Pergamon

Neural Networks, Vol. 8, No. 4, pp. 643-657, 1995
Copyright © 1995 Elsevier Science Ltd
Printed in the USA. All rights reserved
0893-6080/95 $9.50 + .00

0893-6080(94)00101-4

CONTRIBUTED ARTICLE

A Connectionist System for Learning and Recognition of Structures: Application to Handwritten Characters

JAYANTA BASAK, NIKHIL R. PAL, AND SANKAR K. PAL

Indian Statistical Institute

(*Received* 10 *January* 1994; *revised and accepted* 12 *October* 1994)

Abstract—*A connectionist system for learning and recognition of structures is developed. The system is a cascade of two different modules, one for detecting linear structures (primitives) and the other for integrating these linear structures. A connectionist model implementing Hough transform has been used for the first module. The peaks in the Hough space are found by iterative verification method. A multilayered perceptron (four layers) with suitably chosen number of nodes and links has been used for the second module. As long as the size of the output layer of first module remains fixed (even if the size of input image changes), the same second module can be used and this is because the modules operate independently. The system performance is tested on handwritten Bengali character set.*

Keywords—Hough transform, Multilayered perceptron, Line detection, Character recognition.

1. INTRODUCTION

An object can be described in terms of descriptions of its parts (primitives) and spatial arrangements (relations) of the parts (noted as structural description). Recognition of the structures basically involves matching of a candidate structure with some prototype structures stored in the model base (Shapiro & Haralick, 1982; Shapiro et al., 1984; Boyer, Vayda, & Kak, 1986; Basak et al., 1993). The main difficulty in structure matching problems is that the presence of noise (and/or vagueness) may change the description of some of the primitives, thereby affecting the matching performance. Assigning some weights to the primitives and to the relations (reflecting their importance in characterizing various classes) helps, to some extent, in achieving noise tolerance and in handling impreciseness in input. These weights will be higher for the primitives and relations that are most consistent (i.e., important) in characterizing a class.

Structural description is widely used in different problems like shape matching, stereo matching, character recognition, etc. In all these problems descriptions should be such that the effect of noise gracefully degrades the performance of the system. Therefore, to design a recognition system based on structural description, one should pay attention to the proper extraction of the primitives and assignment of weights to the primitives and the relations. The extracted primitives (features) should be as robust as possible. Moreover, the system should be able to assign these weights automatically (supervised or unsupervised learning).

For designing a pattern recognition or vision system, one wishes to achieve robustness, adaptability (capability of learning the variations), and fastness (for real-time applications). Neural networks (Lippmann, 1987; Feldman & Ballard, 1982; Fahlmann & Hinton, 1987), having capability to learn from examples, and having robustness and scopes for parallelism, have recently been used for designing more intelligent recognition systems.

The objective of the present investigation is to design a scheme for structural pattern learning and recognition within a connectionist framework. The problem of handwritten character recognition is considered as a candidate for the development of the scheme. Before describing the proposed system, we give a brief review of the neural network-based character recognition systems.

In the literature, there exist various approaches based on neural networks for the character recognition problem. Possibly the first attempt was made by Fukushima (1987, 1982) for 2-D object recognition. The model (neocognitron) can recognize position and scale

Acknowledgements: The work was done while Prof. S. K. Pal held Jawharlal Nehru Fellowship. Thanks are also due to Mr. S. Chakraborty for preparing the diagrams and Mr. A. Mukherjee for his active help in capturing the character images.

Requests for reprints should be sent to Jayanta Basak, Machine Intelligence Unit, Indian Statistical Institute, Calcutta 700 035, India.

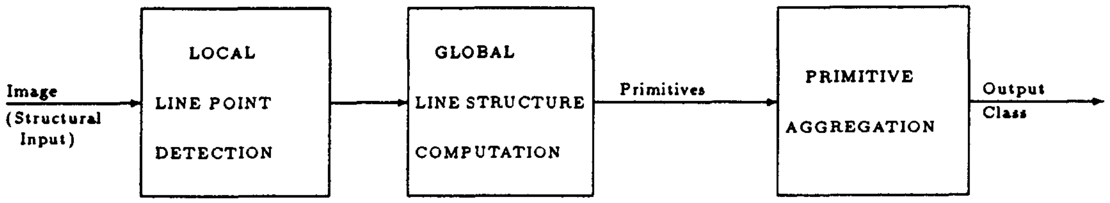


FIGURE 1. Block diagram showing the basic operations of the proposed connectionist system.

invariant handwritten numerals. Attempts, so far made, for neural network-based handwritten character recognition can be classified into two categories. In the first category, classification or matching task is based on the derived features from the character images. The second category, on the other hand, does not need to extract the features separately; the classification is performed directly from the pixel level information. Some of the investigations of the first category are as follows. Lua and Gan (1990) used the method of cooperative and competitive processing to recognize Chinese characters. They used Hebbian learning mechanism. Later, in another study, they used adaptive resonance theory for recognition of characters (Gan & Lua, 1992). Cursive scripts written by different writers were categorized using Kohonen's self-organizing feature map (Schomaker, 1993). Yong (1988) also developed a connectionist model for Chinese character recognition using strokes as the features. Recently, a newly designed network, namely CLF network (conjunction of localized features), was used to recognize handwritten numerals (Takács, Johnson, & Pieroni, 1993). In most of these investigations, the feature vectors were chosen in such a manner that they formed compact disjoint clusters corresponding to individual characters. It is therefore necessary that the features should be invariant under noisy, ambiguous environment. In other words, the process of feature extraction should be robust. But choosing the right kind of features in some scripts (like Indian) may be difficult.

Some of the attempts in the second category are as follows. Le Cun et al. (1991) used multilayered perceptron (Rumelhart & McClelland, 1986) to recognize handwritten numerals. Le Cun (1989) developed a theoretical framework for optimal selection of network structure and used it for character recognition. Denker et al. (1991) also used MLP to recognize handwritten zip codes. One of the merits of using MLP is that it can generate complex decision regions. Because these approaches take the entire image patterns directly for learning through the back-propagation rule, learning the classes from the images of complicated characters (like, Indian scripts) may take quite a long time. Moreover, the required arrangement of hidden nodes for different scripts may be different if the pixel level information is given as input.

Therefore, it seems that if the merits of robust feature selection and the characteristics of MLP in integrating the features can be exploited to design the recognition system, that would possibly lead to a more intelligent scheme for structural pattern recognition. In the present work, a six-layered connectionist system has been designed for both feature extraction and recognition. The system accepts skeleton version of images (regions) as input. Features are extracted by finding out the linear structures in the image. Then these features are hierarchically integrated by a multilayered perceptron model. In this case, the network has been designed with a view to reducing redundancy. The nearby linear structures (close in position and orientations) have been grouped hierarchically. The first three layers of the network extract the features, and other three layers integrate them to map the decision space.

The effectiveness of the model in identifying distorted versions of handwritten characters has been demonstrated. Note that though the method has been implemented on handwritten character recognition problem, it can also be applied to other structural pattern recognition and learning tasks with suitable modifications.

2. PRINCIPLE OF FEATURE EXTRACTION AND RECOGNITION

In describing the principle of structural learning and recognition process, we consider the structures to be composed of mostly linear segments. One basic requirement, as mentioned in Section 1, is that the primitives should be as robust as possible. This very fact leads to the idea that if similar structures over a neighborhood are considered as a whole for primitive extraction and attributed in a proper way, then it would result in features with better invariance. The larger the size of the neighborhood, the higher will be the invariance and the lower will be the details of the feature information. This argument holds true for any structural shape recognition problem.

The block diagram in Figure 1 shows the basic operations of the system. The system first finds out the local line points present in the skeletonized structures in the image. This is performed with the help of template matching. These templates are efficiently embedded within the links between the first and second layer of the system. The local line points are then grouped according to their orientations in the third layer of the system. This grouping is performed with the help of Hough transform. An efficient scheme for implementation of Hough transform in the connectionist framework has been designed in the present methodology. The activations in the third layer representing the linear structures are then grouped hierarchically with a MLP model.

2.1. Hough Transform and Feature Extraction

The Hough transform works as follows. Each point lying on a straight line in the image space (corresponding to nonzero pixels[1]) can be expressed as

$$r = x \cos \theta + y \sin \theta \tag{1}$$

where (x, y) is the coordinate of the concerned point in the image space and r and θ are parameters specifying the line. (r is the normal distance of the line from the origin and θ is the angle subtended by the normal with the positive x-axis.) According to this equation, several (r, θ) values can be computed depending on the (x, y) value of the point. As a result, a point in the image space corresponds to a line (sinusoidal in nature) in the (r, θ) space. For each such point in the image space, the cumulative contribution (accumulator value, Ballard & Brown, 1982) in the (r, θ) space is computed. If there exists a straight line in the image space, then the contribution to a particular (r, θ) value would be very high because all points lying on the line in image space would produce some contribution to that (r, θ) value. In practice, a cluster of activations in the (r, θ) space would be formed due to the presence of a line in the image space. Therefore, if the clusters in (r, θ) space (i.e., parameter space) can be identified, then the possible lines in the image space would be detected. In other words, if the local line direction at a pixel in the image space is specified, then it should be transformed to a unique point in the Hough space. Let us consider this point (in the Hough space) to be the representative point of that pixel. In the following claim we will consider only the representative points. A straight line in the image space can also be identified as

$$y = mx + c \tag{2}$$

where m (slope of the line) and c (intercept of the line with the y-axis) are the parameter values. However, eqns (1) and (2) essentially represent the same phenomenon. Next we present an interesting property of Hough transform in (m, c) space.

CLAIM. If a curve in the image space is continuous and second-order differentiable, then the points lying on the curve will lie in a contiguous space after Hough transformation.

[1] In this investigation nonzero pixels correspond to object region.

Proof. Suppose the image contains a curve given by $y = f(x)$. For the sake of simplicity we consider the image space to be continuous. Consider a point (x_0, y_0) on the curve. The equation of the tangent to the curve at (x_0, y_0) can be written as

$$y - y_0 = (x - x_0)f'(x_0)$$

that is,

$$y = [y_0 - x_0 f'(x_0)] + x f'(x_0). \tag{3}$$

In other words, in the parameter space the point (x_0, y_0) will be transformed to (m_1, c_1) where

$$m_1 = f'(x_0)$$

and

$$c_1 = y_0 - x_0 f'(x_0).$$

Now let us consider a nearby point on the curve $(x_0 + \Delta x, y_0 + \Delta y)$ (i.e., Δx and Δy are chosen to be very small). Suppose this point maps to a point (m_2, c_2) in the parameter space. In a similar way it can be shown that

$$m_2 = f'(x_0 + \Delta x)$$
$$c_2 = y_0 + \Delta y - (x_0 + \Delta x) f'(x_0 + \Delta x).$$

Moreover, Δy can be written as

$$\Delta y = \Delta x f'(x_0).$$

Therefore, it can be written that

$$m_2 - m_1 = \Delta x f''(x_0)$$

and

$$c_2 - c_1 = \Delta y - x_0 \Delta x f''(x_0) - \Delta x f'(x_0 + \Delta x).$$

Because the first and second derivatives exist and are finite, values of $c_2 - c_1$ and $m_2 - m_1$ can be small enough by selecting Δx to be arbitrarily small. In other words, the nearby points on the curve in the (x, y) space will lie in contiguous space in the transformed region. ■

It is to be noted here that instead of considering (m, c) space one can consider (r, θ) space also. In fact, the second one has been used in the sequel. The equation for transforming the image space into (r, θ) space is given by eqn (1). It is obvious that if a curve occupies contiguous space in the (m, c) domain then it will occupy contiguous space in the (r, θ) domain also.

If the parameter space is divided into a number of slots, then each slot corresponds to a particular straight line segment. The contribution (accumulator value) present in each slot represents the total number of pixels present in the corresponding line segment in the image space. Because of the above *claim,* a curve in the image space contributes to a contiguous chain of slots in the parameter space. This is equivalent to an approximation of a curve by a sequence of line seg-

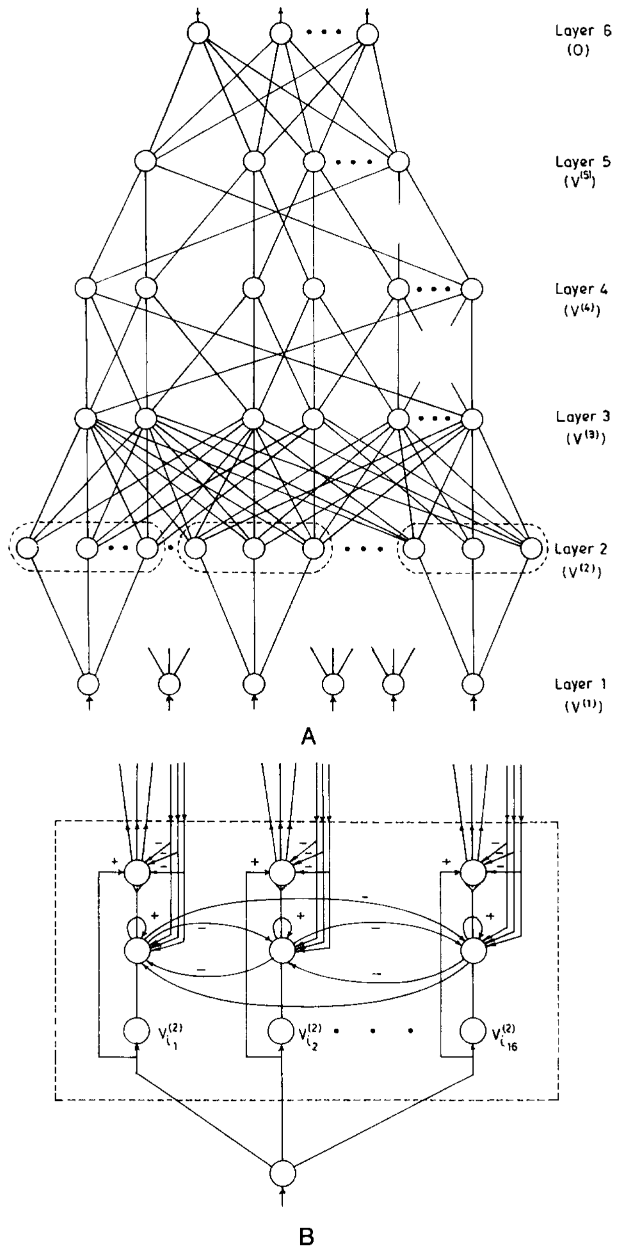


FIGURE 2. (A) Schematic diagram of the structure of the system. The layers of the network are two dimensional. Each link from the second to the third layer represents a bottom-up and a top-down link. The self-feedback connections in the third layer are not explicitly shown. The dashed box in the second layer indicates a group of nodes connected to same input node. (B) Connections between second-layer nodes connected to the same input node. The structure of each node is shown explicitly. The part that competes with other nodes is represented as a conjugation of two nodes.

ments. The level of approximation is dependent on the size of slots.

A linear structure in the image would correspond to a cluster, in addition to some spurious activations in Hough space. It is therefore necessary to extract (segment) the clusters out from the spurious responses. But the selection of the proper threshold for segmentation is a problem. Moreover, because the contribution of a curve in image space is distributed over a sequence of slots in the parameter space, the response values of some of the slots may be low and, as a result, get eliminated due to thresholding. Lowering the threshold value, on the other hand, may not be able to eliminate some of the spurious responses. However, Hough transform has ample scope for massive parallelism because the activation value in each slot can be changed independently.

Because neural networks provide a robust, massively parallel computational framework, Hough transform can possibly be efficiently implemented with connectionist models. Moreover, the problem of selecting suitable threshold can be avoided in a connectionist framework using iterative verification method (Stanfill & Waltz, 1986; Basak et al., 1993).

In the iterative verification process, the activation values in the slots of Hough space are verified against the image pixels. It is assumed each pixel in the image should vote to at most one slot in the Hough space. In the formation of initial accumulator values, each pixel can vote to more than one slot (the slots that satisfy the parametric equation for the line). The slots in the verification process compete for associating the pixels with them, and once a pixel gets associated with some slot, it supports that slot only and does not support other slots. Thus, if some spurious activations exist in the parameter space, they would lose in competition in associating the pixels, and would not get further support from the pixels. If there exists a negative self-feedback mechanism to automatically decay the activation values in the parameter space, then the spurious activation values would gradually decrease, whereas the genuine peaks would get stabilized to some nonzero values due to the support from pixels (which the peak has win over).

In the connectionist implementation, a neuron is allocated to each slot. The activation of each neuron essentially represents the amount of contributions to that slot. The clusters in the Hough space depend on the (x, y) values of the nonzero pixels in the image space. A feedback pathway is maintained from the layer corresponding to the parameter space to the layer representing the association between the pixels and the slots, to properly associate the slot activation values with the pixels. The architecture is described in Section 3 and the dynamics of the network is discussed in Section 5 in detail.

2.2. Concept of Primitive Aggregation

The activations in the Hough space represent the basic primitives of the structures. The basic primitives should be suitably integrated to represent the higher-level features (information) with better noise invariance (robust features). Because the pattern of integrating the features is highly dependent on the type of classes, it will be convenient to learn these patterns in a hierarchical (layered) connectionist model under supervised mode.

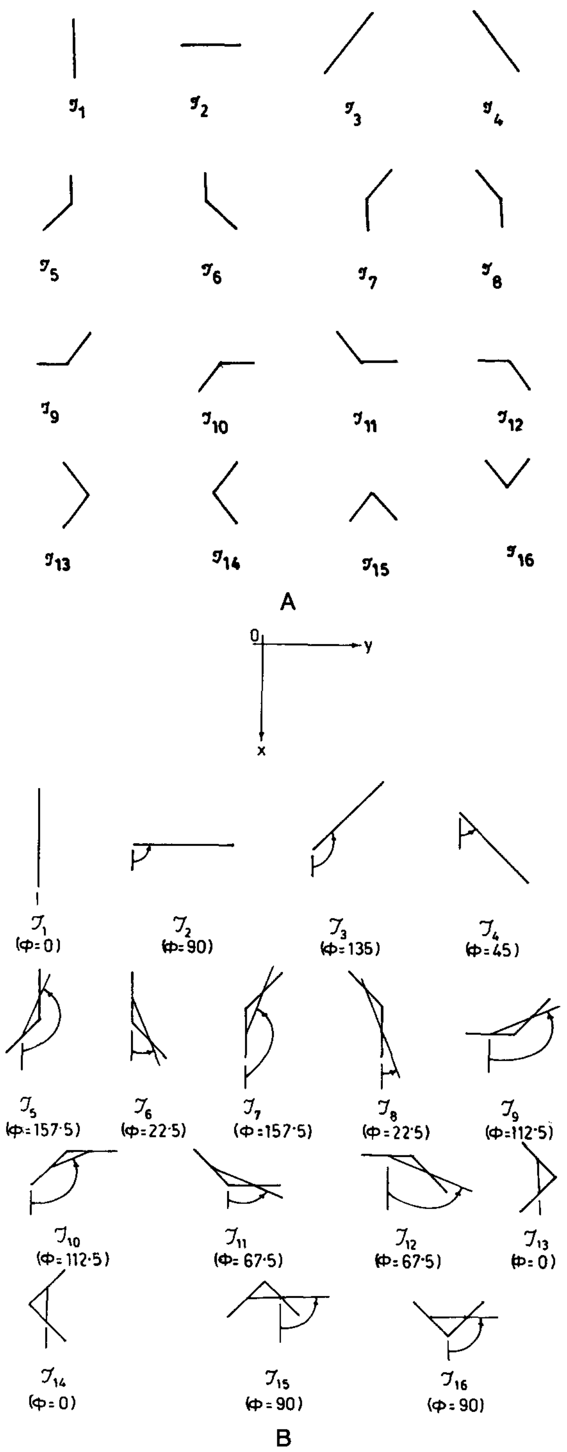
To provide better noise invariance, the primitives should be grouped over local neighborhood (in the Hough space). This is due to the fact that even if the primitive varies in its position (in the Hough space) due to the presence of noise, the effect would be reduced in the next layer of the hierarchy. Moreover, this kind of grouping may provide insensitivity to the small amount of orientations of the structures.

The hierarchical structure of the system should also be able to extract out more invariant properties from a group of primitives. Possibly this can be performed by grouping the primitives over larger neighborhoods. A variation of multilayered perceptron (with suitably selected number of hidden layers and nodes, the connections between the layers being constrained within local neighborhoods) may be used. This is described in the following sections.

3. STRUCTURE OF THE SYSTEM

The proposed connectionist system consists of six layers (Figure 2). The input layer of the network contains a 2-D array of neurons. The size of the array is the same as that of the image (say $I \times J$). Each neuron accepts an activation value equal to the normalized intensity ([0, 1]) of the corresponding pixel. In the second layer there are 16 neurons corresponding to each input neuron (i.e., second layer contains $16 \times I \times J$ neurons). The second layer associates the image space with the parameter space and each group of 16 neurons in the second layer corresponds to the 16 templates, as shown in Figure 3. The third layer corresponds to the parameter space. Each neuron in the third layer represents a slot in the parameter space of Hough transform. This layer essentially approximates the structures (single pixel thick skeletons of characters) by line segments. The size of the third layer depends on the allowed resolution in the parameter space.

The connections between the first and second layers have been exaggerated in Figure 2B. Each group of 16 second-layer neurons, connected to a single input neuron, is presented within a dotted box. Each second layer neuron has three parts, as shown in the figure. The first part holds the activation value corresponding to the respective line template. The position of the second-layer neurons within a group can be arranged according to which templates they correspond. For example, the first neuron corresponds to the first template, the second one corresponds to the second template, and so on. The activation level received by the first part of a second-layer neuron is determined by the respective template connections to the input layer and input node activations. The second part of each neuron takes part in competing with other neurons within the same group. The output of the second part always modulates the acti-



$\mathcal{T}_1$
$\mathcal{T}_2$
$\mathcal{T}_3$
$\mathcal{T}_4$
$\mathcal{T}_5$
$\mathcal{T}_6$
$\mathcal{T}_7$
$\mathcal{T}_8$
$\mathcal{T}_9$
$\mathcal{T}_{10}$
$\mathcal{T}_{11}$
$\mathcal{T}_{12}$
$\mathcal{T}_{13}$
$\mathcal{T}_{14}$
$\mathcal{T}_{15}$
$\mathcal{T}_{16}$
A
0
y
x
$\mathcal{T}_1$ (ϕ=0)
$\mathcal{T}_2$ (ϕ=90)
$\mathcal{T}_3$ (ϕ=135)
$\mathcal{T}_4$ (ϕ=45)
$\mathcal{T}_5$ (ϕ=157·5)
$\mathcal{T}_6$ (ϕ=22·5)
$\mathcal{T}_7$ (ϕ=157·5)
$\mathcal{T}_8$ (ϕ=22·5)
$\mathcal{T}_9$ (ϕ=112·5)
$\mathcal{T}_{10}$ (ϕ=112·5)
$\mathcal{T}_{11}$ (ϕ=67·5)
$\mathcal{T}_{12}$ (ϕ=67·5)
$\mathcal{T}_{13}$ (ϕ=0)
$\mathcal{T}_{14}$ (ϕ=0)
$\mathcal{T}_{15}$ (ϕ=90)
$\mathcal{T}_{16}$ (ϕ=90)
B

FIGURE 3. (A) Line segments corresponding to 16 different templates used for line detection. (B) The eight possible directions that can be represented by these segments are shown.

vation level of the third part. If the second part of some neuron loses in the competition, then the third part gets inactive, and if the second part wins, then the third part of the neuron becomes active. The third part of each neuron computes the difference between the signals coming from the input layer and the maximum feedback it is receiving from the third layer. The difference of these two signals is sent to the third layer if the corresponding neuron is a winner within its group.

The fourth layer of the network (Figure 2A) takes activations from the third and groups (smoothes) them over local neighborhoods. The objective of the fourth layer is to smooth the activation values present in the third layer to achieve robustness of the system. The fifth layer also groups activations from the fourth layer over local neighborhoods. This layer is intended to integrate linear segments over local neighborhoods to find some invariant structural properties. The size of the fourth layer is the same as that of the third layer. On the other hand, the size of the fifth layer depends on the chosen neighborhood size and the amount of overlap between local neighborhoods. The size of the local neighborhood in the fourth and fifth layers and the amount of overlap in the fifth layer will be discussed in Section 5. The number of neurons in the sixth or the output layer is equal to the number of classes (structures) to be learned and recognized. Each neuron in the output layer has connection with all nodes in the fifth layer.

The network works as follows. The input layer accepts images of skeletonized structures. As mentioned earlier, each neuron in the input layer is connected to 16 neurons in the second layer. All 16 neurons within a group in the second layer have competition between them. Each neuron in the second layer is connected to all neurons in the third layer through bottom-up and top-down links. The bottom-up links carry activations from the second to the third layer and the third layer feeds back the activation to the second through the top-down links. Each neuron in both second and third layers has a (r, θ) value associated with it. The (r, θ) value actually determines to which neuron it should send activation and from which it should receive. A neuron in the third layer would receive activation from a neuron in the second layer when there is a match between the (r, θ) values resident in these neurons. The exact mathematical model for activating a neuron is discussed in Section 4. In addition, each third layer neuron has a negative self-feedback connection.

In the initialization process, the local line directions are computed in the second layer using the template weights embedded into the links from first layer to second layer. Then the neurons in the third layer are activated from the neurons in the second layer through bottom-up links. After initialization, the nodes in the third layer feed back the activation values to the second layer through the top-down links. The second layer nodes, in turn, send the differential support to the nodes in the third layer. The activations of the nodes in the third layer are updated according to the negative self-feedback and the differential support. In this process the spurious activations do not receive any differential support and are reduced due to negative self-feedback. On the other hand, the true activations attain stable states when the differential support equals the negative self-feedback.

After stabilization the third layer represents the clusters (corresponding to linear segments in the image space) in the parameter space. The weights of the links from the first to the second and from second to the third layer are fixed. The weights of the links from the third to fourth, fourth to fifth, and fifth to sixth layer are learned using the back-propagation learning rule. If the size of the neighborhood in the fourth and fifth layers is very large, then there will be redundancy in the network and the back-propagation technique would require more time to converge. On the other hand, if the neighborhood size is very small, then the network may not be able to extract out the invariant properties of the character images. As a result, the performance of the network may be deteriorated. The selection of the approximate size of the neighborhood is discussed in Section 5.

The following notations are used in the subsequent discussions. The activation value of the ith input (first layer) neuron is represented by $v_i^{(1)}$. The second layer neurons corresponding to ith input neuron are indexed as i_j (i.e., the jth neuron among the 16 neurons connected to the ith input neuron). The activation value of the i_jth neuron in the second layer is denoted by $v_{i_j}^{(2)}$. The r and θ values present in the i_jth neuron in the second layer are denoted by $r_{i_j}^{(2)}$ and $\theta_{i_j}^{(2)}$, respectively. Similarly, the activation value, r, and θ stored in the ith third layer neuron are denoted by $v_i^{(3)}$, $r_i^{(3)}$, and $\theta_i^{(3)}$, respectively. Note that indexing of only second-layer neurons is done depending on the first-layer neurons. The neurons in the other layers are indexed independently. The activation values in the fourth and fifth layers are represented by $v_i^{(4)}$ and $v_i^{(5)}$, respectively, corresponding to the ith neuron in both layers. The output layer activations are denoted as o_is.

4. COMPUTATION OF LOCAL LINE POINTS

As mentioned before, the second layer contains information about the local line segments in the image. The local information about the possible line segments is extracted by matching suitable templates at each pixel. The templates ($\mathcal{T}_1, \ldots, \mathcal{T}_{16}$ in Figure 3) are designed considering all possible line segments that can appear over a 3×3 neighborhood in a digital grid. It is evident from the nature of the templates that they can be directly implemented in a connectionist framework by properly assigning the weights of the links. The links from the first layer to the second layer represent the template connections. Note that more than one template will respond at a junction point. Moreover, even if the concerned point is not a junction point, more than one template may produce nonzero response values. For example, if a vertical line segment is present in the image, then the $\mathcal{T}_1$ template will produce full response and $\mathcal{T}_5$,

$\mathcal{T}_6$, $\mathcal{T}_7$, $\mathcal{T}_8$ templates will produce partial responses. The situation may become confusing if no template produces full response, but more than one template produce partial responses. As a result, if a single template type is associated with each pixel to represent the possible line direction at that point, then the result may become erroneous. It is, therefore, more reasonable to associate more than one template type along with their response values with a single pixel.

Another problem of using templates to extract the local line segments is that there can be discontinuity in the template responses of the pixels belonging to the same line. Besides, the template type with the highest response (dominant template) present at a particular pixel on a line may widely change due to the presence of a small amount of noise. For example, one pixel shift of a point in a vertical line segment may cause the template $\mathcal{T}_{13}$ or $\mathcal{T}_{14}$ (instead of $\mathcal{T}_1$) to dominate at that location.

Again, note that the 16 templates correspond to only eight possible directions in [0, 180] degrees because of the fact that the templates determine the directions only on the basis of 3×3 neighborhood. Figure 3 shows how these eight directions of a line segment are measured. On the other hand, the parameter space in the third layer should be able to represent all the directions of the line structures with better resolution for effective primitive extraction. This problem can be avoided if the directions represented by different templates are iteratively averaged over local neighborhood.

The templates can be efficiently implemented by embedding the template weights into the links from the first layer to the second layer. Each group of 16 neurons in the second layer, connected to an input neuron, correspond to 16 different template structures. The position of a neuron in the group of 16 precisely identifies the template represented by that neuron.

The orientation values (directions of local line segments) are associated with the corresponding neurons in the second layer. Each neuron in the second layer stores two different values v and ϕ to represent the line strength and the orientation of the line corresponding to the respective template. The line strengths computed in the second layer are then fed back to the corresponding input neurons. Each input neuron then accepts the maximum line strength coming from the 16 second-layer neurons connected to it. Thus, the input neuron stores the maximum line strength of the corresponding pixel. Depending on the local direction of the line, the possible (r, θ) value in the parameter space is computed at each neuron in the second layer (Figure 3). This is performed with the following equations (Figure 4):

$$\theta = \begin{cases} \phi - 90 & \text{if } \phi \geq 90, \\ \phi + 90 & \text{if } y > x \text{ and } \phi < 90, \\ \phi + 270 & \text{otherwise,} \end{cases} \tag{4}$$

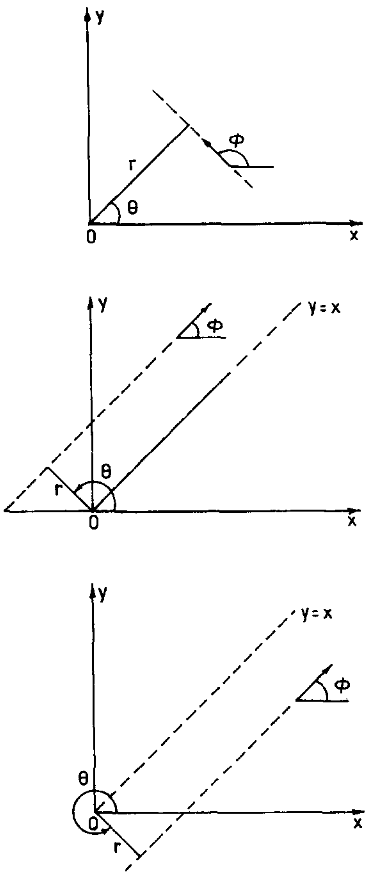


FIGURE 4. Three different cases that can occur for a line segment. In three cases the value of θ would have different relations with ϕ.

where (x, y) is the coordinate of the pixel with respect to the image reference frame. Note that the origin of the image reference frame is fixed at the upper left corner of the image. The value of r is computed once the value of θ has been found out. It is computed by eqn (1), which indicates that these calculations involve only local operations and can be implemented with local processors.

The orientation values are averaged according to the following rule.

$$\phi_{i_p}(t+1) = \frac{v_{i_p}^{(2)}\phi_{i_p}(t) + \kappa \sum_{j \neq i} \max_q[v_{j_q}^{(2)}\phi_{j_q}(t)\mathcal{N}(T_{i_p}, T_{j_q})]}{v_{i_p}^{(2)} + \kappa \sum_j \max_q[v_{j_q}^{(2)}\mathcal{N}(T_{i_p}, T_{j_q})]} \tag{5}$$

where t represents the number of iterations, κ is a constant that determines the relative importance of activations received from neighborhoods, $\mathcal{N}$ is the neighborhood function that takes values in $\{0, 1\}$ and determines the connections between the nodes in the second layer, and T_{i_p} represents the template type corresponding to the i_pth node in the second layer. The template type refers to one of the 16 templates ($\mathcal{T}_1, \ldots, \mathcal{T}_{16}$) as mentioned before. The template type corresponding to a second-layer node is determined according to the position of that node with respect to the input node. The use of max[ ] operator takes care of the fact that a second-layer node always responds to its strongest neighbor.

The neighborhood function can be described in the following way. The 16 different template types can be presented as an ordered pair (η_1, η_2) as explained below. The pair is defined by considering each line segment in Figure 3 as a collection of two segments joined at the center pixel. For example, the type at the i_pth neuron can be defined as

$$T_{i_p} = (\eta_{i_p1}, \eta_{i_p2}).$$

η_{i_p1} and η_{i_p2} represent two labels within the eight neighborhood of the processor i, as shown in Figure 5.

For example, the template type $\mathcal{T}_1$ can be represented as $\mathcal{T}_1 = (-3, 3)$ or $(3, -3)$. Similarly, $\mathcal{T}_2$ can be represented as $\mathcal{T}_2 = (-1, 1)$ or $(1, -1)$. Note that any template (η_1, η_2) is the same as (η_2, η_1). Let the positions of the processing elements i and j (in the input layer) be (x_i, y_i) and (x_j, y_j), respectively. Let a variable η be defined as

$$\eta = (x_j - x_i) + 3(y_j - y_i).$$

It is evident that if j is in the eight neighborhood of i then η will take the same set of values as shown in Figure 5. The j_qth neuron in the second layer (of type T_{j_q}) corresponding to the jth input neuron would affect the i_pth neuron (of type T_{i_p}) corresponding to the ith input neuron when either of the following conditions holds:

$$\text{A: } (\eta = \eta_{i_p1}) \wedge (\eta_{j_q1} = -\eta_{i_p1} \vee \eta_{j_q2} = -\eta_{i_p1})$$

$$\text{B: } (\eta = \eta_{i_p2}) \wedge (\eta_{j_q1} = -\eta_{i_p2} \vee \eta_{j_q2} = -\eta_{i_p2}).$$

We elaborate it with an illustration. Let the i_pth neuron in the second layer correspond to a template type $\mathcal{T}_5$ or $(2, -3)$. In that case, according to condition A, input neuron j should be placed in such a way that $\eta = 2$ [i.e., j should be placed at the lower left corner of i, and T_{j_q} should be $(-2, *)$]. Similarly, according to condition B, processing element j should be placed in such a way that $\eta = -3$ [i.e., j should be just above the ith processing element, and T_{j_q} should be $(3, *)$]. If either condition A or condition B is satisfied, then only j_qth neuron will be allowed to cooperate with i_pth neuron in the second layer. Mathematically, it can be written as

$$\mathcal{N}(T_{i_p}, T_{j_q}) = \max\{1 - (|\eta - \eta_{i_p1}| + |(\eta_{j_q1} + \eta_{i_p1}) \times (\eta_{j_q2} + \eta_{i_p1})|)(|\eta - \eta_{i_p2}| + |(\eta_{j_q1} + \eta_{i_p2}) \times (\eta_{j_q2} + \eta_{i_p2})|), 0\}. \quad (6)$$

It is to be noted here that the template types are not changed as the orientation values change (i.e., the cooperative connections in the second layer are fixed). After computation of possible direction of the representative line segment in each neuron in the second layer, (r, θ) values are computed according to eqns (4) and (1). The (r, θ) values are then stored in the second layer.

	x_{i-1}	x_i	x_{i+1}
y_{i-1}	-4	-3	-2
y_i	-1	0	1
y_{i+1}	2	3	4

FIGURE 5. Labels of the eight neighborhood of a processor.

Note that the size of the input layer is dependent on the size of the input image. For example, if the input image is of size 100×100 then the input layer of the network should have 100×100 neurons. Because the second layer consists of 16 neurons corresponding to each input neuron, the total number of neurons in the second layer is 16×10^4. Each neuron in the second layer represents a possible line direction that can be present at a pixel. The size of the second layer could have been drastically reduced if four directions corresponding to templates $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ and $\mathcal{T}_4$ were only considered.

5. COMPUTATION OF GLOBAL LINE STRUCTURES

The line structures present in the image are extracted in the third layer. This layer actually aggregates the local line response values (extracted in the second layer) according to their orientations and strengths. The clusters of activations in the parameter space (third layer of the network) are formed after stabilization of the negative self-feedback and the differential support received from the second layer. Each neuron in the second layer is connected to all neurons in the third layer. The weights of the links are fixed.

The output of each neuron also consists of three different values (v, θ, r). The output r and θ values of each neuron in the third layer depend on the position of the neuron. The output r and θ values of each second-layer neuron depend on the position of the pixel and the local line direction, which is computed by eqns (4) and (1).

The updating of the activation values of the neurons in the third layer is derived by minimizing the error of mismatch between the activations of the second-layer neurons and the feedback support from the third-layer neurons. The total error between the activations of the second layer neurons and the feedback values is given as

$$E = \frac{1}{2} \sum_i \left[v_i^{(1)} - \max_{i_j} \left(b_{i_j} \frac{v_{i_j}^{(2)}}{v_i^{(1)}} \right) \right]^2. \quad (7)$$

Note that, the activation value of a neuron in the first layer is denoted by $v^{(1)}$, that in the second layer is denoted by $v^{(2)}$, and that in the third layer by $v^{(3)}$. The second-layer neurons representing the local line response values at the ith pixel are denoted by i_js. The feedback to the i_jth second-layer neuron is denoted by b_{i_j}. The amount of feedback to a second-layer neuron is dependent on the difference between the resident (r, θ) values of the second- and third-layer neurons. The difference is modeled by Zadeh's standard π-function (Zadeh et al., 1975). The graphical representation of $\pi(x, x_0, \Delta x)$ is shown in Figure 6. The value of the feedback is given as

$$b_{i_j} = \frac{1}{h}\sum_k v_k^{(3)}\pi(\theta_k^{(3)}, \theta_{i_j}^{(2)}; \Delta\theta)\pi(r_k^{(3)}, r_{i_j}^{(2)}; \Delta r) \quad (8)$$

where h is a constant that determines the average level of activation in the third layer after stabilization. The (r, θ) values stored in the second and third layers are denoted by ($r^{(2)}$, $\theta^{(2)}$) and ($r^{(3)}$, $\theta^{(3)}$), respectively. Equation (7) indicates the fact that the nodes in the second layer that correspond to the same pixel in the image (i.e., same input node) do not cooperate, rather they compete between themselves. The use of max[ ] operator ensures that only the winner-take-all nodes in the second layer corresponding to each pixel would be able to determine the total error.

In the error expression [eqn (7)] the feedback value is modulated by the activation value $v_{i_j}^{(2)}$ of the second-layer neuron. This is because of the fact that a second-layer neuron with a low activation value may receive very high feedback. On the other hand, a neuron with a high activation value may receive a low amount of feedback. If only the feedback value was considered, then the neuron that has low activation value but high feedback would win. This may not be desirable in many cases. To consider effects of both the present activation value and the feedback value, the product of these two terms has been used.

In the formulation of error expression, only the mismatch between the activation values present in the input nodes and the feedback support has been considered. The error expression should be modeled in such a way that the redundant activations in the third layer get minimized. This can be achieved by adding an extra constraint on the total activation in the third layer. Thus, the modified error becomes

$$E = \frac{1}{2}\sum_i \left[v_i^{(1)} - \max_{i_j}\left(b_{i_j}\frac{v_{i_j}^{(2)}}{v_i^{(1)}} \right) \right]^2 + \frac{1}{2} w_s \sum_k (v_k^{(3)})^2 \quad (9)$$

where w_s provides the relative effectiveness of the extra constraint. In other words, the error expression can be written as

$$E = \frac{1}{2}\sum_i \left[v_i^{(1)} - \left(b_{i_l}\frac{v_{i_j}^{(2)}}{v_i^{(1)}} \right) \right]^2 + \frac{1}{2} w_s \sum_k (v_k^{(3)})^2 \quad (10)$$

where i_l is the winner-take-all node in the second layer corresponding to the ith input node (the competition in

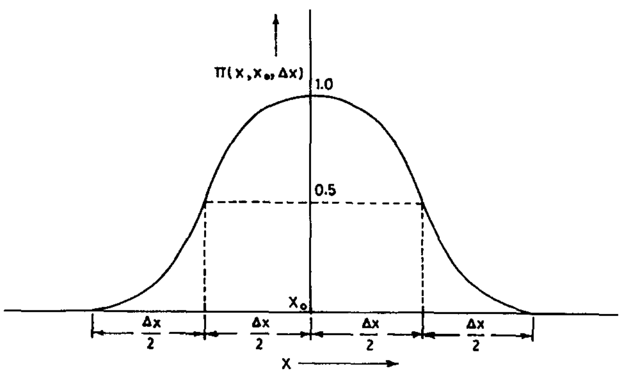


FIGURE 6. Graphical representation of $\pi(x, x_0, \Delta x)$.

the second layer takes place within each group of 16 nodes).

The changes in the activation values of the third-layer neurons are given as

$$\Delta v_k^{(3)} = -\gamma \frac{\partial E}{\partial v_k^{(3)}}, \quad (11)$$

where γ is the constant of proportionality. The rule can be derived as

$$\Delta v_k^{(3)} = \gamma \left[\frac{1}{h}\sum_i \left[v_i^{(1)} - b_{i_l}\frac{v_{i_l}^{(2)}}{v_i^{(1)}} \right] \pi(\theta_k^{(3)}, \theta_{i_l}^{(2)}; \Delta\theta) \times \pi(r_k^{(3)}, r_{i_l}^{(2)}; \Delta r)\frac{v_{i_l}^{(2)}}{v_i} - w_s v_k^{(3)} \right]. \quad (12)$$

It is therefore seen that w_s acts as the weight of the negative self-feedback. Depending on the value of w_s, the activation values in the third layer will be determined. If w_s is very high, then the proper activations will also be reduced to a great extent. On the other hand, if w_s is small the redundant activations may not be removed.

Equation (12) can be interpreted in the following way. Each neuron in the third layer feeds back its activation value to the second layer (which is given by b_{i_l}) through top-down links. A second-layer neuron is able to receive the feedback coming from the third layer only when the resident (r, θ) value matches with the (r, θ) value present in the third-layer neuron. This matching is modeled in terms of the standard π-function, as shown in eqn (8). The way of incorporating the π-function is described later. Each second-layer neuron has two parts. One of them retains the actual activation value $v^{(2)}$. The other computes the modulated activation value ($b_{i_l}(v_{i_l}^{(2)}/v_i^{(1)})$). The second part of all second-layer nodes (retaining the modulated activation value), connected to the same input node, compete between themselves. After competition is over, only the winner-take-all node becomes able to send activation to the third layer through bottom-up links. The amount of activation from the second layer to the third layer [given as $(v_i^{(1)} - b_{i_l}(v_{i_l}^{(2)}/v_i^{(1)}))(v_{i_l}^{(2)}/v_i^{(1)})$] is dependent on the input activation and modulated feedback activation. The amount of activation is defined as *differential support*. The differential support from a second-layer neuron can be mathematically modeled as

$$e_{i_j} = \begin{cases} \left[v_i^{(1)} - b_{i_j} \dfrac{v_{i_j}^{(2)}}{v_i^{(1)}} \right] \dfrac{v_{i_j}^{(2)}}{v_i^{(1)}} & \text{if } mb_{i_j} > mb_{i_l} \text{ for all } l \neq i \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where

$$mb_{i_j} = b_{i_j} \left(\frac{v_{i_j}^{(2)}}{v_i^{(1)}} \right).$$

A neuron in the third layer would be able to receive differential support from a neuron in the second layer only when the resident (r, θ) values in the second- and third-layer neurons match. The matching in the (r, θ) values between the second- and third-layer neurons is also modeled in terms of π-function [eqn (12)]. Note that the Δr and $\Delta\theta$ values for matching in the second and third layers are the same [eqns (8) and (12)]. The activation values in the third layer are updated according to the difference between the differential supports received from the second layer and the negative self-feedback. It is evident from the derivation of $\Delta v^{(3)}$ [eqn (11)] that the error value always decreases (i.e., $\Delta E \leq 0$). Because the error value is always finite [eqn (9)], in the limit, $\Delta E \to 0$ and, as a result, $\Delta v_k^{(3)} \to 0$ for all k (i.e., the system will reach a stable state).

The system needs the standard π-function to be implemented into the links between the second and third layers. The weights of the links are set fixed whenever the second-layer neurons get activated and the computation of (r, θ) values is complete. The weights (of both bottom-up and top-down links) are set in the following way:

$$w_{i_jk} = \frac{1}{h} \pi(\theta_k^{(3)}, \theta_{i_j}^{(2)}; \Delta\theta) \pi(r_k^{(3)}, r_{i_j}^{(2)}; \Delta r)$$

$$z_{ki_j} = \frac{1}{h} \pi(\theta_k^{(3)}, \theta_{i_j}^{(2)}; \Delta\theta) \pi(r_k^{(3)}, r_{i_j}^{(2)}; \Delta r)$$

where w_{i_jk} is the weight of the bottom-up link from the i_jth second-layer node to the kth third-layer node, and z_{ki_j} is the weight of the top-down link from the kth third-layer node to the i_jth second-layer node. In the setting of the weights the (r, θ) values are available at the terminal nodes of the links. (This process of weight setting should not be treated as a learning process.) The process becomes active whenever a new image is presented to the network and the computation of (r, θ) values in the second layer is completed. This is necessary both for learning and recognition. The weights remain fixed so long as the input image is not changed. Whenever a new image is presented to the network the weights are reset and fixed according to the new (r, θ) values, computed in the second layer.

The activation level in the third layer depends on the value of h and self-feedback w_s. In the following discussion we show an empirical relation between the activation level (v) in the third layer and h and w_s. Let an image contain a straight line segment of length l (i.e., l pixels). Let each pixel on the line correspond to a line strength of unity. When the image is mapped onto the connectionist model, all pixels will activate a single third-layer neuron under noiseless, ideal condition. Therefore, the updating of the activation value of that third-layer neuron representing the line segment can be written as

$$\Delta v^{(3)} = \gamma \left[\frac{l}{h} \left(1 - \frac{v^{(3)}}{h} \right) - w_s v^{(3)} \right]. \tag{14}$$

Under stable condition, $\Delta v^{(3)} = 0$. Therefore, $v^{(3)}$ will take a value

$$v^{(3)} = \frac{lh}{l + w_s h^2}. \tag{15}$$

If $w_s h^2 \gg l$ then $v^{(3)} = l/(w_s h)$, and if $l \gg w_s h^2$ then $v^{(3)} = h$. Therefore, in the first case the information about the length of the line segments is preserved. In other words, with the first condition the activation values in the third layer would correspond to the original accumulator contents (with some scaling), and with the second condition, the activation values in the third layer will have peaks of constant magnitude. This reveals an empirical parametric relation for choosing the parameters like w_s and h in the connectionist implementation of the Hough transform. In the present investigation, the values of h and w_s are chosen in such a way that the information about the line segments in the image is also restored in the activation values of the third-layer neurons.

The third layer of the network represents the parameter space of Hough transform. The size of the third layer depends on the chosen resolution of the parameter values used in the Hough transform. Using the connectionist framework for computing the peaks in Hough space, the problem of selecting suitable threshold to segment out the peaks in Hough space has been omitted. However, in the neural network model also, we need to choose the parameter values like w_s and h. But because an empirical relation between these parameters has already been derived [eqn (15)], it may provide a better parameter selection criteria. Moreover, the present scheme provides an alternative way to select the peaks instead of using the thresholding scheme, although selection of proper threshold in the standard Hough transform technique may provide comparable outputs.

6. FEATURE INTEGRATION

The activation values in the third layer represent the line segments present in the input image structures. The activation values also store the information about the length of the line segments with properly selected parameter values. These linear segments are hierarchically integrated to recognize the shapes of the structures presented to the network. The integration of the segments is performed with a multilayered perceptron model that accepts the activation values from the third layer of the system.

The MLP we considered has two hidden layers (fourth and fifth layers of the system) and one output layer (i.e., the sixth layer of the system). The connections between two consecutive layers in the MLP model are restricted over local neighborhoods (Figure 7). The restriction of the links rids the network of unnecessary details, and enables it to learn the structure with a greater speed. The size of each hidden layer and the nature of connections with its previous layer are determined according to the desired function of the layer. The way of selecting the size of hidden layers is discussed below.

The fourth layer of the system groups activation values from the third layer over local neighborhoods. The purpose of this layer is to smooth the activation values present in the third layer over local neighborhoods. The size of this layer is exactly the same as that of the third layer. The size of the third layer is 54×35. The manner of selection of the size is discussed in the next section. Every node in the fourth layer is connected to a 7×5 neighborhood of the corresponding node in the third layer. The weights of the links are set during the back-propagation learning process.

The fifth layer groups the activation values from the fourth layer over local neighborhoods. The purpose of the fifth layer is to extract out the structural properties within the neighborhood of a primitive. The neighborhoods in the fourth layer are selected in such a way that there exists sufficient overlap between two neighboring regions of activities. In fact, the size of the fifth layer would depend both on the nature of the desired overlap and the size of the neighborhoods. The relationship between the size of the layer, the size of the neighborhood, and the nature of overlap is derived below.

The shape of neighborhoods is chosen to be rectangular. Let each neighborhood be of size $m \times n$. Let p_x and p_y be the fraction of overlaps in the two orthogonal directions, respectively, which means that $p_x m$ and $p_y n$ neurons (of the fourth layer) send activations to two neighboring neurons in the fifth layer. Let the size of the fourth layer be $M \times N$. In that case, each pair of two neighboring neurons in the fifth layer corresponds to a gap of $m(1 - p_x)$ neurons in the fourth layer in one direction and $n(1 - p_y)$ neurons in the other direction. Therefore, the size of the fifth layer (say, $M' \times N'$) is given by

$$M' = \frac{M}{m(1 - p_x)}$$

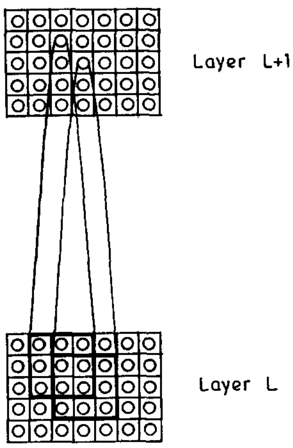


FIGURE 7. The neighborhood connections between two different layers. This is valid for connections from the third to fourth and fourth to fifth layers.

and

$$N' = \frac{N}{n(1 - p_y)}.$$

The sixth or the output layer finds out the global structure of the character. This layer accepts the activation values from all neurons in the fifth layer.

The rule used for back propagation is given as

$$\Delta w_{ji}^{(l-1)} = \eta \delta_j^{(l)} v_j^{(l-1)} \qquad (16)$$

where $w_{ji}^{(l-1)}$ stands for the weight of the link connecting the jth node in layer $l - 1$ to the ith node in layer l. The δ values are given by

$$\delta_j^{(6)} = (t_j - o_j) f'(u_j^{(6)})$$

where $u_j^{(6)}$ is the total input to the jth node in the sixth or output layer. $f(\ )$ is the transfer function of the nodes. For other layers, the δ values are given by

$$\delta_j^{(l)} = f'(u_j^{(l)}) \sum_k \delta_k^{(l+1)} w_{kj}^{(l)}.$$

Because the size of the third layer depends only on the resolution of the Hough space, it is virtually independent of the size of the input image. This indicates that the back-propagation learning rule taking place from the third layer to the sixth layer can be independent of the image size. If the image size is increased, the values w_s or h can be increased accordingly so that the activation levels in the third layer remain unaffected. For example, if image size is doubled in both x and y directions, then also the activation level in the third layer remains unchanged if the value of w_s is doubled. This indicates one novelty of this system. Once the system learns the input structure set, the same system can be used to recognize structures of different sizes. The modifications only need to be done in the first and second layers. Note that this modification does not involve any learning process.

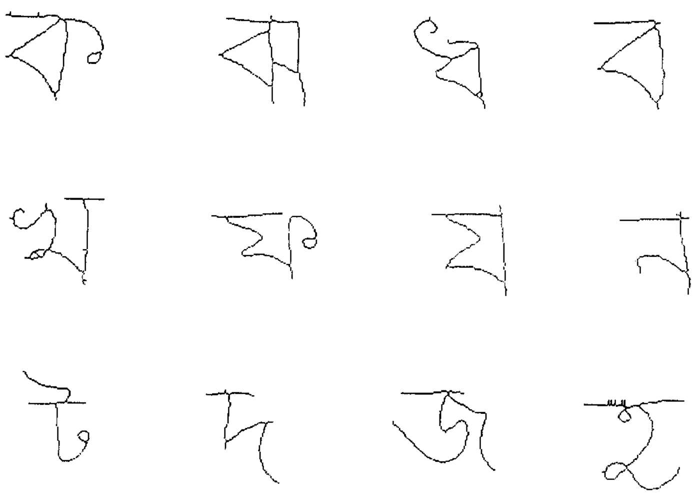


FIGURE 8. A sample character set after thinning.

7. RESULTS

The methodology described in the previous sections was implemented on handwritten Bengali (one of the major Indian language) character recognition problem. Twelve different Bengali characters were chosen for learning and recognition. The experiment was performed in two separate phases. In the first phase, only 17 samples of each character were taken and the system was trained with these samples. The performance of the system was then tested with noisy versions of the samples. In the second phase of the experiment, we enlarged our data set and considered 30 samples for each character. Noise was injected to these samples, and these samples were then divided into training and test set. It was found that with the increase in the number of samples, the performance of the system gets enhanced. Ideally, for such kind of tasks dealing with handwritten characters, the required number of samples is very high. But due to the limited computational facility, we have restricted to a small data set and have shown a clear improvement in the performance with the increase in the number of samples.

In the first part of the experiment, characters were written by 17 different persons. Therefore, the character set contained 17 $\times$ 12 (i.e., 204 characters from 12 different categories). Note that the character set we considered contains linear structures. Moreover, some of them have similar shapes. This particular set was considered to establish the discriminating ability of the proposed system even within less variant categories. In other words, the result we get is more meaningful.

The characters were preprocessed before presentation to the network. The gray level images were segmented to get binary images using gray level thresholding. The output was then smoothed and cleaned to remove noise. This was performed by growing and shrinking operations over eight neighborhood (Rosenfeld & Kak, 1982). The image was then normalized in size (100 $\times$ 100). The two-tone images were then thinned using the thinning algorithm as presented by Rosenfeld and Kak (1982). The thinned versions of some of the characters are shown in Figure 8.

The thinned versions were presented as input to the proposed system. The input image was then transformed into the parameter space as described in Sections 4 and 5. In the present case, for the sake of accuracy and robustness, all 16 templates were considered. But in the actual simulation process there is no need to allocate space for all neurons. In the present system, neurons were allocated only for the nonzero pixels, and for each nonzero pixel the first four prominent directions were considered. The value of κ [eqn (5)] was chosen as 0.5; this makes the contribution of the concerned pixel and those of its neighbors to be the same during the smoothing process of line directions. The line directions present in the second-layer neurons were iterated 10 times.

Considering the complicacy of Bengali characters, the resolution in the Hough space was chosen as $\delta r = 4$ and $\delta\theta = 5$. Note that the values of Δr and $\Delta\theta$ [eqn (12)] may be different from the resolution (i.e., δr and $\delta\theta$, respectively). However, in the present work they were chosen to be the same. For most of the Indian character set this may provide good results. For the English character set, these values would certainly work because the English alphabets are simpler in structure compared to Bengali alphabets. The number of slots the parameter spaces along the θ axis was determined as 72. Because during the computation of Hough transform the origin of the image coordinate system is considered at one particular corner of the image, the θ values lying in the third quadrant are redundant. As a result, the required number of slots in the Hough space along the θ axis becomes 54. Parameter r represents the normal distance of a line in the image space from the origin of the image coordinate system. The maximum distance can occur along the diagonal

TABLE 1
Recognition Score for Distorted Characters (Probability Value Indicating Level of Distortion)

	Probability Value		
Iteration No.	0.05	0.1	0.2
1	96.08	93.14	85.78
2	93.63	80.88	72.55

of the image. Therefore, in the present case it can be $\frac{100}{4} \times \sqrt{2}$, which is approximately 35. Thus, size of the third layer was chosen as 54×35. It is to be noted here that the size of the third layer can be kept fixed independent of the size of the image.

The values of h and w_s were chosen in such a way that the activation values in the output layer retain information about the lengths of the linear structures. In other words, the condition $w_s h^2 \gg l$ is considered. In this image we have $l < 100$ (note that l denotes the number of pixels in a linear structure). The value of h was chosen as 1000 (i.e., $l/h < 0.1$). The value of γ/h was selected as 0.1 and $w_s h$ was selected as 0.5. Therefore, the value of w_s happens to be 5×10^{-4} and γ becomes 100. Note that the factor γ/h determines the rate of updating of the states of the nodes in the network. Therefore, if γ/h is too small then the first three layers of the system would take a long time to converge and, on the other hand, if γ/h is large then there can be oscillations in the updating process. The values of h and w_s were selected depending on the maximum length of the image.

The size of the fourth layer was considered to be of the same size as the third layer. With a neighborhood size of 7×5 this gives an overlap of approximately 82% between adjacent neighborhoods in the third layer. In the fourth layer, approximately 50% overlap (between adjacent neighborhoods) is considered with a neighborhood size of 9×7. Therefore, the size of the fifth layer becomes 12×10. We used neighborhoods of size 13×11, which provides slightly more than 50% overlap. Because in the present work only 12 characters were used, the output layer consists of 12 nodes. Each node is connected to all neurons in the fifth layer.

The value of η (rate of learning) in the back-propagation rule was varied from a higher value to a smaller one. The learning started with η equal to 0.5. After each 20 iterations it was decreased by 0.1 until it became 0.2. Then η was decreased to 0.1 after 30 iterations and to 0.05 after another 30 iterations. The final tuning of the weights was performed in another 30 iterations with a value of η equal to 0.05. The network was trained with on-line learning, and the change in weights of the links was noted after every epoch. Finally, the normalized change in weights reduced below 0.00005 after 150 iterations. The total processing time for learning the character set was found to be approximately 25 h (24 h 59 min 38 s) on a SPARC 1 workstation (without floating point coprocessor). The training was performed with the entire character set (204 samples). After the training was over, the same set of characters was presented to the network for recognition, and it was found that the system was able to correctly recognize all characters.

In the next phase of the experiment, we demonstrated the effectiveness of the trained system in identifying distorted structural patterns. To generate distorted versions, each pixel can be randomly shifted (with a probability value) within its eight neighborhood, preserving the connectivity. This process can also be done iteratively to provide severe distortion.

The recognition score of the trained system is shown in Table 1 when different distorted versions (as generated by various probability values and iterations they are on) were given as input. Some of the distorted versions are shown in Figure 9 as an illustration. From

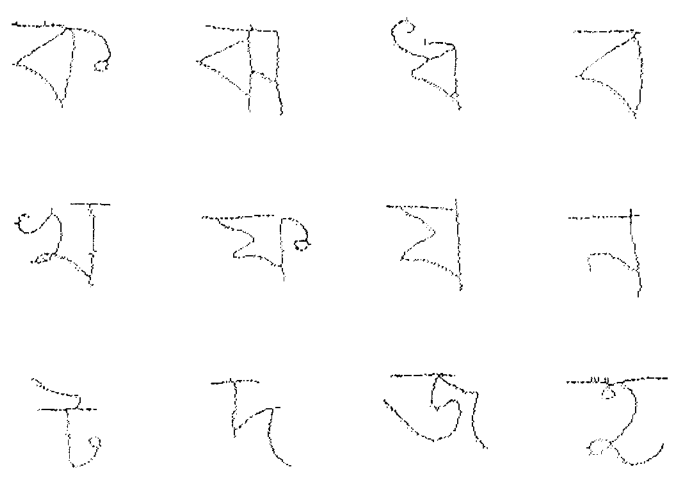


FIGURE 9. A sample set of distorted versions of the characters with probability of distortion = 0.2 and iteration no. = 2.

Table 1 it is clear that the performance of the network gracefully degrades with the level of distortion. Note that the effect of two iterations with a low probability value (0.1) is more severe than that in a single iteration with a higher probability value (0.2). The connectionist system is therefore seen to be able to recognize the handwritten characters even when some of their basic linear structures are distorted.

In the second part of the experiment, the data set was extended to include 30 different samples of each handwritten character. Noise was injected into these 30 samples (with a noise level of 20% with two iterations) to generate another 30 samples for each character. We have taken 25 original samples and 30 noisy samples for training the system. The set of weight vectors obtained in the first part of the experiment was used as a starting point in the second part of the experiment. The rate of learning was decreased whenever it was found that the network was going to oscillate around some minima. We started with a high value of η equal to 0.5 and the weights were updated for 14 iterations. Then the value of η was decreased to 0.4 and the weights were updated for 12 iterations. η was then chosen as 0.3 and the weights were updated for 13 iterations. The weights were updated for eight iterations with a value of η equal to 0.2 and 15 iterations with a value of η equal to 0.1. This part of the experiment (training phase) took approximately 85 h of CPU time in the same SPARC workstation.

The remaining five samples of each character (total 60 samples) were then presented to the network for recognition. The system was found to recognize all 60 samples correctly. This indicates the generalization capability of the system to an extent, with the increase in the number of training samples. Although we have restricted to a medium-sized data set due to insufficient computing facility, it seems that the performance of the system would get enhanced with a even larger data set. We also tested the performance of the system with another set of noisy data samples: 20% noise was injected into the 30 samples of each character (with one iteration) and the resulting 360 samples were tested with the trained system. The network was found to successfully recognize almost 86% of the samples. The performance of the network was also tested with the training samples (including the noisy samples chosen for training) and the network was found to recognize them correctly in 98% of the cases.

8. DISCUSSION AND CONCLUSION

In the present article a connectionist system for learning and recognition of structural patterns has been developed. This includes a scheme for robust feature extraction, and integration of features using multilayered perceptron model. The merits of the proposed system lie in the fact that it can select peaks automatically in the Hough space without using any threshold selection scheme, and can perform selective integration of the features during learning. Another advantage of this system is that it is able to learn structures independent of their size. If the sizes of the structures are increased, the parameters in the third layer of the system can be adjusted to get the same activation levels in the third layer. Therefore, the same MLP model (i.e., from the third to the sixth layer of the system) can be used for further learning. Because the system approximates curves with line segments using Hough transform, it seems to be efficient in handling broken line segments also.

To demonstrate the effectiveness of the model, we considered, as an example, the problem of recognizing handwritten Bengali characters of similar shapes. It was found that the performance gracefully degrades with the distortion level in input. The system was found to generalize, to a certain extent, reasonably well within the limitations of computing facility. The methodology can also be applied for recognition of other structural patterns such as industrial objects. In that case, skeletonized images of the industrial parts should be presented with a suitable frame of reference. In this model, features considered consist only of line segments. If the structural patterns consist of other features, like blobs, then the second and third layers need to be suitably modified.

REFERENCES

Ballard, D. H., & Brown, C. M. (1982). *Computer vision.* Englewood Cliffs, NJ: Prentice-Hall Inc.

Basak, J., Chaudhury, S., Pal, S. K., & Dutta Majumder, D. (1993). Matching of structural shape descriptions with hopfield net. *International Journal of Pattern Recognition and Artificial Intelligence,* **7,** 377–404.

Basak, J., Murthy, C. A., Chaudhury, S., & Dutta Majumder, D. (1993). A connectionist model category perceptron: Theory and implementation. *IEEE Neural Networks,* **4,** 257–269.

Boyer, K. L., Vayda, A. J., & Kak, A. C. (1986). Robotic manipulation experiments using structural stereopsis for 3d vision. *IEEE Expert,* 73–94.

Denker, J. S., Gardner, W. R., Graf, H. P., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., Baird, H. S., & Guyon, I. (1991). Neural network recognizer for hand-written zip code digits. In D. Touretzky (Ed.), *Proceedings of Neural Information Processing Systems.* Morgan-Kaufmann.

Fahlmann, S. E., & Hinton, G. E. (1987). Connectionist architecture for artificial intelligence. *IEEE Computer,* **20,** 100–109.

Feldmann, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science,* **6,** 205–254.

Fukushima, K. (1987). Neural network model for selective attention in visual pattern recognition and associative recall. *Applied Optics,* **26,** 4985–4992.

Fukushima, K., & Miyake, S. (1982). Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition,* **15,** 455–469.

Gan, K. W., & Lua, K. T. (1992). Chinese character classification using an adaptive resonance network. *Pattern Recognition,* **25,** 877–882.

Le Cun, Y. (1989). *Generalization and network design strategies*

(Tech. Rep. CRG-TR-89-4). Department of Computer Science, University of Toronto, Canada.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1991). Handwritten digit recognition with a back-propagation network. In D. Touretzky (Ed.), *Proceedings Neural Information Processing Systems.* Morgan-Kaufmann.

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE Acoustics, Speech, and Signal Processing Magazine,* **4,** 4–22.

Lua, K. T., & Gan, K. W. (1990). Recognizing chinese characters through interactive activation and competition. *Pattern Recognition,* **23,** 1311–1321.

Rosenfeld, A., & Kak, A. C. (1982). *Digital picture processing.* New York: Academic Press.

Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing: Explorations in microstructures of cognition (Vol. I).* Cambridge, MA: Bradford Books/MIT Press.

Schomaker, L. (1993). Using stroke- or character-based self-organizing maps in the recognition of on-line, connected cursive script. *Pattern Recognition,* **26,** 443–450.

Shapiro, L. G., & Haralick, R. M. (1982). Organization of relational models for scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **PAMI-4,** 595–602.

Shapiro, L. G., Moriarty, J. D., Haralick, R. M., & Mulgaonkar, P. G. (1984). Matching three-dimensional objects using a relational approach. *Pattern Recognition,* **17,** 385–405.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM,* **29,** 1213–1228.

Takács, B., Johnson, O., & Pieroni, G. G. (1993). Two-stage learning of handwritten characters by clf networks. *Neural Network World,* **1,** 41–51.

Yong, Y. (1988). Chinese character recognition via neural networks. *Pattern Recognition Letters,* **7,** 19–25.

Zadeh, L. A., Fu, K. S., Tanaka, K., & Shimura, M. (1975). *Fuzzy sets and their applications to cognitive and decision process.* London: Academic Press.