–[11] based information processing systems. Here the basic aim is to emulate the human neural information processing system, thereby making the system artificially intelligent. The approach is highly robust and noise insensitive and hence can be applied even when information is ill defined and/or defective/partial.

Neural networks are designated by the network topology, connection strength between pairs of neurons, node characteristics, and the status updating rules. The neurons operate in parallel, thereby providing output in real time. Since there are interactions among all the neurons, the collective computational property inherently reduces the computational task.

A wide variety of neural network [12]–[15] models has been proposed in the literature. Though they share some common features, they differ in structure and details. Among the different models proposed, the following are very popular.

- Hopfield's model of associative memory;
- Kohonen's model of self-organization;
- Carpenter and Grossberg's model of adaptive-resonance;
- Multilayer perceptron (MLP) of Rumelthart *et al.*

In this article we will be concerned with the MLP type of NN. The MLP requires several input patterns from different classes for learning [16], [17] and acts as a multidimensional pattern discriminator (supervised classifier). If the types of images to be processed are of similar nature (i.e., have some common characteristics) then one can train the network with a set of images and use the trained net on future images. However, if the images do not share some common features and a set of images with known targets (may be synthetic images) is not available, this network, as such, may not be used for image processing.

In the present work a *self-organizing multilayer neural network* architecture in the line of MLP but suitable for image processing is proposed. Unlike MLP, there is no concept of *supervised learning* in the present case. The learning technique employed is *self-supervised*, thereby attaining self-organizing capability. To calculate the error of the system, concepts of fuzzy sets are used. Different mathematical models for calculating the error of the system have also been described in this respect. A comparative study on the rate of learning for these models is also done. Given a single input, the system automatically finds out the structure in the input data by *self-supervision/self-organization*. The final output comes in two types only, one with output status 0 and the other with output status 1.

One application of the proposed network is demonstrated here. The network has been employed to extract objects from noisy environments. The simulation study was done using a synthetic image corrupted by $N(0, \sigma^2)$ additive noise and a real image. The results obtained are found to be quite satisfactory even when the *SNR* is as low as 0.75; where the *SNR* is defined as

$$\text{range of gray levels}/\sigma.$$

Manuscript received April 12, 1992; revised August 3, 1992.

A. Ghosh and S. K. Pal are with the Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 B. T. Road, Calcutta 700 035, India.

N. R. Pal is with the Division of Computer Science, University of West Florida, Pensacola, FL 32514, on leave from the Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 B. T. Road, Calcutta 700 035, India.

IEEE Log Number 9205335.

1063-6706/93$03.00 © 1993 IEEE

The results obtained show that the extracted objects preserve their shapes and boundaries to a great extent, even when the noise level is very high.

II. Fuzzy Sets and Their Measures of Information

In this section a brief overview on the measures of fuzziness of a fuzzy set is given.

A. Definition of a Fuzzy Set

A fuzzy set [3]–[5] A in a space of points $X = \{x\}$ is a class of events with a continuum of grades of membership and is characterized by a membership function $\mu_A(x)$, which associates with each point in X a real number in the interval [0, 1] with the value of $\mu_A(x)$ at x representing the grade of membership of x in A. Formally, a fuzzy set A with its finite number of supports $x_1, x_2, x_3, \cdots, x_n$ is defined as a collection of ordered pairs

$$A = \{(\mu_A(x_i), x_i), i = 1, 2, \ldots, n\}; \tag{1}$$

where the support of A is the subset of X defined as

$$S(A) = \{x | x \in X \,\&\, \mu_A(x) > 0\}. \tag{2}$$

This characteristic function, $\mu_A(x)$, in fact, can be viewed as a weighting coefficient which reflects the ambiguity in a set, and as it approaches unity, the grade of membership of an element in A becomes higher. For example, $\mu_A(x) = 1$ indicates a strict containment of x in A. If, on the other hand, x does not belong to A, $\mu_A(x) = 0$. Any intermediate value would represent the degree to which x could be a member of A.

B. Measures of Fuzziness of a Fuzzy Set

Let us now discuss some measures which give the degree of fuzziness in a fuzzy set A [5], [18]. The degree of fuzziness expresses the average amount of ambiguity in making a decision whether an element belongs to the set or not. Such a measure (I, say) should have the following properties:

1. $I(A) = minimum \quad iff \quad \mu_A(x_i) = 0 \; or \; 1 \; \forall i.$
2. $I(A) = maximum \quad iff \quad \mu_A(x_i) = 0.5 \; \forall i.$
3. $I(A) \geq I(A^*)$, where A^* is a sharpened version of A, defined as
$$\mu_{A^*}(x_i) \geq \mu_A(x_i) \quad \text{if } \mu_A(x_i) \geq 0.5$$
$$\leq \mu_A(x_i) \quad \text{if } \mu_A(x_i) \leq 0.5.$$
4. $I(A) = I(A^c)$, where A^c is the complement set of A.

Several authors [18]–[22] have made attempts to define such measures. A few measures relevant to this work are described here.

1) Index of Fuzziness: The index of fuzziness of a fuzzy set A having n supporting points is defined as

$$\nu(A) = \frac{2}{n^k} d(A, \underline{A}), \tag{3}$$

where $d(A, \underline{A})$ denotes the distance between fuzzy set A and its nearest ordinary set $\underline{A}$. An ordinary set $\underline{A}$ nearest to the fuzzy set A is defined as

$$\mu_{\underline{A}}(x) = \begin{cases} 0 & \text{if, } \mu_A(x) \leq 0.5 \\ 1 & \mu_A(x) \geq 0.5. \end{cases} \tag{4}$$

The value of k depends on the type of distance used. For example, $k = 1$ is used for generalized Hamming distance, where as $k = 0.5$ for Euclidean distance. The corresponding indices of fuzziness are called the linear index of fuzziness $\nu_l(A)$ and the quadratic index of fuzziness $\nu_q(A)$. Thus we have

$$\nu_l(A) = \frac{2}{n} \sum_{i=1}^{n} |\mu_A(x_i) - \mu_{\underline{A}}(x_i)|$$
$$= \frac{2}{n} \sum_{i=1}^{n} \left[\min\{\mu_A(x_i), (1 - \mu_A(x_i))\} \right]. \tag{5}$$

Similarly, for Euclidean distance, we have

$$\nu_q(A) = \frac{2}{\sqrt{n}} \sqrt{\left[\sum_{i=1}^{n} \{\mu_A(x_i) - \mu_{\underline{A}}(x_i)\}^2 \right]}. \tag{6}$$

2) Entropy: Entropy of a fuzzy set as defined by De Luca and Termini [19] is given by

$$H(A) = \frac{1}{n \ln(2)} \sum_{i=1}^{n} \{S_n(\mu_A(x_i))\}, \tag{7}$$

with

$$S_n(\mu_A(x_i)) = -\mu_A(x_i) \ln(\mu_A(x_i)) - \{1 - \mu_A(x_i)\} \ln\{1 - \mu_A(x_i)\}. \tag{8}$$

Another definition of entropy is given by Pal and Pal [22] is

$$H(A) = \frac{1}{n(\sqrt{e} - 1)} \sum_{i=1}^{n} \{S_n(\mu_A(x_i)) - 1\}, \tag{9}$$

with

$$S_n(\mu_A(x_i)) = \mu_A(x_i) e^{1 - \mu_A(x_i)} + (1 - \mu_A(x_i)) e^{\mu_A(x_i)}. \tag{10}$$

All the above measures lie in [0, 1] and satisfy properties 1 through 4.

It may be noted that a definition of *higher order entropy* [$H^r(A)$, $r \geq 1$] for a fuzzy set A (using both logarithmic and exponential gain functions) has recently been reported [23] which takes into account the properties of collection of supports in the set. Equations (7) through (10) correspond to $r = 1$. Here we have considered only $r = 1$, and it will be seen in Section IV that the network structure and the membership function incorporate the properties associated with collection of supports (local information of pixels).

In the following sections we shall use these measures to compute the *error* or *measure of instability* of a multilayer self-organizing neural network. Since understanding the working mechanism of the proposed network requires a knowledge of a multilayer perceptron, a brief description of the same follows.

III. Multilayer Perceptron

A concept central to the practice of *pattern recognition* [17], [16] is that of discrimination. The idea is that a pattern recognition system learns adaptively from experience and does various discrimination, each appropriate for its purpose.

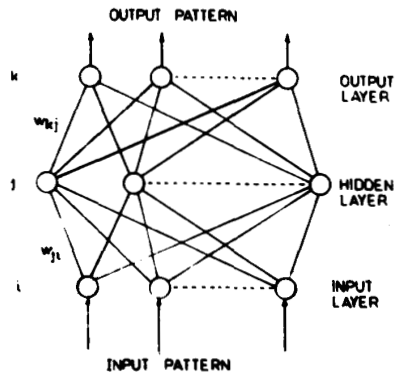


Fig. 1. Schematic representation of multilayer perceptron.

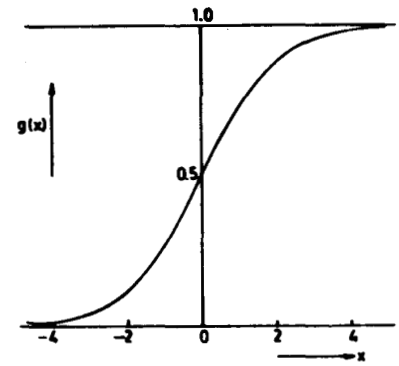


Fig. 2. Sigmoidal activation function.

For example, if only belongingness to a class is of interest, the system learns from observations on patterns that are identified by their class labels and infers a discrimination for classification.

One of the most exciting developments during the early days of pattern recognition was the **perceptron** [24]. This may be defined as a network of elementary processors arranged in a manner reminiscent of biological neural nets which will be able to learn how to recognize and classify patterns in an autonomous manner. In such a system the processors are simple linear elements arranged in one layer. This classical (single-layer) perceptron, given two classes of patterns, attempts to find a linear decision boundary separating the two classes. If the two sets of patterns are linearly separable, the perceptron algorithm is guaranteed to find a separating hyperplane in a finite number of steps. However, if the pattern space is not linearly separable, the perceptron fails and it is not known when to terminate the algorithm in order to get a reasonably good decision boundary. Keller and Hunt [25] attempted to provide a good stopping criterion (may not estimate a good decision boundary) for linearly nonseparable classes by incorporating the concept of fuzzy set theory into the learning algorithm of the perceptron. Thus, a single-layer perceptron is inadequate for situations with multiple classes and nonlinear separating boundaries. Hence the invention of the multilayer network with nonlinear learning algorithms known as the multilayer perceptron (MLP) [8].

A schematic representation of a multilayer perceptron (MLP) is given in Fig. 1. In general, such a net is made up of sets of nodes arranged in layers. Nodes of two different consecutive layers are connected by links or weights, but there is no connection among the elements of the same layer. The layer where the inputs are presented is known as the input layer. On the other hand the output producing layer is called the output layer. The layers between the input and the output layers are known as hidden layers. The output of nodes in one layer is transmitted to nodes in another layer via links that amplify or attenuate or inhibit such outputs through weighting factors. Except for the input layer nodes, the total input to each node is the sum of weighted outputs of the nodes in the previous layer. Each node is activated in accordance with the input to the node and the activation function of the node. The total input (I_i) to the ith unit of any layer is

$$I_i = \sum_j w_{ij} o_j, \tag{11}$$

with o_j the output of the jth neuron in the previous layer and w_{ij} the connection weight between the ith node of one layer and the jth node of the previous layer. The output of a node i is obtained as

$$o_i = f(I_i), \tag{12}$$

where f is the activation function [8]. Mostly the activation function is sigmoidal, with the form (Fig. 2)

$$f(x) = \frac{1}{1 + e^{-(x-\theta)/\theta_0}}. \tag{13}$$

The function is symmetrical around θ, and θ_0 controls the steepness of the function. θ is known as the threshold/bias value.

Initially very small random values are assigned to the links/weights. In the learning phase (*training*) of such a network, we present the pattern $X = \{x_i\}$, where x_i is the ith component of the vector X, as input and ask the net to adjust its set of weights in the connecting links and also the thresholds in the nodes such that the desired output $\{t_i\}$ is obtained at the output nodes. After this, we present another pair of X and $\{t_i\}$, and ask the net to learn that association also. In fact, we desire the net to find a simple set of weights and biases that will be able to discriminate among all the input/output pairs presented to it. This process can pose a very strenuous learning task and is not always readily accomplished. Here the desired output $\{t_i\}$ basically acts as a teacher which tries to minimize the error.

In general, the output $\{o_i\}$ will not be the same as the target or desired value $\{t_i\}$. For a pattern p, the error is

$$E = \frac{1}{2}\sum_i (t_i - o_i)^2, \tag{14}$$

where the factor of one half is inserted for mathematical convenience. The procedure for learning the correct set of weights is to vary the weights in a manner such that the error E is reduced as rapidly as possible. This can be achieved by

moving in the direction of negative gradient of E. In other words, the incremental change for a particular pattern p is

$$\Delta w_{ji} \propto -\frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial I_j}\frac{\partial I_j}{\partial w_{ji}}$$
$$= \eta \delta_j o_i \qquad \text{[from eq. (11)]} \tag{15}$$

with

$$\delta_j = -\frac{\partial E}{\partial I_j} = -\frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial I_j}$$
$$= -\frac{\partial E}{\partial o_j} f'(I_j) \qquad \text{[from eq. (12)]}. \tag{16}$$

As E can be directly calculated in the output layer, for the links connected to the output layer the change in weight is given by

$$\Delta w_{ji} = \eta \left(-\frac{\partial E}{\partial o_j}\right) f'(I_j) o_i. \tag{17}$$

If the weights do not affect the output nodes directly (for links between the input and the hidden layer, and also between two consecutive hidden layers), the factor $\partial E/\partial o_j$ cannot be calculated so easily. In this case we use [8] the chain rule to write

$$\frac{\partial E}{\partial o_j} = \sum_k \frac{\partial E}{\partial I_k}\frac{\partial I_k}{\partial o_j} = \sum_k \frac{\partial E}{\partial I_k}\frac{\partial}{\partial o_j}\sum_i w_{ki} o_i$$
$$= \sum_k \frac{\partial E}{\partial I_k} w_{kj} = \sum_k (-\delta_k) w_{kj}. \tag{18}$$

Hence

$$\Delta w_{ji} = \begin{cases} \eta\left(-\dfrac{\partial E}{\partial o_j}\right) f'(I_j) o_i \\ \eta\left(\displaystyle\sum_j \delta_k w_{kj}\right) f'(I_j) o_i \end{cases} \tag{19}$$

for the output layer and other layers, respectively. In particular, if

$$o_j = \frac{1}{1 + e^{-(\sum_i w_{ji} o_i - \theta_j)}} \tag{20}$$

then

$$f'(I_j) = \frac{\partial o_j}{\partial I_j} = o_j(1 - o_j) \tag{21}$$

and thus we get

$$\Delta w_{ji} = \begin{cases} \eta\left(-\dfrac{\partial E}{\partial o_j}\right) o_j(1 - o_j) o_i \\ \eta\left(\displaystyle\sum_k \delta_k w_{kj}\right) o_j(1 - o_j) o_i \end{cases} \tag{22}$$

for the output layer and other layers, respectively.

It may be mentioned here that a large value of η corresponds to rapid learning but might result in oscillations. A *momentum* term of $\alpha \Delta w_{ji}(t)$ can be added to increase the learning rate and thus expression (15) can be modified as

$$\Delta w_{ji}(t+1) = \eta \delta_j o_i + \alpha \Delta w_{ji}(t), \tag{23}$$

where the term $(t+1)$ is used to indicate the $(t+1)$th time instant, and α is a proportionality constant. The second term is used to specify that the change in w_{ji} at the $(t+1)$th instant should be somewhat similar to the change undertaken at instant t.

A few approaches to object extraction using *neural networks* can be found in [26]–[29]. In addition to these, several authors [30]–[32] have used the multilayer perceptron also for image segmentation/texture discrimination. Blanz and Gish [30] used a three-layer perceptron trained with a set of similar images for pixel classification. On the other hand, Babaguchi *et al.* [31] used a similar concept for histogram thresholding. It is to be noted that all these perceptron-based techniques require a set of images for learning, which may not always be available in real life situations.

IV. Self-Organizing Multilayer Neural Network

It has already been mentioned that an MLP, as it is, cannot be used for image segmentation/object extraction when only one input image is available. In such cases there will not be enough scope of learning (*supervised*). For this type of problem it will be appropriate if we can apply some sort of *self-supervised* [17], [16] learning technique. Besides this, for an $M \times N$ image if we connect each neuron with every other one, the connectivity will be drastically high. In the following subsections we will be describing a new self-organizing [10], [11], [33]–[36] multilayer neural network model whose basic principles will be similar to the previously described MLP, but will differ structurally and functionally to a great extent.

A. Description and Operation of the Network

Before describing the architecture of the network, we need to define a neighborhood system. For an $M \times N$ lattice (L), the dth order neighbor, N_{ij}^d, of any element (i,j) is defined as

$$N_{ij}^d = \{(i,j) \in L\}$$

such that

- $(i,j) \notin N_{ij}^d$
- if $(k,l) \in N_{ij}^d$, then $(i,j) \in N_{kl}^d$.

Different ordered neighborhood systems can be defined considering different sets of neighboring pixels of (i,j). $N^1 = \{N_{ij}^1\}$ can be obtained by taking the four nearest-neighbor pixels. Similarly, $N^2 = \{N_{ij}^2\}$ consists of the eight pixels neighboring (i,j) and so on (as shown in Fig. 3). Due to the finite size of the used lattice (the size of the image being fixed), the neighborhoods of the pixels on the boundaries are necessarily smaller unless a periodic lattice structure is assumed.

1) Architecture: In Fig. 4 we depict the three-layered version of the proposed network architecture. In each and every layer there are $M \times N$ neurons (for an $M \times N$ image). Each neuron corresponds to a single pixel. Besides the *input* and *output* layers, there can be a number of *hidden* layers (more than zero). Neurons in the same layer do not have any connections among themselves. Each neuron in a layer is connected to the corresponding neuron in the previous layer and to its neighbors (over N^d); thus each neuron in layer $i (i > 1)$ will have $|N^d| + 1$ (where $|N^d|$ is the number of pixels in N^d) links to the $(i-1)$th layer. For N^1, a neuron

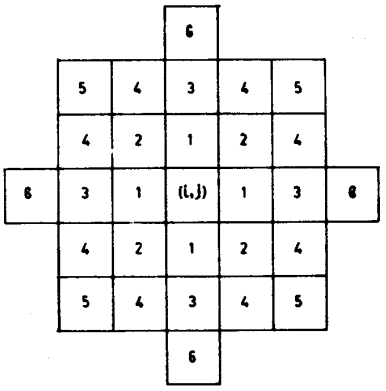


Fig. 3. Neighborhood system N^d.

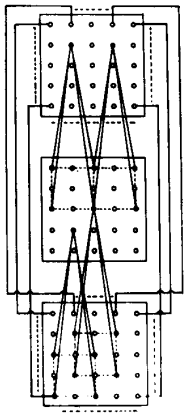


Fig. 4. Schematic representation of Self-organizing multilayer neural network.

has five links, whereas for N^2 nine links will be associated with every neuron. However, for boundary nodes (pixels), the number of links may be less than $|N^d| + 1$. Every neuron in the output layer is also connected to the corresponding neuron in the input layer. It may be noted that this architecture differs from the standard MLP in two major points:

- the distribution of links and
- the feedback connection from the output layer to the input layer.

2) Initialization: The input to a neuron in the input layer is given as a real number in [0, 1] which is proportional to the gray value of the corresponding pixel. Since we are trying to eliminate noise and extract spatially compact regions, all initial weights are set to 1. No external bias is imposed on the weights. Random initialization (of weights) may act as a pseudo noise and the compactness of the extracted regions may be lost. As all the weights are set to unity, the total input (initially) to any node lies in [0, n_l] (where n_l is the number of links a neuron has); hence the most unbiased choice for the threshold value θ (for the input output transfer function, eq. (13) would be $n_l/2$ (the middle most value of the total input range).

3) Operation: The input value (I_i) to any neuron in the ith layer (except the input layer) is calculated using (11). The transfer function f (eq. (13)) is then applied to get the output status of the neurons in this layer. These outputs are then fed as input to the next layer. Starting from the input layer, this way the input pattern is passed on to the output layer and the corresponding output states are calculated. The output value of each and every neuron lies in [0, 1].

Here our intention is to extract spatially compact regions through the process of self-organization using only one noise-corrupted realization of a scene. The way the network is organized, under ideal condition when the image is not noisy, the output status of most of the neurons in the output layer will be either 0 or 1. But due to the effect of noise the output status of the neurons in the output layer usually will be in [0, 1]; thus the status value will represent the degree of brightness (darkness) of the corresponding pixel in the image. Therefore, the output status in the output layer may be viewed to represent a *fuzzy set "bright (dark) pixels."* The number of supports of this fuzzy set is equal to the number of neurons in the output layer. The measure of *fuzziness* of this set, on the global level, may be considered the *error* or *instability of the whole system* as this will reflect the deviation from the desired state of the network (considering properties 1 through 4 of subsection II-B). Thus when we do not have any *a priori* target output value, we can take the fuzziness value as a measure of system error and back-propagate it to adjust the weights (mathematical expressions for this are given later) so that the system error reduces with the passage of time and in the limiting case becomes zero. The error measure, E, can also be taken as a suitable function of a fuzziness measure, i.e.,

$$E = g(I), \tag{24}$$

where I is a *measure of fuzziness* (eqs. (5)–(7), (9)) of the fuzzy set.

After the weights have been adjusted properly, the output of the neurons in the output layer is fed back to the corresponding neurons in the input layer. The second pass is then continued with this as input. The iteration (updating of weights) is continued as in the previous case until the network stabilizes; i.e., the error value (measure of fuzziness) becomes negligible. When the network stabilizes, the output status of the neurons in the output layer becomes either 0 or 1. Neurons with output value 0 constitute one group and those having output value 1 constitute the other group. It may be mentioned here that the scene can have any number of compact regions.

It is to be noted that the system actually does some sort of *self-supervised* learning and thereby *self-organizes* and finds out the structure in the input data. Thus for problems such as clustering [17], [16] where there is no concept of *a priori* teaching, such systems will be of the utmost importance. In self-supervised learning the system learns to respond to "interesting" patterns in the input. In general, such a scheme should be able to form the basis for the development of feature detectors and should discover statistically salient features of the input population. Unlike the other learning methods, there is no *a priori* set of categories into which the patterns are to be classified. Here, the system must develop its own featural

representation of the input stimuli which captures the most salient features of the population of the input pattern. In this context it can be mentioned that this type of learning resembles the biological learning to a great extent.

B. Weight Correction for Different Error Measures

The mathematical derivation for weight updating rules using back-propagation with different fuzziness measures (eqs. (5)–(7) and (9)) are as follows. The derivations are given only for correcting the weights of the links connected to the output layer. For other layers similar expressions, as in the second part of (19), are applicable.

1) Weight Correction for Linear Index of Fuzziness: Let us consider

$$\begin{aligned} E &= g(\nu_l) \\ &= \nu_l, \end{aligned} \tag{25}$$

where the linear index of fuzziness is

$$\nu_l = \frac{2}{n}\sum_j \{\min(o_j, \overline{1-o_j})\}, \tag{26}$$

n being the number of neurons in the output layer. Here

$$-\frac{\partial E}{\partial o_j} = \begin{cases} -\dfrac{2}{n} & \text{if, } 0 \le o_j \le 0.5 \\ \dfrac{2}{n} & 0.5 \le o_j \le 1. \end{cases} \tag{27}$$

Thus from (19) we obtain

$$\Delta w_{ji} = \begin{cases} \eta_1\left(-\dfrac{2}{n}\right) f'(I_j) o_i & \text{if, } 0 \le o_j \le 0.5 \\ \eta_1\left(\dfrac{2}{n}\right) f'(I_j) o_i & 0.5 \le o_j \le 1 \end{cases} \tag{28}$$

or

$$\Delta w_{ji} = \begin{cases} -\eta f'(I_j) o_i & \text{if, } 0 \le o_j \le 0.5 \\ \eta f'(I_j) o_i & 0.5 \le o_j \le 1, \end{cases} \tag{29}$$

where $\eta = \eta_1 \times 2/n$.

2) Weight Correction for Quadratic Index of Fuzziness: Let us choose the error function, E, as

$$\begin{aligned} E &= g(\nu_q) \\ &= \nu_q^2, \end{aligned} \tag{30}$$

where the quadratic index of fuzziness is

$$\nu_q = \frac{2}{\sqrt{n}} \sqrt{\left[\sum_j \{\min(o_j, \overline{1-o_j})\}^2\right]}. \tag{31}$$

Now

$$\nu_q^2 = \frac{4}{n}\left[\sum_j \{\min(o_j, \overline{1-o_j})\}^2\right] \tag{32}$$

and

$$-\frac{\partial E}{\partial o_j} = \begin{cases} \dfrac{4}{n}\{-2o_j\} & \text{if, } 0 \le o_j \le 0.5 \\ \dfrac{4}{n}\{2(1-o_j)\} & 0.5 \le o_j \le 1.0. \end{cases} \tag{33}$$

Thus

$$\Delta w_{ji} = \begin{cases} \eta_2 \dfrac{4}{n}(-2o_j) f'(I_j) o_i & \text{if, } 0 \le o_j \le 0.5 \\ \eta_2 \dfrac{4}{n}\{2(1-o_j)\} f'(I_j) o_i & 0.5 \le o_j \le 1.0. \end{cases} \tag{34}$$

In other words,

$$\Delta w_{ji} = \begin{cases} \eta(-o_j) f'(I_j) o_i & \text{if, } 0 \le o_j \le 0.5 \\ \eta\{(1-o_j)\} f'(I_j) o_i & 0.5 \le o_j \le 1.0 \end{cases} \tag{35}$$

where $\eta = \eta_2 \times (4/n) \times 2$.

If we define the target output $\{t_j\}$ as follows:

$$t_j = \begin{cases} 0 & \text{if, } 0 \le o_j \le 0.5 \\ 1 & 0.5 \le o_j \le 1.0, \end{cases} \tag{36}$$

then this fuzziness measure becomes equivalent to the sum of the squared errors.

It may be noted here that $\partial E/\partial o_j$ is not defined when $o_j = 0.5$ for both ν_l and ν_q. Thus, when implementing the algorithm, if o_j takes a value of 0.5 for some node, then movement can be made in any one of the directions by a small amount.

3) Weight Correction for Logarithmic Entropy: Now we consider

$$E = g(H) \tag{37}$$

$$= H, \tag{38}$$

where H is the entropy of a fuzzy set (eq. (7)) defined as

$$H(A) = -\frac{1}{n\ln(2)}\sum_{j=1}^{n}\{o_j \ln o_j + \overline{1-o_j}\ln\overline{1-o_j}\}. \tag{39}$$

Thus

$$-\frac{\partial E}{\partial o_j} = \frac{1}{n\ln(2)}\ln\frac{o_j}{1-o_j}. \tag{40}$$

Here we notice that

$$\text{as} \quad o_j \to 0 \text{ or } 1, \ \text{abs}\left\{\ln\frac{o_j}{1-o_j}\right\} \to \infty$$

whereas

$$\text{as} \quad o_j \to 0.5, \ \text{abs}\left\{\ln\frac{o_j}{1-o_j}\right\} \to 0.$$

For a fuzzy set, we know that the fuzziness value is minimum when the membership values of all the elements are 0 or 1 and maximum when they all are 0.5. So for the network the error is minimum (i.e., the network has stabilized) when all the output values are either 0 or 1 and is maximum when they all are 0.5 (the network is most unstable). From the previous discussion (eqs. (16), (19), and (40)) we notice that abs$(\partial E/\partial o_j)$ is minimum $(= 0)$ when all the output values are 0.5. Thus, if we use the gradient descent search, the rate of learning is minimum at the most unstable state. Hence, to expedite the learning, it is desirable to make the weight correction large when the network is most unstable (i.e., when all the output values are 0.5). In other words, for a neuron, the weight correction for its links should be maximum when its output status is very close to 0.5 and is minimum when its

output status is close to 0 or 1. This can be achieved by taking (refer to eq. (19), first part)

$$\Delta w_{ji} \propto -\frac{\frac{\partial E}{\partial o_j}}{\left|\frac{\partial E}{\partial o_j}\right|^q}, \qquad q > 1 \tag{41}$$

i.e.,

$$\Delta w_{ji} = -\eta_3 \frac{\frac{\partial E}{\partial o_j}}{\left|\frac{\partial E}{\partial o_j}\right|^q} f'(I_j) o_i, \qquad q > 1, \tag{42}$$

where $|\partial E/\partial o_j|$ represents the magnitude of the gradient.

Such a choice for Δw_{ij} does not violate the necessary requirements for gradient descent search. In any gradient descent search, the gradient vector determines the direction of movement, while the quantum of movement is controlled by a scalar multiplier which is generally preselected by the user. The magnitude of the gradient vector is not very important as far as the gradient descent search is concerned. In the present case since we are changing only the magnitude of movement, keeping the direction same, it is also equivalent to a scalar multiplication. However, if the scalar multiplier is too small, search would be slow; if it is too large, it may result in oscillation.

Henceforth we will be considering only $q = 2$. When $q = 2$, for the logarithmic entropy we get

$$\Delta w_{ji} = -\eta_3 \frac{1}{\frac{\partial E}{\partial o_j}} f'(I_j) o_i = \eta_3 (n \ln 2) \frac{1}{\ln \frac{o_j}{1-o_j}} f'(I_j) o_i$$
$$= \eta \frac{1}{\ln \frac{o_j}{1-o_j}} f'(I_j) o_i \tag{43}$$

with $\eta = \eta_3 \times (n \ln 2)$, i.e.,

$$\Delta w_{ji} = \begin{cases} -\eta \dfrac{1}{\ln \frac{1-o_j}{o_j}} f'(I_j) o_i & \text{if, } 0 \le o_j \le 0.5 \\ \eta \dfrac{1}{\ln \frac{o_j}{1-o_j}} f'(I_j) o_i & 0.5 \le o_j \le 1.0. \end{cases} \tag{44}$$

The expression of Δw_{ji} has been divided into two parts in order to preserve the analogy with that of the index of fuzziness.

4) Weight Correction for Exponential Entropy: If we consider

$$E = g(H) \tag{45}$$
$$= H, \tag{46}$$

where H is the exponential entropy of a fuzzy set (eq. (9)) with

$$H(A) = \frac{1}{n(\sqrt{e}-1)} \sum_{j=1}^{n} \{o_j e^{1-o_j} + (1-o_j) e^{o_j} - 1\}, \tag{47}$$

then

$$\frac{\partial H}{\partial o_j} = \frac{1}{n(\sqrt{e}-1)} \{(1-o_j) e^{1-o_j} - o_j e^{o_j}\}. \tag{48}$$

Applying an argument similar to that of logarithmic entropy, for exponential entropy also we get

$$\Delta w_{ji} = -\eta_4 \frac{1}{\frac{\partial E}{\partial o_j}} f'(I_j) o_i$$
$$= -\eta \frac{1}{(1-o_j) e^{1-o_j} - o_j e^{o_j}} f'(I_j) o_i \tag{49}$$

with $\eta = \eta_4 \times n(\sqrt{e}-1)$. In other words,

$$\Delta w_{ji} = \begin{cases} -\eta \dfrac{1}{(1-o_j) e^{1-o_j} - o_j e^{o_j}} f'(I_j) o_i \\ \qquad \text{if } 0 \le o_j \le 0.5 \\ \eta \dfrac{1}{o_j e^{o_j} - (1-o_j) e^{1-o_j}} f'(I_j) o_i \\ \qquad 0.5 \le o_j \le 1.0 \end{cases} \tag{50}$$

Considering (29), (35), (44), and (50), one can see that in each of the four cases, the expression for Δw_{ji} has a common factor $\eta f'(I_j) o_i$. Therefore, for the sake of comparison of different learning rates the common factor $[\eta f'(I_j) o_i]$ can be ignored. The remaining part of the expression for Δw_{ji} will be referred to as the learning rate because only that factor is different for different measures of fuzziness.

In the next section we shall investigate the rate of learning under the previously described schemes.

V. Learning Rate for Different Error Measures

As mentioned earlier, ignoring the common factor of $\eta f'(I_j) o_i$ in (29), (35), (44), and (50), the learning rate for different error measures can be written as

$$\begin{aligned} \Delta w_{ji}^{l} &\propto -1 \quad \text{for} \quad 0 \le o_j \le 0.5 \\ &\propto \ 1 \quad \text{for} \quad 0.5 \le o_j \le 1.0 \end{aligned} \tag{51}$$

for the *linear index* of fuzziness, as

$$\begin{aligned} \Delta w_{ji}^{q} &\propto \ -o_j \quad \text{for} \quad 0 \le o_j \le 0.5 \\ &\propto 1 - o_j \quad \text{for} \quad 0.5 \le o_j \le 1.0 \end{aligned} \tag{52}$$

for the *quadratic index* of fuzziness, as

$$\begin{aligned} \Delta w_{ji}^{le} &\propto -\frac{1}{\ln \frac{1-o_j}{o_j}} \quad \text{for} \quad 0 \le o_j \le 0.5 \\ &\propto \frac{1}{\ln \frac{o_j}{1-o_j}} \quad \text{for} \quad 0.5 \le o_j \le 1.0 \end{aligned} \tag{53}$$

for the *logarithmic entropy* ($q = 2$), and as

$$\begin{aligned} \Delta w_{ji}^{ee} &\propto -\frac{1}{(1-o_j) e^{1-o_j} - o_j e^{o_j}} \quad \text{for} \quad 0 \le o_j \le 0.5 \\ &\propto \frac{1}{o_j e^{o_j} - (1-o_j) e^{1-o_j}} \quad \text{for} \quad 0.5 \le o_j \le 1.0 \end{aligned} \tag{54}$$

for the *exponential entropy* ($q = 2$).

A critical examination of (51) through (54) reveals that in each of the four cases the learning rate is ≤ 0 for $o_j \le 0.5$ and

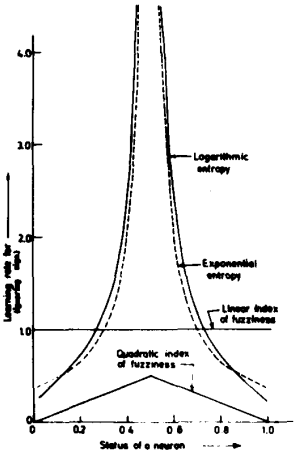


Fig. 5. Rate of learning with variation of output status for different error measures.

is ≥ 0 for $o_j \geq 0.5$. In other words, the direction of change is always the same in all the four cases. Therefore, if one compares different learning rates separately for $o_j \in [0, 0.5]$ and $o_j \in [0.5, 1.0]$, one can forget about the sign (direction) of learning rate. For example, if $\Delta w_{ji}^{le} = -5$ and $\Delta w_{ji}^{ee} = -3$, then although $\Delta w_{ji}^{le} \leq \Delta w_{ji}^{ee}$, the rate of learning is more for Δw_{ji}^{le}.

Fig. 5 depicts the rate of learning for different error measures with variation of status value of an output neuron. Because of the reasons mentioned in the previous paragraph, only the magnitude of Δw_{ji}^{*} (where $* = l/q/le/ee$) have been plotted. From the figures it is noticed that for all the error measures, the rate of learning is low when the neurons are very close to stable states (i.e., status value 0|1) and is high when the neurons are highly unstable (i.e., status value 0.5). The curves are also symmetric around 0.5 of the status value. For linear index of fuzziness the learning rate is constant.

A comparative study of the different curves shows that the learning rate is minimum (for all status values) for the quadratic index of fuzziness and its absolute value is ≤ 0.5. For linear index of fuzziness it is constant and is $= 1$. On the other hand, for entropy measures the rates of learning are high. For status values very close to 0.5, the learning rate is more for logarithmic entropy than that of the exponential entropy, whereas for output values close to 0|1 the learning rate is less for the logarithmic entropy. In other words, the change in rate of learning for exponential entropy is less than that of logarithmic entropy. So outputs are going to be more noise immune for the exponential entropy than for the logarithmic entropy; but will take more time to converge. Regarding learning rate the following proposition can be made.

Proposition: The learning rate for quadratic index of fuzziness is the minimum.

Proof: To prove the proposition it suffices to prove

1. $\Delta w_{ji}^{q} < \Delta w_{ji}^{l}$
2. $\Delta w_{ji}^{q} < \Delta w_{ji}^{le}$
3. $\Delta w_{ji}^{q} < \Delta w_{ji}^{ee}$.

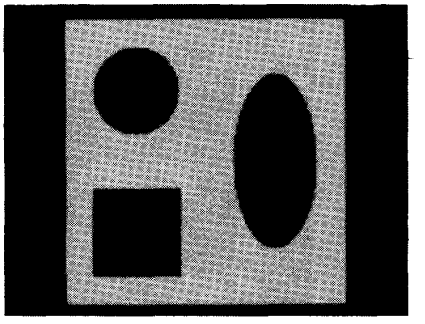


Fig. 6. Original synthetic image.

Proof of condition 1 ($\Delta w_{ji}^{q} < \Delta w_{ji}^{l}$): Leaving aside the signs

$$\Delta w_{ji}^{q} = \eta\kappa$$

with

$$\kappa = \begin{cases} o_j & \text{if, } 0 \leq o_j \leq 0.5 \\ 1 - o_j & 0.5 \leq o_j \leq 1.0. \end{cases}$$

But $\Delta w_{ji}^{l} = \eta \times 1.0$. Evidently $\Delta w_{ji}^{q} < \Delta w_{ji}^{l}$.

Proof of condition 2 ($\Delta w_{ji}^{q} < \Delta w_{ji}^{le}$): Leaving aside the signs

$$\Delta w_{ji}^{le} = \begin{cases} \eta \dfrac{1}{\ln \dfrac{1 - o_j}{o_j}} & \text{if, } 0 \leq o_j \leq 0.5 \\ \eta \dfrac{1}{\ln \dfrac{o_j}{1 - o_j}} & 0.5 \leq o_j \leq 1.0. \end{cases}$$

Proving the relation $\Delta w_{ji}^{q} < \Delta w_{ji}^{le}$ is equivalent to proving

$$\frac{1}{\ln \dfrac{1 - o_j}{o_j}} > o_j \quad \text{for} \quad 0 \leq o_j \leq 0.5$$

and

$$\frac{1}{\ln \dfrac{o_j}{1 - o_j}} > 1 - o_j \quad \text{for} \quad 0.5 \leq o_j \leq 1.0.$$

Now to prove

$$\frac{1}{\ln \dfrac{1 - o_j}{o_j}} > o_j \quad \text{for} \quad 0 \leq o_j \leq 0.5$$

or,

$$\ln \frac{1 - o_j}{o_j} < \frac{1}{o_j}$$

or,

$$\ln \left(\frac{1}{o_j} - 1 \right) < \frac{1}{o_j}$$

or,

$$\ln(x - 1) < x \quad \text{with} \quad x = \frac{1}{o_j}.$$

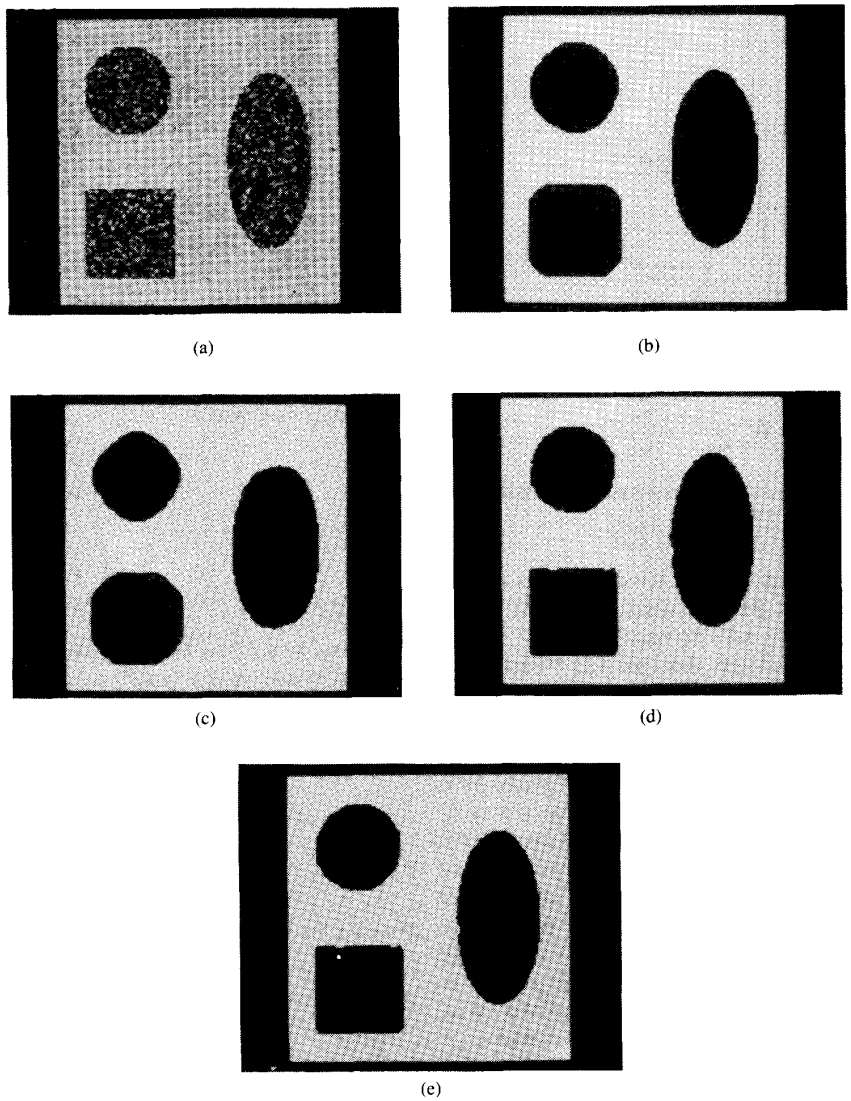


Fig. 7. Results for a noisy version ($\sigma = 10$) of the synthetic image. (a) Input. (b) Extracted object with linear index of fuzziness. (c) Extracted object with quadratic index of fuzziness. (d) Extracted object with logarithmic entropy. (e) Extracted object with exponential entropy.

Suppose

$$y = \ln(x - 1).$$

Then

$$\begin{aligned} x &= 1 + e^y \\ &= 1 + 1 + \frac{y}{1} + \frac{y^2}{2!} + \cdots \infty \\ &> y \qquad [\text{since } x \geq 2 \text{ and thereby } y \geq 0]. \end{aligned}$$

Hence, $\ln(x - 1) < x$, thus proving the first part.

In order to prove the other part

$$\frac{1}{\ln \dfrac{o_j}{1 - o_j}} > 1 - o_j \quad \text{for} \quad 0.5 \leq o_j \leq 1.0$$

or,

$$\ln \frac{o_j}{1 - o_j} < \frac{1}{1 - o_j}$$

or,

$$\ln(x - 1) < x \quad \text{with} \quad x = \frac{1}{1 - o_j}.$$

The rest of the proof is as in the previous case; hence we have the relation

$$\frac{1}{\ln \dfrac{1 - o_j}{o_j}} > o_j \quad \text{for} \quad 0 \leq o_j \leq 0.5$$

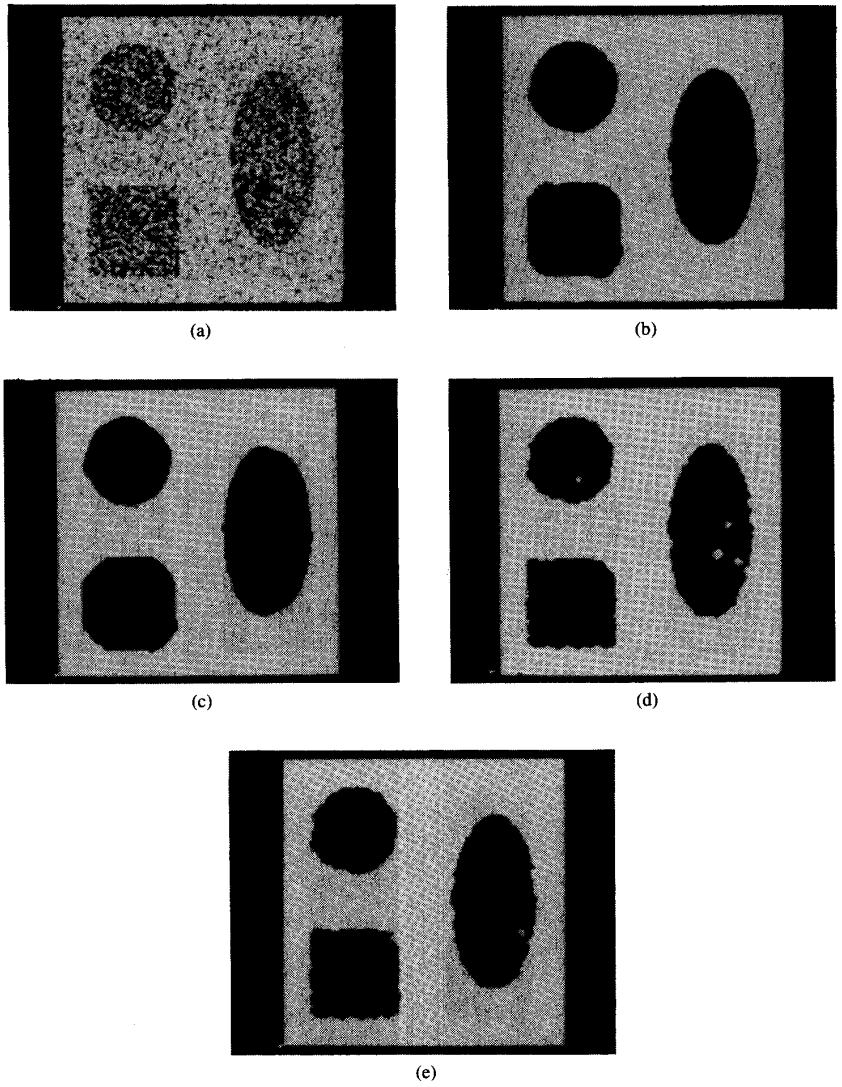


Fig. 8. Results for a noisy version ($\sigma = 20$) of the synthetic image. (a) Input. (b) Extracted object with linear index of fuzziness. (c) Extracted object with quadratic index of fuzziness. (d) Extracted object with logarithmic entropy. (e) Extracted object with exponential entropy.

and

$$\frac{1}{\ln \frac{o_j}{1-o_j}} > 1 - o_j \quad \text{for} \quad 0.5 \leq o_j \leq 1.0.$$

Proof of condition 3 ($\Delta w_{ji}^{q} < \Delta w_{ji}^{ee}$): Leaving aside the signs

$$\Delta w_{ji}^{ee} = \begin{cases} \eta \dfrac{1}{(1-o_j)e^{1-o_j} - o_j e^{o_j}} & \text{if, } 0 \leq o_j \leq 0.5 \\ \eta \dfrac{1}{o_j e^{o_j} - (1-o_j)e^{1-o_j}} & 0.5 \leq o_j \leq 1.0. \end{cases}$$

Proving the relation $\Delta w_{ji}^{q} < \Delta w_{ji}^{ee}$ is equivalent to proving

$$\frac{1}{(1-o_j)e^{1-o_j} - o_j e^{o_j}} > o_j \quad \text{for} \quad 0 \leq o_j \leq 0.5$$

and

$$\frac{1}{o_j e^{o_j} - (1-o_j)e^{1-o_j}} > 1 - o_j \quad \text{for} \quad 0.5 \leq o_j \leq 1.0.$$

Now to prove

$$\frac{1}{(1-o_j)e^{1-o_j} - o_j e^{o_j}} > o_j \quad \text{for} \quad 0 \leq o_j \leq 0.5$$

or,

$$(1-x)e^{1-x} - xe^{x} < \frac{1}{x} \quad \text{for} \quad x \in [0, 0.5]$$

(for notational simplicity x has been used in place of o_j) or

$$x(1-x)e^{1-x} - x^2 e^{x} < 1$$

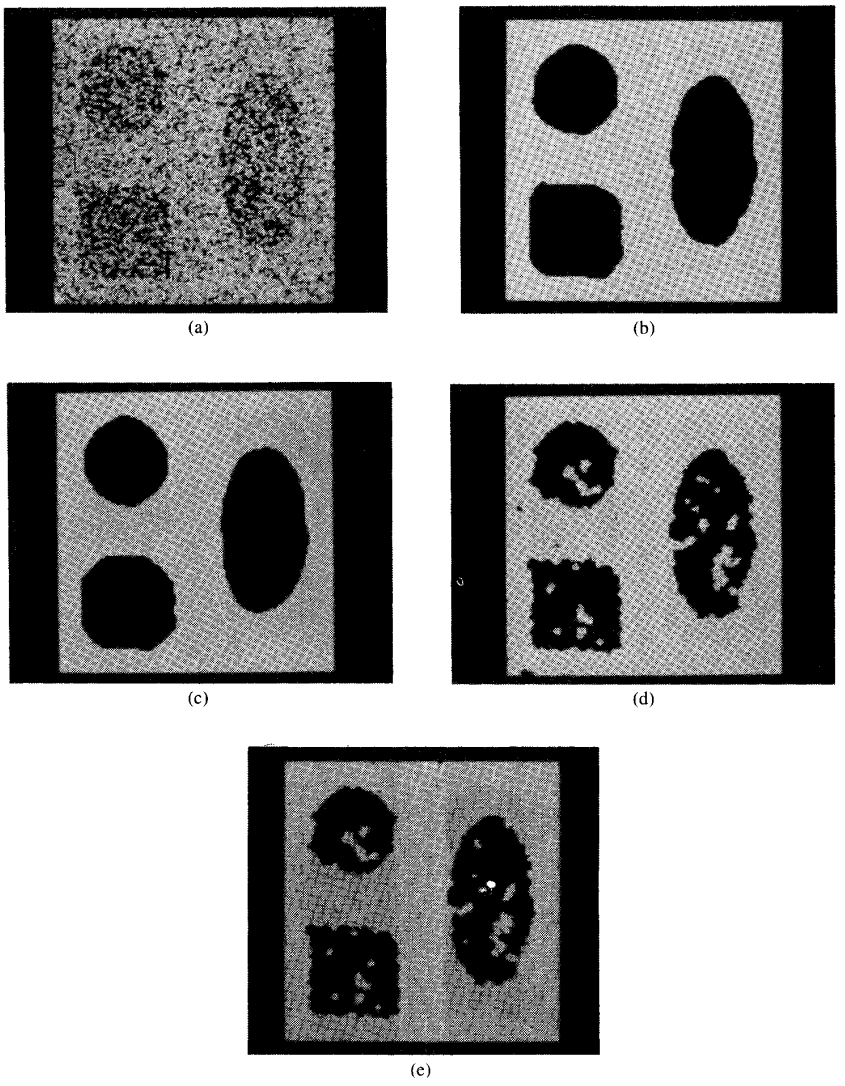


Fig. 9. Results for a noisy version ($\sigma = 32$) of the synthetic image. (a) Input. (b) Extracted object with linear index of fuzziness. (c) Extracted object with quadratic index of fuzziness. (d) Extracted object with logarithmic entropy. (e) Extracted object with exponential entropy.

or,

$$xe^{1-x} - x^2(e^{1-x} + e^x) < 1$$

or,

$$xe^{1-x} < 1 + x^2(e^{1-x} + e^x).$$

Let

$$f(x) = xe^{1-x}$$
$$f'(x) = (1-x)e^{1-x} > 0, x \in [0, 0.5]$$

$\rightarrow xe^{1-x}$ monotonically increases for x in $[0, 0.5]$

$\rightarrow xe^{1-x}$ attains the maximum value at $x = 1/2$ for $x \in [0, 0.5]$.

Hence,

$$\max\{xe^{1-x}\} = \frac{1}{2}\sqrt{e} < 1, x \in [0, 0.5].$$

On the other hand,

$$\min\{1 + x^2(e^{1-x} + e^x)\} = 1.$$

Hence the relation.

To prove the other part

$$\frac{1}{o_j e^{o_j} - (1-o_j)e^{1-o_j}} > 1 - o_j \quad \text{for} \quad 0.5 \leq o_j \leq 1.0$$

or,

$$o_j e^{o_j} - (1-o_j)e^{1-o_j} < \frac{1}{1-o_j}$$

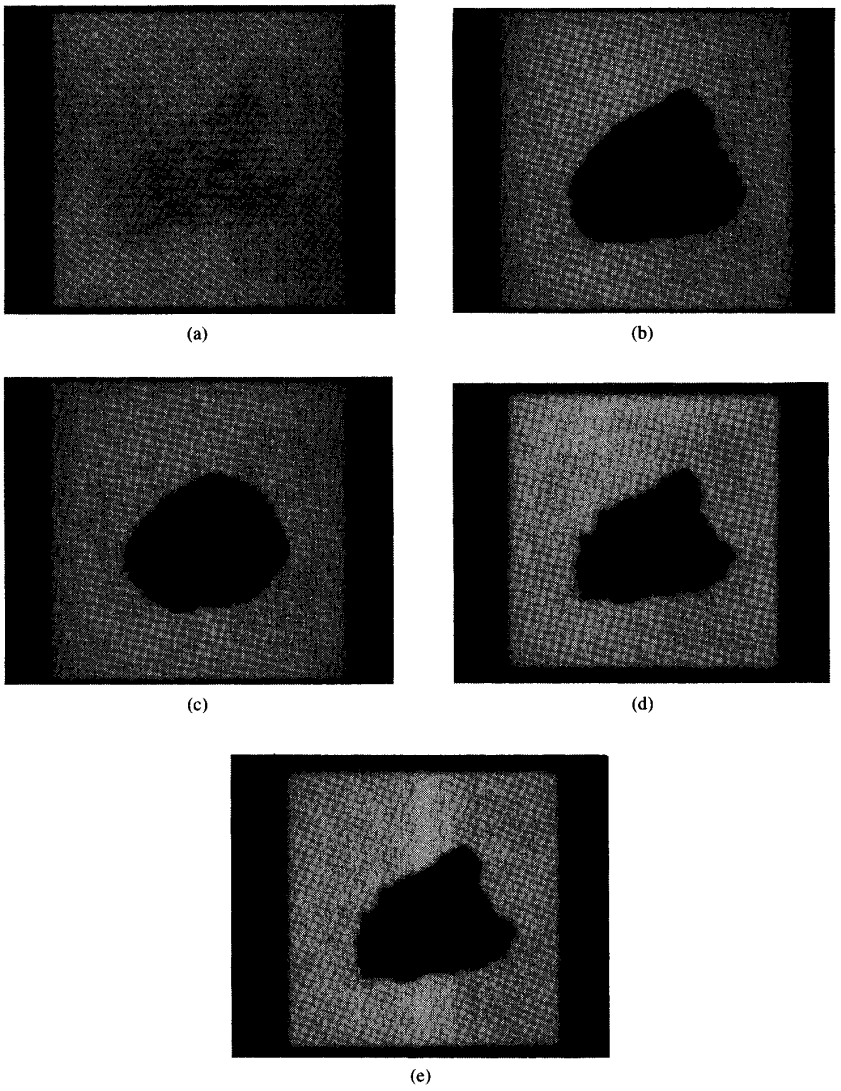


Fig. 10. Results for the noisy tank image. (a) Input. (b) Extracted object with linear index of fuzziness. (c) Extracted object with quadratic index of fuzziness. (d) Extracted object with logarithmic entropy. (e) Extracted object with exponential entropy.

or,

$$(1-x)e^{1-x} - xe^{x} < \frac{1}{x} \quad \text{with} \quad x = 1 - o_j, x \in [0, 0.5].$$

The rest of the proof is as in the previous case and thus we can write

$$\frac{1}{(1-o_j)e^{1-o_j} - o_j e^{o_j}} > o_j \quad \text{for} \quad 0 \le o_j \le 0.5$$

and

$$\frac{1}{o_j e^{o_j} - (1-o_j)e^{1-o_j}} > 1 - o_j \quad \text{for} \quad 0.5 \le o_j \le 1.0.$$

This completes the proof of the proposition.

VI. An Application to Object Extraction

A. The Object Extraction Problem

An application of the proposed network architecture is shown in object extraction (specially from noisy environments) problems. The object extraction problem can be stated as follows: *Given a noisy realization of a scene the objective is to estimate the original scene that has resulted in the observation.*

To solve the problem a three-layered version of the proposed network with N^2 is used. A neuron thus gets input from nine neurons in the previous layer. The threshold value θ in this case is $9/2 = 4.5$. The input gray levels are mapped in [0, 1] by a linear transformation and is given as input to the network. The network is then allowed to be settled. When the network has stabilized, the neurons having the status values 0 constitute

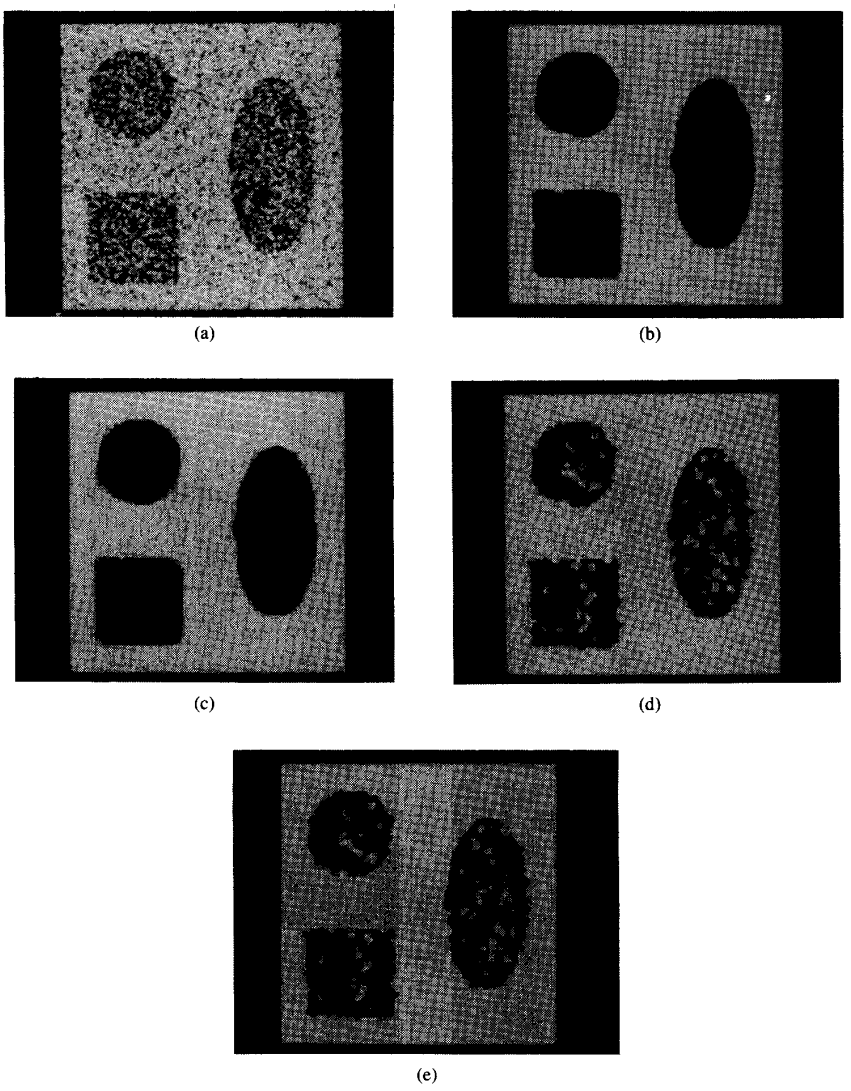


Fig. 11. Results for a noisy version ($\sigma = 20$ and N^1) of the synthetic image. (a) Input. (b) Extracted object with linear index of fuzziness. (c) Extracted object with quadratic index of fuzziness. (d) Extracted object with logarithmic entropy. (e) Extracted object with exponential entropy.

one region type say, object (background), and the remaining neurons with output status 1 constitute another region type, say background (object). Investigation has also been done with N^1.

B. Computer Simulation and Results

In order to check the effectiveness of the proposed technique, computer simulation has been done on a synthetic bitonic image (Fig. 6) corrupted by noise. The corrupted versions were obtained by adding noise from $N(0, \sigma^2)$ distribution with different values of σ (10, 20, 32). Three noisy inputs are shown in Figs. 7(a), 8(a), and 9(a). The images are of dimension 128×128 and have 32 levels. A simulation study has also been done on a real image of a *noisy tank* (Fig. 10(a)). The noisy tank image is of size 64×64 with 64 gray levels. For the simulation study η value has been taken as 0.2.

The objects extracted by the proposed technique with different expressions of error for different noisy versions of the synthetic image are included in Figs. 7–9. Fig. 10 depicts the objects extracted from the *noisy tank* image with different error models. As a typical illustration, curves (drawn on the same scale) reflecting the (amount of) change of error (i.e., ΔE) for different error measures with (time) number of iterations for a noisy input image (with $\sigma = 20$) are depicted in Fig. 12.

Examining the results, it can very easily be inferred that, as the noise level increases, the quality of the output, as expected, deteriorates, but approximate shapes and outlines are maintained. Comparing results of different error models, it is noticed that outputs with index of fuzziness measures

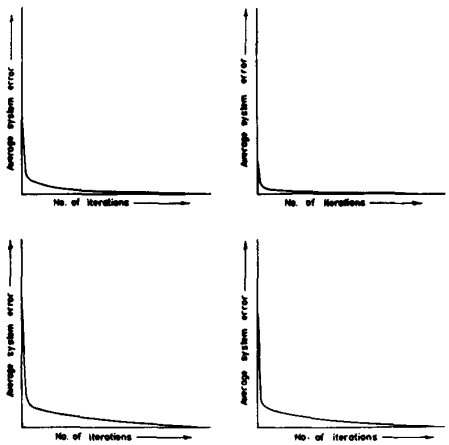


Fig. 12. Change of average system error with time for the image in Fig. 8(a). (a) Linear index of fuzziness. (b) Quadratic index of fuzziness. (c) Logarithmic entropy. (d) Exponential entropy.

(linear/quadratic) are better than those obtained by entropy measures ($q = 2$). Among the two different entropy measures, the exponential function is found to be more immune to noise. This is possibly due to different learning rates. For a fixed value of η, the learning rate is low for the indices of fuzziness, whereas it is higher for entropy measures. When the learning rate is high, a particular neuron influences its neighbors to a great extent; thus the noisy elements affect the results strongly. The system thus fails to remove all the noise. Of the two entropy measures the exponential one is more noise immune due to its lower learning rate at the initial stage of learning. A critical examination of the results reveal that the index of fuzziness (both linear and quadratic) is consistently better than entropy measures for maintaining the compactness of the extracted objects. But shapes of objects are better preserved by entropy measures. This observation can be explained as follows: Since for the index of fuzziness, the rate of learning is slow, it smoothes out noises and creates compact regions, while for entropy measures because of rapid learning all noisy pixels may not be removed, particularly when the *SNR* is very low. On the other hand, entropy measures enable the network to preserve object boundaries as learning rate is very high near the most ambiguous region ($o_j \simeq 0.5$).

The study is carried out with first order (N^1) neighborhood system and, as expected, results are not as good as with N^2. As an illustration the outputs (for different error measures) corresponding to the input image shown in Fig. 8(a) are depicted in Fig. 11. Investigation has also been done for higher values of q.

Fig. 12 depicts the variation of average system error (ASE), drawn to scale, with time (number of iterations). A comparative study of the curves reflects the following: The initial ASE's for the entropy measures are much higher than those for the indices of fuzziness. In fact for the quadratic index of fuzziness the initial ASE is the minimum. For all cases the ASE drops to a very low value (close to zero) within a few iterations, which indicates a fast convergence of the system. Note that in each of the four cases the ASE drops to almost the same low value in more or less the same number of iterations, but the initial ASE's for the entropy measures are much higher than those for linear and quadratic indices of fuzziness. Hence the rate of change of the ASE for an entropy measure is much higher than that for either of the indices of the fuzziness. Again, of the two indices of fuzziness, the linear one has a much higher initial value of the ASE, but both of them drop almost to the same low value in more or less the same number of iterations. This indicates that the rate of change of ASE for the linear index of fuzziness is greater than that for the quadratic index of fuzziness. These observations conform to the proposition stated and proved in Section V.

VII. Discussions and Conclusion

The limitations of the feedforward multilayer perceptron with back-propagation of error for image processing (segmentation/object extraction) have been addressed.

A *self-organizing multilayer neural network* suitable for image processing applications is proposed in this regard. Each neuron in the network corresponds to an image pixel. A neuron in one layer is connected to the corresponding neuron in the previous layer and the neighbors of that neuron. Neurons in the output layer are also connected to the corresponding neurons in the input layer. The output of the neurons in the output layer has been viewed as a *fuzzy set*, and measures of fuzziness have been used to model the error (instability of the network) of the system. Various mathematical models for calculation of fuzziness of this fuzzy set have been described. The weight updating rules under each model have been developed. This error is then back-propagated to correct weights so that the system error is reduced in the next stage. A comparative study (both analytical and experimental) on the rate of learning for different error measures is also done.

An application of the proposed architecture has been shown in object extraction problem from noisy environments. The algorithm has been implemented on a set of noisy images, and the approximate shapes and boundaries of the extracted objects are found to be satisfactory even for very low values of *SNR*, establishing the noise immunity of the proposed technique. Results also show that the rate of learning affects the output, especially when the noise level is very high. The outputs are better for lower learning rates (*indices of fuzziness*) and deteriorate with increase of rate of learning (*entropy measures* with $q > 2$). Thus when the noise level is low, methods with higher learning rates are preferred, but when the noise level is high, lower learning rate will be suitable.

Generalization of this type of self-organizing multilayer neural network for clustering problems raises two issues: *how to define the neighborhood system for higher dimensional feature space and how to take into account the multidimensional input vectors*. Solution to the first problem is not difficult to achieve, but the second one needs to be investigated.

Acknowledgment

The authors would like to thank the reviewers for their stimulating comments.

REFERENCES

[1] R. C. Gonzalez and P. Wintz, *Digital Image Processing*. Reading, MA: Addison-Wesley, 1977.

[2] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic Press, 1982.

[3] L. A. Zadeh, "Fuzzy sets," *Inform. and Control*, vol. 8, pp. 338–353, 1965.

[4] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[5] S. K. Pal and D. D. Majumder, *Fuzzy Mathematical Approach to Pattern Recognition*. New York: John Wiley (Halsted Press), 1986.

[6] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," in *Proc. Nat. Acad. Sci. U.S.*, pp. 3088–3092, 1984.

[7] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization of problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[8] D. E. Rumelhart, J. McClelland, and P. D. P. research group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA: MIT Press, 1986.

[9] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. New York: Addison-Wesley, 1989.

[10] T. Kohonen, *Self-Organization and Associative Memory*. Berlin: Springer-Verlag, 1989.

[11] K. Fukushima, "Neocognitron: A self-organizing multilayer neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, pp. 193–202, 1980.

[12] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, pp. 3–22, 1987.

[13] B. M. Forrest *et al.*, "Neural network models," *Parallel Computing*, vol. 8, pp. 71–83, 1988.

[14] L. O. Chua and L. Yang, "Cellular neural network: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257–1272, 1988.

[15] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1273–1290, 1988.

[16] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[17] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*. Reading, MA: Addison Wesley, 1974.

[18] A. Kandel, *Fuzzy Mathematical Techniques with Applications*. New York: Addison-Wesley, 1986.

[19] A. Deluca and S. Termini, "A definition of non probabilistic entropy in the setting of fuzzy set theory," *Inform. and Control*, vol. 20, pp. 301–312, 1972.

[20] A. Kaufmann, *Fuzzy Subsets—Fundamental Theoretical Elements*, vol. 1. New York: Academic Press, 1980.

[21] B. Kosko, "Fuzzy entropy and conditioning," *Inform. Sci.*, vol. 40, pp. 165–174, 1986.

[22] N. R. Pal and S. K. Pal, "Object background segmentation using new definition of entropy," *Proc. Inst. Elec. Eng.*, pt E, pp. 284–295, 1989.

[23] N. R. Pal and S. K. Pal, "Higher order fuzzy entropy and hybrid entropy of a set," *Inform. Sci.*, vol. 61, no. 3, pp. 211–231, 1992.

[24] M. Minsky and S. Papert, *Perceptrons*. Cambridge, MA: MIT Press, 1969.

[25] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 7, no. 6, pp. 693–699, 1985.

[26] G. L. Bilbro, M. White, and W. Synder, "Image segmentation with neurocomputers," in *Neural Computers* (R. Eckmiller and C. V. D. Malsburg, Eds.) New York: Springer-Verlag, 1988.

[27] A. Ghosh, N. R. Pal, and S. K. Pal, "Image segmentation using a neural network," *Biol. Cybern.*, vol. 66, no. 2, pp. 151–158, 1991.

[28] A. Ghosh and S. K. Pal, "Neural network, self- organization and object extraction," *Pattern Recog. Lett.*, vol. 13, no. 5, pp. 387–397, 1992.

[29] A. Ghosh, N. R. Pal, and S. K. Pal, "Object extraction using Hopfield type neural network," *Int. J. Pattern Recog. and Artificial Intelligence*, to be published.

[30] W. E. Blanz and S. L. Gish, "A connectionist classifier architecture applied to image segmentation," in *Proc. 10th Int. Conf. Pattern Recog.*, 1990, pp. 272–277.

[31] N. Babaguchi, K. Yamada, K. Kise, and Y. Tezuku, "Connectionist model binarization," in *Proc. 10th Int. Conf. Pattern Recog.*, 1990, pp. 51–56.

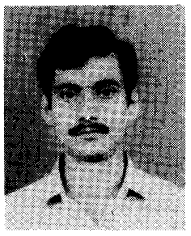
[32] B. S. Manjunath, T. Simchony, and R. Chelappa, "Stochastic and deterministic networks for texture segmentation," *IEEE Trans. Acoust., Speech, Signal Processing.*, vol. 38, no. 6, pp. 1039–1049, 1990.

[33] K. Fukushima, "A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, pp. 119–130, 1988.

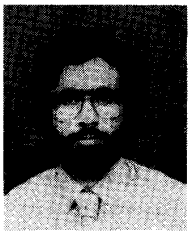
[34] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, pp. 59–69, 1982.

[35] S. Amari and M. A. Arbib, "Competition and cooperation in neural nets," *Syst. Neurosci.*, pp. 119–165, 1977.

[36] S. Amari, "Mathematical theory of self-organization in neural nets," in *Organization of Neural Networks: Structures and Models*, W. V. Seelen, G. Shaw, and U. M. Leinhos, Eds.\quad New York: Academic Press, 1988.

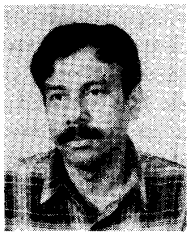


Ashish Ghosh was born in the suburb of Calcutta, India, in 1966. He received the B.E. degree in electronics and telecommunication from Jadavpur University, Calcutta, in 1987 and the M.Tech. degree in computer science from the Indian Statistical Institute, Calcutta, in 1989. At present he is working as a Senior Research Fellow in the Indian Statistical Institute, Calcutta, towards the Ph.D. degree. He received the Young Scientist award in computer science from the Indian Science Congress Association in 1992. His research interests include neural networks, image processing, fuzzy sets and systems, and pattern recognition.



Nikhil R. Pal (M'91) obtained the B.Sc. (Hons.) in physics and the M.B.M. (operations research) in 1979 and 1982, respectively from the University of Calcutta. He received the M.Tech. and Ph.D. degrees in computer science from the Indian Statistical Institute, Calcutta, in 1984 and 1991, respectively.

He was with Hindusthan Motors Ltd., W.B., from 1984 to 1985 and with Dunlop India Ltd., W.B., from 1985 to 1987. At present he is associated with the Electronics and Communication Sciences Unit of the Indian Statistical Institute, Calcutta, and is currently visiting the University of West Florida, Pensacola. He is also a guest lecturer at the University of Calcutta. His research interests include image processing, pattern recognition, artificial intelligence, fuzzy sets and systems, uncertainty measures and neural networks.



Sankar K. Pal (M'81-SM'84-F'92) is a Professor in Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta. He obtained the M.Tech. and Ph.D. degrees in radiophysics and electronics from Calcutta University and the Ph.D./DIC in electrical engineering from Imperial College, London, in 1974, 1979, and 1982 respectively. He worked at Imperial College during 1979–83 (as a Commonwealth Scholar and an MRC Postdoctoral Fellow), at the University of California, Berkeley, and the University of Maryland, College Park, during 1986–87 (as a Fulbright fellow), and at the NASA Johnson Space Center, Houston, Texas during 1990–1992 (as an NRC-NASA Senior Research Fellow).

He served as a Professor-in-Charge, Physical & Earth Sciences Division, ISI, during 1988–90. He was also a guest teacher in Computer Science, Calcutta University during 1983–86.

His research interests include Pattern Recognition, Image Processing, Neural Nets, and Fuzzy Sets and Systems. He is a co-author of the books *Fuzzy Mathematical Approach to Pattern Recognition*, Wiley (Halsted Press), N.Y., 1986 and *Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data*, IEEE Press, N.Y., 1992. He has about 150 research papers to his credit including ten in edited books and about 100 in International Journals. He is enlisted in 'Reference Asia', Asia's Who's Who of Men and Women of Achievements, vol. IV, 1989.

Dr. Pal received the 1990 Shanti Swarup Bhatnagar Prize (which is the highest and most coveted award for a scientist in India) in engineering sciences for his contribution in the field of pattern recognition and image processing. He is an Associate Editor of the *International Journal of Approximate Reasoning*, North-Holland, IEEE TRANSACTIONS ON FUZZY SYSTEMS, a reviewer of the *Mathematical Reviews* (American Mathematical Society), and a Life Fellow of IETE, New Delhi.