# A HYBRID SCHEME FOR HANDPRINTED NUMERAL RECOGNITION BASED ON A SELF-ORGANIZING NETWORK AND MLP ClASSIFIERS

UJJWAL BHATTACHARYA[*,†], TANMOY KANTI DAS[†], AMITAVA DATTA[‡], SWAPAN KUMAR PARUI[†] and BIDYUT BARAN CHAUDHURI[†]

[†] *Computer Vision and Pattern Recognition Unit,*
[‡] *Computer and Statistical Service Centre,*
*Indian Statistical Institute, 203, B. T. Road, Kolkata-108, India*
[*] *ujjwal@isical.ac.in*

This paper proposes a novel approach to automatic recognition of handprinted Bangla (an Indian script) numerals. A modified Topology Adaptive Self-Organizing Neural Network is proposed to extract a vector skeleton from a binary numeral image. Simple heuristics are considered to prune artifacts, if any, in such a skeletal shape. Certain topological and structural features like loops, junctions, positions of terminal nodes, etc. are used along with a hierarchical tree classifier to classify handwritten numerals into smaller subgroups. Multilayer perceptron (MLP) networks are then employed to uniquely classify the numerals belonging to each subgroup. The system is trained using a sample data set of 1800 numerals and we have obtained 93.26% correct recognition rate and 1.71% rejection on a separate test set of another 7760 samples. In addition, a validation set consisting of 1440 samples has been used to determine the termination of the training algorithm of the MLP networks. The proposed scheme is sufficiently robust with respect to considerable object noise.

*Keywords*: Handprinted numeral recognition; OCR; self-organizing neural network; MLP; topological and structural features; tree classifier; vector skeleton.

## 1. Introduction

During the last few decades a great deal of research work has been done in the field of optical character recognition. Still, recognition of handwritten characters remains a challenging problem even today. Machine recognition of handwritten English (Arabic) ZIP codes has been implemented with reasonable success in the USA. Considerable research for recognition of handwritten numerals has been conducted in several other scripts as well.[1,21,23,29] Recently, the need has been felt for such research work in various regional languages. This paper is concerned with recognition of handwritten numerals in Bangla, the second-most popular language and script in the Indian subcontinent and the fifth-most popular language in the world.[30,22]

Considerable research work for recognizing printed Bangla characters has been reported in previous literature.[5] But not much work has been done for recognition of handwritten Bangla characters. The latter problem, because of large variability in the input, is much more difficult than the former one. The variability is caused by variation of shapes resulting from writing habit, style, education, mood, etc. as well as factors such as writing instrument, writing surface and scanning quality. To simplify the recognition scheme, many existing methods impose some constraints on handwriting with respect to tilt, size, relative positions, stroke connections, distortions, etc. In this paper, we consider Bangla numerals written inside fixed rectangular boxes, called handprinted numerals. Automatic recognition of such offline handprinted numerals has several industrial applications such as automatic postal sorting, share certificate sorting, recognition of various other special forms etc.[23,28]

It has already been established in the literature that in a difficult classification task like handwritten character recognition, higher classification accuracy can be achieved by a combination of several feature extraction methods and several classifiers.[4,16,26,27,29] In the present work, we develop a two-stage classifier for recognition of handprinted Bangla numerals. However, before classifying an input numeral pattern, we obtain a graph representation of the shape of the input pattern. Such a graph representation is obtained by a modified version of TASONN (Topology Adaptive Self-Organizing Neural Network) model proposed by Datta *et al.*[7] Hereafter, we shall term this modified TASONN as MTASONN. Subsequently, a set of topological and structural features is extracted from the graph. This feature set is used to design a hierarchical tree classifier which accommodates almost all possible structurally different shapes of handprinted Bangla numerals in its leaf nodes at various levels. This helps to divide varying shapes of handprinted Bangla numerals into smaller subgroups on the basis of some common topological and structural properties making the recognition task at the next stage much easier. In the second stage, a multilayer perceptron network (MLP) is used to recognize the numeral shapes belonging to each leaf node. The feature set in the second stage includes certain structural features of the skeletal shape and the object pixel densities in each cell of a $4 \times 4$ rectangular grid (as shown in Fig. 1) over the raster image of the skeleton.

Neural network (NN) based character recognition systems are found to be mainly used in the high accuracy systems because they perform satisfactorily in the presence of incomplete or noisy data and they can learn from examples.[12] Another advantage is the parallel nature of the NN algorithms. In the present approach, we use two different types of neural networks — MTASONN (modified TASONN) to obtain a graph representation and multilayer perceptrons for classification. The proposed two-staged classifier has a hierarchical tree classifier in the first stage and multilayer perceptrons in the second stage. Thus, the present scheme considers an intelligent hybrid approach to recognition of handprinted Bangla numerals.

A block diagram of the proposed handprinted Bangla numeral recognition scheme is shown in Fig. 2. The scanned image of a numeral first undergoes a prepro-

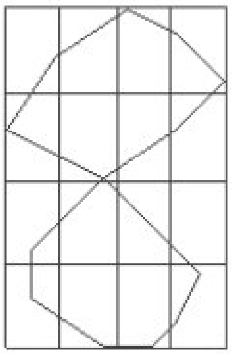


Fig. 1. The raster images of the skeletal shape of a Bangla numeral is divided into 4 × 4 zones.

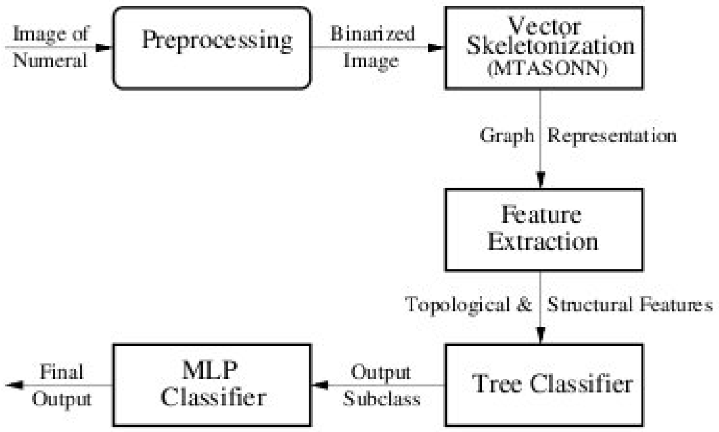


Fig. 2. A block diagram of the proposed recognition scheme.

cessing module and then MTASONN algorithm provides the graph representation of its shape. MTASONN algorithm is computationally more efficient than TASONN algorithm and yet provides an acceptable approximation to the medial axis representation of the input shape in the form of a skeleton graph. MTASONN model is described in Sec. 2. In Sec. 3, a certain pruning technique based on simple heuristics is proposed to eliminate from the graph small edges and merge redundant nodes in order to get a better skeletal representation of the actual shape of the numeral pattern. The topological and structural features of the skeleton graph are described in Sec. 4. The hierarchical tree classifier is then considered to classify the input numerals into smaller subgroups. Finally, an already trained MLP network is invoked for the final recognition of the numeral patterns belonging to each subgroup. The experimental results and conclusions are given in Secs. 5 and 6, respectively.

In an intermediate communication 90.56% correct recognition rate was reported.[2] However, significant improvement over this work has been implemented here in various areas like preprocessing, designing of the tree classifier and enhancement of the feature sets. The present improved approach provides 93.26% correct recognition rate on a test set of 7760 samples.

## 2. Neural Network Models

### 2.1. *Modified TASONN (MTASONN) model*

A vector skeleton of the binarized input numeral pattern is obtained by using a neural network model MTASONN which is a modified version of TASONN.[7] TASONN model, a variant of Neural Gas Network (NGN) model,[19,20] is found to be efficient in skeletal shape representation. Further, in MTASONN, a few modifications to the original TASONN model have been incorporated and these significantly reduce the computational load and yet provides an acceptable skeletal output. The modifications include the following:

(a) TASONN[7] model starts with an empty list of nodes and the network grows in size by means of a certain "node creation" mechanism. In MTASONN, the learning starts with an initial nonempty list of nodes. A small percentage, say 10%, of the object pixels are selected randomly to form the initial list of nodes.
(b) TASONN model updates a strength variable $\beta$ for every pair of nodes at the end of each iteration. In the modified version, these strengths are computed in a different manner in the form of a link matrix which is recomputed at the end of every phase (a phase consists of several iterations after which the network converges with the current set of nodes. The term *phase* will be defined shortly).
(c) TASONN algorithm stops when the length of every link joining two neighboring nodes is less than a parameter $\delta > 0$. In MTASONN, a different stopping criterion is considered. The present modified algorithm allows the network to grow starting from its initial size until the number of nodes reaches a maximum value, say, 15% of the object pixels. In the present application, such a modified algorithm provides an adequate vector skeleton of an input character.

In the current model, let the initial set of nodes be $[\pi_1, \pi_2, \ldots, \pi_n]$ where no prior link between these nodes is assumed. The term "link" here is used in a logical sense and it means an edge in the output graph (output network). These links are established during learning. Similar types of learning of the links from the input in a dynamically growing network are earlier used by several other authors.[6,8–11]

The input feature vectors here are the two-dimensional coordinates of the object pixels of the binarized numeral image. Let $S = \{P_1, P_2, \ldots, P_N\}$ be the set of object pixels where $N$ is the number of object pixels and $P_j = (x_j, y_j)$. The initial set of $n$ weight vectors, $W_i(t) = (w_{i1}(t), w_{i2}(t))$ at $t = 0$ are chosen randomly from the uniform distribution over the set $S$. Suppose, the object pixel $P_j$ is presented to the network during the $t$th iteration. Here it is to be noted that presentation of one input vector to the network and related actions are called one *iteration*. All iterations corresponding to the whole set of input vectors constitute a *sweep*.

Several sweeps make a *phase* and one phase is completed when the weight vectors corresponding to the current set of nodes stabilizes, i.e. when

$$\|W_i(t) - W_i(t')\| < \varepsilon\,, \quad \forall\, i\,, \tag{1}$$

where $t$ and $t'$ are the iteration numbers at the end of two consecutive sweeps and $\varepsilon$ is a predetermined small positive quantity. In the present application, we considered $\varepsilon = 0.001$. Two nearest nodes $\pi_k$ and $\pi_l$ with weight vectors $W_k(t)$ and $W_l(t)$ from $P_j$ are selected as follows.

$$\mathrm{dist}(P_j, W_k(t)) = \min_i [\mathrm{dist}(P_j, W_i(t))] \tag{2}$$

and

$$\mathrm{dist}(P_j, W_l(t)) = \min_{i(\neq k)} [\mathrm{dist}(P_j, W_i(t))]\,. \tag{3}$$

The weight vectors $W_k(t)$ and $W_l(t)$, for the first nearest node $\pi_k$ and second nearest node $\pi_l$ respectively, are updated as follows

$$W_k(t+1) = W_k(t) + \alpha_1(t)[P_j - W_k(t)] \tag{4}$$

and

$$W_l(t+1) = W_l(t) + \alpha_2(t)[P_j - W_l(t)]\,. \tag{5}$$

The quantities $\alpha_1(t)$ and $\alpha_2(t)$ $(0 < \alpha_2(t) \le \alpha_1(t) < 1)$ are the gain terms at time $t$. These quantities should decrease to zero over time, satisfying certain conditions similar to those imposed on stochastic approximation processes,[17] that is, for $r = 1, 2$

$$\begin{aligned} &\text{(i)} \quad \alpha_r(t) \to 0 \text{ as } t \to \infty \\ &\text{(ii)} \quad \sum_{t=0}^{\infty} \alpha_r(t) = \infty \quad \text{and} \\ &\text{(iii)} \quad \sum_{t=0}^{\infty} \alpha_r^2(t) < \infty\,. \end{aligned} \tag{6}$$

In our present implementation, we have chosen $\alpha_1(s)$ and $\alpha_2(s)$, where $s$ is the sweep number, as follows.

$$\alpha_1(0) = \alpha_2(0) = 0.001 \tag{7}$$

$$\alpha_1(s) = \frac{\alpha_1(0)}{1 + s/50} \text{ when } s > 0 \tag{8}$$

$$\alpha_2(s) = \frac{\alpha_2(0)}{1 + s/25} \text{ when } s > 0\,. \tag{9}$$

It has been found during our simulation runs that the above choice of $\alpha_1(s)$ and $\alpha_2(s)$ gives satisfactory results in the present problem.

Upon repeated presentations of the input vectors from the input pattern, the weight vectors tend to follow the distribution of the input space due to the weight adaptation scheme[7] described above. Thus the algorithm causes the topological ordering of the weight vectors in the input space in the sense that nodes that are adjacent in the lattice will have similar weight vectors.

At the end of a phase, a temporary link matrix $L$ is constructed. This link matrix $L$ is an upper triangular matrix of size $n \times n$, where $n$ is the current number of nodes. Here it should be noted that the value of $n$ increases by 1 from one phase to the next since only one node is inserted during a phase. Both the rows and columns of $L$ indicate the nodes of the output graph during a phase. At the beginning of each phase, the entries of $L$ are initialized with zeroes. Before the completion of a phase, for each input vector $P_j$, the first two nearest nodes, say $\pi_k$ and $\pi_l$ $(k < l)$, are determined and the entry $a_{kl}$ of the $(k,l)$-th cell of $L$ is incremented by 1. Thus one sweep through the set of all object pixels is needed to compute this link matrix. Once the link matrix is computed, we search for the cell $(k,l)$ with the maximum $a_{kl}$ value and a new node is inserted in the middle of the link connecting the $k$th and $l$th nodes. The initial weight vector of this newly inserted node becomes $[W_k(T) + W_l(T)]/2$. A new phase starts with the new set of nodes (where the number of nodes increases by 1).

Finally, when the learning converges, that is, when it grows to its maximum size, the final link matrix is used to construct the output network giving the vector skeleton of the input character. A pair of nodes, say $k$ and $l$, are connected provided the value of $a_{kl}$ is positive. In fact, this link matrix $L$ represents a graph whose vertices are defined by the individual rows or columns and the edges are defined by the nonzero entries of $L$. Thus, the shape of the input numeral pattern is extracted in the form of a skeleton graph providing a line-segment approximation to the medial axis of the pattern.

## Algorithm for MTASONN

```
Step 1. [Initialization]
        Consider the set of object pixels $P_j (j = 1, 2, \ldots N)$ as the input vectors;
        Set initial number of nodes $N_I = 0.1N$ and final number of nodes
        $N_F = 0.15N$;
        Set current number of nodes $n = N_I$ and $\varepsilon = 0.001$;
        Select randomly $n$ object pixels and use their coordinates
        as the initial set of weights $W_i(0) (i = 1, 2, \ldots, n)$;

Step 2. Repeat Steps 2–4 until $n > N_F$ [Phase]

        Do [Sweep]
            For $j = 1$ to $N$ [Iteration]
                Present input vector $P_j$ to the network of $n$ number of nodes;
                Determine first two nearest nodes to $P_j$ using conditions
            (2) and (3);
            Modify the weights corresponding to these two nodes using
            Eqs. (4) and (5);
        until the change in each weight vector during the last sweep $< \varepsilon$ as
        given by condition (1);

Step 3. [Construct the current upper triangular link matrix $L$]
        Initialize all the entries of the link matrix $L$ to zero;

        For $j = 1$ to $N$
            Set $a_{kl} = a_{kl} + 1$, where $k$th and $l$th nodes are the two nearest
            nodes from $P_j$;

Step 4. [Insertion of a new node]
        Insert a new node in the middle position between the nodes $\pi_k$ and $\pi_l$
        such that the value $a_{kl}$ of $L$ is the maximum for $1 \leq k, l \leq n$;
        Set $n = n + 1$;
Step 5. [Output skeleton graph from input pattern]
        Compute the final link matrix as in Step 3;
        Construct the graph with the final set of nodes as its vertices;
        Two vertices $k$ and $l$ $(k < l)$ in this graph are connected by
        an edge if the entry $a_{kl}$ in $L$ is nonzero.
Step 6. Stop.
```

### 2.2. *Multilayer perceptron (MLP) model*

In the proposed scheme, recognition at the second stage is done by a multilayer perceptron neural network. The MLP model is possibly the most well-known neural network architecture[15] where the strengths of connections between nodes in different layers are called weights. Such weights are usually initialized with small random values and in the present application we consider random values obtained from the uniform distribution over $[-0.5, 0.5]$. The final weights may be obtained iteratively by using the so-called backpropagation (BP) algorithm[24] as given by

$$w_{ij}(t+1) = w_{ij}(t) - \eta \frac{\partial E_p(t)}{\partial w_{ij}(t)} + \alpha_m \Delta w_{ij}(t-1) \tag{10}$$

where $w_{ij}(t)$ is the weight connecting a node $i$ and another node $j$ in the next upper layer at time $t$, $\eta > 0$ and $0 < \alpha_m < 1$ are constants, called respectively the learning rate and momentum factor. $\Delta w_{ij}(t-1)$ is the change in the corresponding weight during time $t-1$ and $E_p$ is the squared error of the network response corresponding to an input pattern $p$.

However, in real life applications, the above iterative weight modification rule often leads to very slow convergence or gets stuck at a bad local minimum. In the literature, there exist several modifications of this algorithm with improved convergence performance. In the present application, we used a modified BP algorithm which considers a distinct self-adaptive learning rate corresponding to each individual connection weight. Using such self-adaptive learning rates, the weight modification rule (10) becomes[3]

$$w_{ij}(t+1) = w_{ij}(t) - \beta_{ij}(t)\frac{\partial E_p(t)}{\partial w_{ij}(t)} + \alpha_m \Delta w_{ij}(t-1) \tag{11}$$

where $\beta_{ij}(t) = h(\eta_{ij}(t))$. $h(x) = \frac{d}{1+e^{-x+\frac{d}{2}}}$ is called the effective value function and $d > 0$ is a constant. The constant $d$ determines the maximum value which can be assumed by a learning rate. In the present application, we experimentally observed that a good choice for the value of $d$ is 4.0.

Another issue associated with the use of an MLP network is the choice of network architecture. In between the input layer at the bottom and the output layer at the top, there exists one or more layers of nodes, called hidden layers. In the present application, during simulation runs, we have observed that multiple hidden layers do not improve and sometimes even deteriorates the training performance of the network in terms of its rate of convergence. However, the selection of number of nodes in the only hidden layer is another problem and a good choice of this number cannot in general be obtained in a straightforward manner and the selection of this number involves conflicting interests.[18] A standard approach to this problem is to start with a large number of nodes in the hidden layer and then to reduce its size. There exists a large number of ways available in the literature for such removal of nodes in the hidden layer. In the present application, we considered a method proposed by Hagiwara[13] for removal of extra hidden nodes.

Finally, the strategic selection of the point of termination of the iterative learning of the connection weights of an MLP network is an important issue. To avoid overtraining, and thus achieving a better generalization performance of the network it is suggested to use a validation set of samples. Usually, in simulations using backpropagation training algorithm, the system error gradually decreases on both

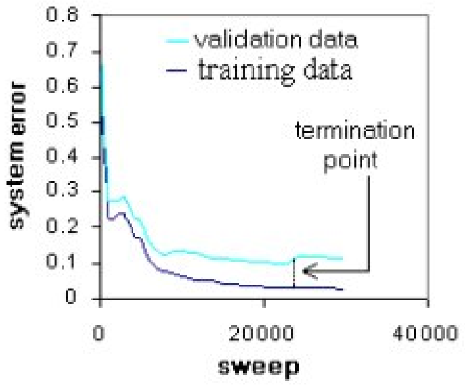


Fig. 3. Error value on the validation set indicates the termination of learning process.

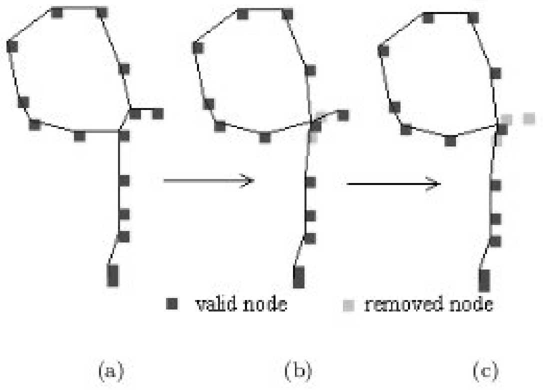


Fig. 4. An example of pruning neighboring $n$-junction and small edges.

the training and validation sets during the initial learning sweeps. After a certain amount of learning, this error further decreases on the training set while it starts increasing on the validation set[14] (as shown in Fig. 3). The point of time when the error on the validation set increases for at least three consecutive sweeps for the first instance is noted and the weight values before the error starts increasing, are stored as the final learned weights.

## 3. Pruning of Skeleton Graph

The skeleton graph as obtained by MTASONN model may sometimes contain one or more spurious parts like small loops, edges, etc. which usually lead to rejection or misclassification. Heuristics have been considered for identifying such spurious parts and removing them. The algorithm for such cleaning operations is described below.

**Algorithm for pruning:**

```
Step 1. Search the link matrix to identify nodes which are $n$-junctions
        (defined in the next section);
Step 2. While (two $n$-junctions $\pi_p$ and $\pi_q$ are neighbors)

        begin [Merge the current pair of nodes (Fig. 4)]
           A new node is placed in the middle of the line joining the
           nodes $\pi_p$ and $\pi_q$.
           The list of neighbors of the new node is computed as the union of the
           neighbors of $\pi_p$ and $\pi_q$ excepting $\pi_p$ and $\pi_q$;
           Delete the nodes $\pi_p$ and $\pi_q$ and their associated links;
           Necessary modifications are incorporated in the link matrix.
        end
```

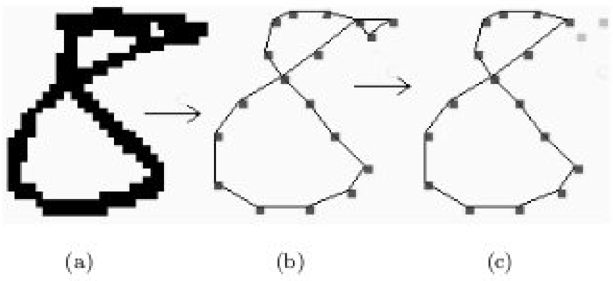


Fig. 5. An example of pruning small loop.

**Step 3.** If an $n$-junction is linked to a terminal node, then this terminal node and the corresponding link are removed.

**Step 4.** [Remove small loop(s) (Fig. 5)]
Small loop(s) consisting of three nodes only are identified. In such a loop, nodes which are not linked to nodes outside the loop are removed along with the corresponding links.

**Step 5.** Stop.

In Fig. 4, an example of pruning of neighboring $n$-junctions and a small edge are shown — Fig. 4(a) shows the initial skeleton obtained by MTASONN, Fig. 4(b) shows the same after merging of two neighboring three-junctions and Fig. 4(c) shows the final skeleton after removal of a small edge. Another example of pruning is shown in Fig. 5. The input pattern is shown in Fig. 5(a) and its skeletal representation as obtained by MTASONN is shown in Fig. 5(b). Final skeleton after pruning of the small loop is shown in Fig. 5(c).

## 4. Feature Extraction and Recognition Scheme

### 4.1. *Features*

Before describing sets of features used at the two stages of the proposed recognition scheme, it is necessary to provide an idea of the shapes of Bangla numerals to the readers. The set of Bangla numerals in the printed form is shown in Fig. 6 and a small sample set consisting of 70 handwritten numeral characters (seven different samples per class) is shown in Fig. 7.

Now let us describe several definitions and notations related to the skeleton graph obtained by MTASONN algorithm followed by its pruning.

**Definition 1.** An *$n$-junction* is a vertex of a graph having $n(> 2)$ neighbors.

**Definition 2.** A *terminal vertex* is a vertex which has only one neighbor.

**Definition 3.** The *highest vertex* is the vertex situated above all other vertices. The *lowest vertex* is similarly defined.

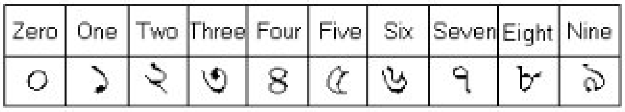

Zero
One
Two
Three
Four
Five
Six
Seven
Eight
Nine

Fig. 6. Set of Bangla numerals in an ideal form.

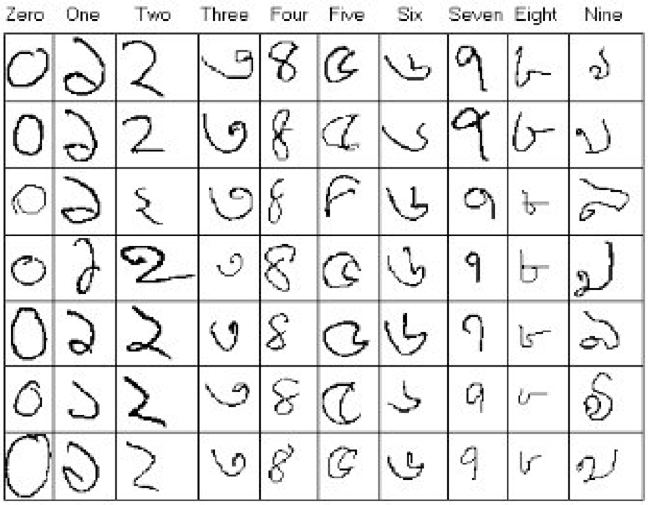


Fig. 7. A subset of handwritten Bangla numerals.

**Definition 4.** The *highest terminal vertex* is the terminal vertex situated vertically above all other terminal vertices. The *lowest terminal vertex* is similarly defined.

**Definition 5.** The *second highest terminal vertex* is the terminal vertex situated vertically above all other terminal vertices except the highest terminal vertex.

**Definition 6.** The *rightmost vertex* is the vertex situated to the right of all other vertices. The *leftmost vertex* is similarly defined.

**Definition 7.** The *rightmost terminal vertex* is the terminal vertex situated to the right of all other terminal vertices. Similarly, the *leftmost terminal vertex* is defined.

**Definition 8.** An *open arm* is a link between a terminal vertex and its neighbor.

**Definition 9.** The *right open arm* is the link between the rightmost terminal vertex and its neighbor. The *left open arm* is similarly defined.

**Definition 10.** The *cycle volume* is the proportion of the number of vertices forming the cycle to the total number of vertices forming the graph.

**Definition 11.** The *cycle centroid* is the centroid of the vertices forming the cycle.

**Definition 12.** The *character height* is the height of the smallest rectangle enclosing the graph. The *character width* is similarly defined.

### 4.2. *Recognition scheme*

Some relevants parts of graphical representations of numeric characters 2 and 9 are shown in Fig. 8. We also introduce a few notations as listed in Table 1.

The two most common features found in the extracted skeleton graph (obtained by pruning after MTASONN algorithm) of the handwritten numeral characters in

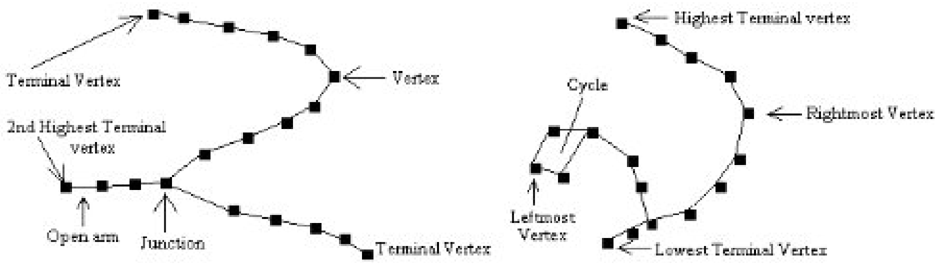


Fig. 8. Relevant parts of the graphs of characters 2 and 9.

Table 1. List of structural features and the corresponding notations.

| Notation | Description |
|---|---|
| $HD1$ | The horizontal distance between the junction and lowest terminal vertex normalized with respect to character width; (If there exists more than one junction, we consider the lowest such junction). |
| $VD1$ | The vertical distance similar to $HD1$. |
| $HD2$ | Horizontal distance between the highest terminal vertex and the lowest terminal vertex normalized with respect to character width; (If there exists only one terminal vertex, this distance is assumed to be zero). |
| $VD2$ | The vertical distance similar to $HD2$. |
| $HD3$ | Horizontal distance between the highest terminal vertex and the second highest terminal vertex normalized with respect to character width; (If there exists only one terminal vertex, this distance is assumed to be zero). |
| $VD3$ | The vertical distance similar to $HD3$. |
| $Vol$ | Cycle volume |
| $DistR$ | Distance between the rightmost vertex and the rightmost terminal vertex. |
| $GradR$ | Slope of right open arm. If there exists only one open arm, the corresponding slope is denoted by $GradR$. |
| $GradL$ | Slope of left open arm. |
| $CCGX$ | Horizontal distance of the C.G. of the vertices forming the cycle from the leftmost vertex normalized with respect to character width. |
| $CCGY$ | Vertical distance between the C.G. of the vertices forming the cycle and the highest vertex normalized with respect to character height. |

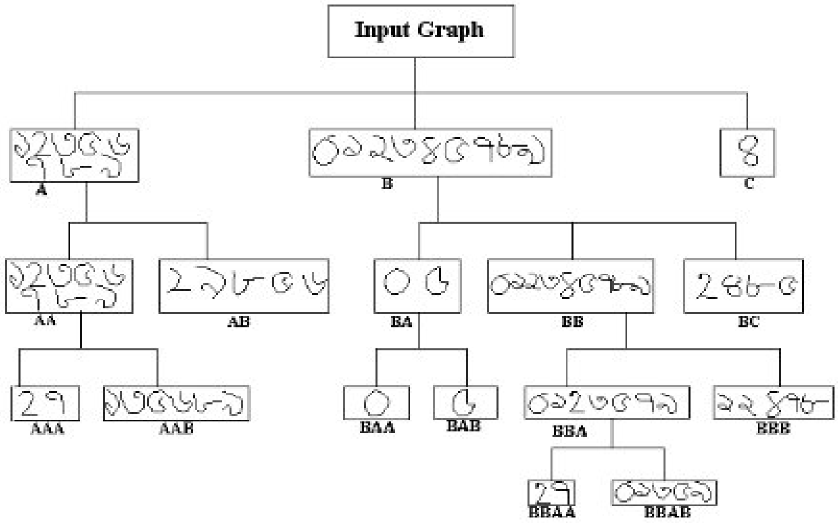


Fig. 9. Hierarchical tree classifier.

the training set are cycles and $n$-junctions. The number of cycles is either 0 or 1 or 2. It is 2 only when the numeral is four. In the first hierarchical level of the proposed tree classifier (Fig. 9), the feature considered is the number of cycles present in the skeleton graph. We consider the number of $n$-junctions as the feature in the second level of the tree classifier. The presence of concavity along a cycle of the graph and the number and position of terminal vertices are the features used in the third and fourth levels of the tree classifier. Each node of the tree classifier represents a particular group of handwritten Bangla numerals. The choice of features and decision rules at each nonterminal node of this tree classifier are described in detail below.

## Root Node (Level 0):

Features: Number of loop(s).
Decision Rule: If no loop exists then Class A; if only one loop exists then Class B; if there are two loops, then Class C consisting of the digit four only. Thus, the node representing Class C is a leaf node.

## Nodes at Level 1:

*Node representing Class A*:
Features: Number of $n$-junctions.
Decision Rule: If there is no $n$-junction then subclass AA; if there is only one $n$-junction then subclass AB. (In our database, no pattern in Class A has more than one $n$-junction).
The node representing subclass AB is a leaf node.

*Node representing Class B*:
Features: Number of $n$-junctions.
Decision Rule: If there is no $n$-junction then subclass BA; if there is only one $n$-junction then subclass BB; if there are two $n$-junctions then subclass BC. (In our database, no pattern in Class B has more than two $n$-junctions.)
The node representing subclass BC is a leaf node.

**Nodes at Level 2:**

*Node representing subclass AA*:
Features: Position of the lowest terminal vertex.
Decision Rule: If this vertex lies in the lowest one-tenth of the bounding box (minimum rectangle enclosing the skeleton graph) then subclass AAA; else subclass AAB.
Both AAA and AAB correspond to leaf nodes of the tree classifier.

*Node representing subclass BA*:
Features: Presence of concavity along the cycle.
Decision Rule: If there is no concavity along the cycle then subclass BAA consisting of only the numeral zero; else subclass BAB consisting of only the numeral five.
Both the subclasses BAA and BAB correspond to leaf nodes.

*Node representing subclass BB*:
Features: Number of terminal vertices.
Decision Rule: If there is only one terminal vertex then subclass BBA; if there are two terminal vertices then subclass BBB. (In our database, no pattern in subclass BB has more than two terminal vertices.)
The node representing subclass BBB is a leaf node.

**Nodes at Level 3:**

*Node representing Class BBA*:
Features: Position of the only terminal vertex.
Decision Rule: If the terminal vertex lies in the lower half of the bounding box of the skeleton graph then subclass BBAA; else subclass BBAB.
Both the above subclasses correspond to leaf nodes.

The list of leaf nodes of our tree classifier are the following: C in level 1, AB and BC in level 2, AAA, AAB, BAA, BAB and BBB in level 3, BBAA and BBAB in the level 4. Among these, C, BAA and BAB represent a single numeral each, viz. four, zero and five respectively. So, for these three nodes no further recognition strategy is required. For the remaining seven of the leaf nodes AB, BC, AAA, AAB, BBB, BBAA and BBAB further recognition is done as follows. To classify a pattern in such a node as a unique numeral we use an MLP network. The architectures of these MLPs are different for different leaf nodes. For each of them we use a different set of structural features and the number of possible output classes is also different.

Table 2. List of structural features and architectures for different MLPs.

| Leaf Node | Features | Architecture | Number of Possible Classes |
|---|---|---|---|
| AB | $HD2, VD2, HD3, VD3, DistR, GradR, GradL$ | $23 \times 6 \times 5$ | 5 |
| BC | $HD2, VD2, GradR, GRADL$ | $20 \times 3 \times 4$ | 4 |
| AAA | $HD2, VD2, GradR, GRADL$ | $20 \times 4 \times 2$ | 2 |
| AAB | $HD2, VD2, DistR, GradR, GradL$ | $21 \times 6 \times 6$ | 6 |
| BBB | $HD1, VD1, HD2, VD2, CCGX, CCGY$ | $22 \times 4 \times 5$ | 5 |
| BBAA | $Vol, HD1, VD1, GRADR$ | $20 \times 3 \times 2$ | 2 |
| BBAB | $Vol, HD1, VD1, CCGX, CCGY$ | $21 \times 4 \times 5$ | 5 |

In addition to these structural features, we also consider pixel densities in each of the cells of a $4 \times 4$ grid over the raster image of the skeleton. This 16-component feature is common for all of the MLP networks considered here. The structural features, network architectures and number of output classes are listed in Table 2.

## 5. Experimental Results

In this section we present results taken on a set of 11,000 handwritten Bangla numerals. These characters are read by 300 dpi HP scanner. We randomly choose 1800 samples (180 samples per class), which is little more than 16% of the whole data set, as the training set and the test set consists of 7760 characters (776 samples per class). The remaining 1440 samples (144 samples per class) have been used to construct the validation set.

Based on the topological and structural features obtained in Sec. 4.1, the corresponding leaf node of the tree classifier is determined and the relevant MLP network (Table 2) is invoked for the final classification. We use BP algorithm along with self-adaptive learning rates,[3] as described in Sec. 2.2, for training the MLP networks. The training of each of the MLPs starts with 50 nodes in the hidden layer. However, by removal of hidden layer nodes, described in Sec. 2.2, sizes of hidden layers of respective MLPs are reduced to different final sizes and these final sizes are shown in Table 2. Training of these MLPs is terminated on the basis of the validation set, as described in Sec. 2.2.

The number of nodes in the output layer of each of the MLPs is the number of concerned classes (Table 2) and these nodes have values in the range between 0 and 1. At the time of recognition, for an input numeral pattern, the nodes in the output layer having the maximum and second maximum values are obtained. If the difference between these two maximum values is less than 0.1, the input numeral is considered ambiguous and rejected. Otherwise, the node with maximum value determines the class of the input pattern.

The proposed hybrid classifier is designed on the basis of the training set. Each sample in the training set is reclassified using the classifier and the classification

Table 3. Confusion matrix obtained on the training set of 1800 samples.

Recognized as ⟶

| Numeral | ০ | ১ | ২ | ৩ | ৪ | ৫ | ৬ | ৭ | ৮ | ৯ |
|---|---|---|---|---|---|---|---|---|---|---|
| ০ | 96.1 | 0.3 | 0.2 | 0.4 | 0.21 | 0.32 | 0.4 | 0.4 | 0.45 | 0.45 |
| ১ | 0.4 | 94.6 | 0.8 | 0.5 | 0.6 | 0.55 | 0.3 | 0.23 | 0.34 | 0.6 |
| ২ | 0.15 | 1.5 | 94.1 | 0.54 | 0.38 | 0.71 | 0.43 | 0.3 | 0.25 | 0.21 |
| ৩ | 0.7 | 0.41 | 0.31 | 95.2 | 0.34 | 0.8 | 0.26 | 0.32 | 0.2 | 0.61 |
| ৪ | 0.18 | 0.6 | 0.41 | 0.4 | 94.76 | 0.3 | 0.53 | 0.74 | 0.73 | 0.42 |
| ৫ | 0.12 | 0.46 | 0.2 | 1.3 | 0.26 | 95.02 | 0.31 | 0.3 | 0.57 | 0.3 |
| ৬ | 0.34 | 0.45 | 0.21 | 0.09 | 0.23 | 0.2 | 95.6 | 0.2 | 0.35 | 0.6 |
| ৭ | 0.4 | 0.2 | 0.3 | 0.24 | 1.1 | 0.73 | 0.4 | 94.8 | 0.56 | 0.64 |
| ৮ | 0.55 | 0.61 | 0.3 | 0.2 | 0.47 | 0.25 | 0.4 | 0.3 | 95.78 | 0.1 |
| ৯ | 0.7 | 0.34 | 0.45 | 0.3 | 0.5 | 0.36 | 0.45 | 0.4 | 0.43 | 94.5 |

Table 4. Confusion matrix obtained on the test set of 7760 samples.

Recognized as ⟶

| Numeral | ০ | ১ | ২ | ৩ | ৪ | ৫ | ৬ | ৭ | ৮ | ৯ |
|---|---|---|---|---|---|---|---|---|---|---|
| ০ | 92.5 | 0.5 | 0.5 | 0.9 | 0.6 | 0.7 | 0.4 | 0.4 | 0.7 | 0.5 |
| ১ | 0.4 | 93.3 | 1.2 | 0.5 | 0.6 | 0.55 | 0.3 | 0.23 | 0.34 | 1.0 |
| ২ | 0.15 | 3 | 92.4 | 0.54 | 0.38 | 0.71 | 0.43 | 0.3 | 0.25 | 0.21 |
| ৩ | 1.1 | 0.53 | 0.48 | 93.6 | 0.34 | 0.8 | 0.26 | 0.32 | 0.2 | 0.62 |
| ৪ | 0.38 | 0.6 | 0.41 | 1.2 | 91.8 | 0.3 | 0.53 | 0.74 | 1.1 | 0.83 |
| ৫ | 0.12 | 0.46 | 0.2 | 2.6 | 0.26 | 93.1 | 0.31 | 0.3 | 0.57 | 0.3 |
| ৬ | 0.34 | 0.45 | 0.21 | 1.6 | 0.23 | 0.2 | 94.1 | 0.2 | 0.35 | 0.6 |
| ৭ | 0.4 | 0.2 | 0.3 | 0.24 | 1.4 | 0.73 | 0.4 | 93.3 | 0.56 | 0.64 |
| ৮ | 0.55 | 0.8 | 0.5 | 0.2 | 0.7 | 0.25 | 0.4 | 0.3 | 95.03 | 0.1 |
| ৯ | 1.1 | 0.8 | 0.45 | 0.3 | 0.5 | 0.81 | 0.45 | 0.4 | 0.43 | 93.51 |

results are shown in the confusion matrix in Table 3. Here the entry in the $i$th row and $j$th column of the confusion matrix shows the percentage of the number of times that the numeral in the $i$th row is classified as the numeral in the $j$th column. Similarly, Table 4 summarizes the classification results with respect to the test set. From these matrices it is noted that the correct classification rate is 95.05% on the training set and 93.26% on the test set. Rejections from the training and test sets are 1.12% and 1.71%, respectively. Numbers of misclassified samples are 69 and 390, respectively from the training and test sets.

## 6. Conclusions

The proposed hybrid scheme provides good classification results for handprinted Bangla numerals. First, a vector skeleton of the numeral pattern is extracted in the form of a graph using a computationally efficient self-organizing neural network model. The advantages of this neural network approach over various conventional techniques include its capability of higher data reduction and robustness with respect to boundary and interior noise. In Fig. 10(a), a character (eight) affected by 50% random noise is shown and its skeletal shape representation (obtained by

Fig. 10. (a) Noisy input. (b) Output skeletal shape.

MTASONN) is shown in Fig. 10(b). This representation can significantly reduce the effect of noise so that the recognition is not affected. This fact also ensures that the method works even if there are small breaks on the contour of a character.

A certain set of topological and structural features of the skeleton graph is then considered on the basis of which a tree classifier is proposed which classifies input numerals into smaller subgroups. Multilayer perceptron classifiers are then used for final classification of numerals belonging to the subgroups.

We have described the present recognition scheme in the context of handprinted Bangla numerals. However, a similar approach may be used for numeral recognition of various other scripts. The hierarchical tree classifier and the feature set may, however, vary in such cases.

**Acknowledgment**

The authors gratefully acknowledge the valuable suggestions by the annonymous reviewers which significantly contributed to the improvement of the present article.

## References

1. A. Amin, "Off-line Arabic character recognition: the state of the art," *Patt. Recogn.* **31** (1998) 517–530.
2. U. Bhattacharya, T. K. Das, A. Datta, S. K. Parui and B. B. Chaudhuri, "Recognition of handprinted Bangla numerals using neural network models," *Advances in Soft Computing — AFSS 2002*, Lecture Notes on Artificial Intelligence, LNAI, Vol. 2275, eds. N. R. Pal and M. Sugeno, Springer Verlag, 2002, pp. 228–235.
3. U. Bhattacharya and S. K. Parui, "Self-adaptive learning rates in backpropagation algorithm improve its function approximation performance," *Proc. IEEE Int. Conf. Neural Networks*, Australia, 1995, pp. 2784–2788.
4. J. Cao, M. Ahmadi and M. Shridhar, "Recognition of handwritten numerals with multiple feature and multistage classifier," *Patt. Recogn.* **28** (1995) 153–160.
5. B. B. Chaudhuri and U. Pal, "A complete printed Bangla OCR system," *Patt. Recogn.* **31** (1998) 531–549.
6. D. Choi and S. Park, "Self-creating and organizing neural networks," *IEEE Trans. Neural Networks* **5** (1994) 561–575.
7. A. Datta, S. K. Parui and B. B. Chaudhuri, "Skeletonization by a topology adaptive self-organizing neural network," *Patt. Recogn.* **34** (2001) 617–629.
8. B. Fritzke, "Let it grow — self-organizing feature maps with problem dependent cell structure," *Artificial Neural Networks*, eds. T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, North-Holland, Vol. 1, 1991, pp. 403–408.
9. B. Fritzke, "Growing cell structures — a self-organizing network for unsupervised and supervised learning," *Neural Networks* **7** (1994) 1441.

10. B. Fritzke, "Growing grid, a self-organizing network with constant neighborhood range and adaptation strength," *Neural Process. Lett.* **2** (1995) 1.
11. B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems*, eds. G. Tesauro, D. S. Touretzky and T. K. Leen, MIT Press, Cambridge MA, Vol. 7, 1995, pp. 625–632.
12. M. D. Garris, C. L. Wilson and J. L. Blue, "Neural network-based systems for handprint OCR applications," *IEEE Trans. Imag. Process.* **7** (1998) 1097–1112.
13. M. Hagiwara, "A simple and effective method for removal of hidden units and weights," *Neurocomputing* **6** (1994) 207–218.
14. M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, The MIT Press, Cambrige, 1995, p. 226.
15. R. Hecht-Nielson, *Neurocomputing*, Chap. 5, Addison-Wesley, NY, 1990.
16. F. Kimura and M. Sridhar, "Handwritten numeral recognition based on multiple algorithms," *Patt. Recogn.* **24** (1991) 976–983.
17. T. Kohonen, *Self-Organization and Associative Memory*, Springer, Berlin, 1989.
18. J. K. Kruschke, "Creating local and distributed bottlenecks in hidden layers of backpropagation networks," *Proc. 1988 Connectionist Models Summer School*, eds. D. Touretzky, G. Hinton and T. Sejnowski, Morgan Kaufmann, San Mateo, CA, 1988, pp. 120–126.
19. T. M. Martinetz, S. G. Berkovich and K. J. Schulten, "Neural-gas network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Networks* **4** (1993) 558–569.
20. T. M. Martinetz and K. J. Schulten, "Topology representing networks," *Neural Networks* **7**, 3 (1994) 507–522.
21. S. Mori, C. Y. Suen and K. Yamamoto, "Historical review of OCR research and development," *Proc. IEEE* **80** (1992) 1029–1058.
22. U. Pal, *On the Development of an Optical Character Recognition (OCR) System for Printed Bangla Script*, Ph.D. Thesis, Indian Statistical Institute, Kolkata, India, 1997.
23. R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: a comprehensive survey," *IEEE Trans. Patt. Anal. Mach. Intell.* **22** (2000) 63–84.
24. D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation," Institute for Cognitive Science Report 8506, University of California, San Diego, 1985.
25. M. Sabourin and A. Mitiche, "Modeling and classification of shape using a Kohonen's associative memory with selective multiresolution," *Neural Networks* **6** (1993) 275–283.
26. S. N. Srihari, E. Cohen, J. J. Hull and L. Kuan, "A system to locate and recognize ZIP codes in handwritten addresses," *IJRE* **1** (1989) 37–45.
27. S. N. Srihari, "Recognition of handwritten and machine printed text of postal address interpretation," *Patt. Recogn. Lett.* **14** (1993) 291–302.
28. S. N. Srihari and S. W. Lam, "Character recognition," Technical Report, State University of New York at Buffalo, available at "www.cedar.buffalo.edu/Publications/TechReps/OCR/ocr.html".
29. C. Y. Suen, "Computer recognition of unconstrained handwritten numerals," *Proc. IEEE* **80** (1992) 1162–1180.
30. P. Uma and S. Kalyanaraman, "Script processing and computer based lexicons of South Asian languages: a perspective on Kalyan Saraswati multi-media operating system," *Proc. Conf. Information Technology, Application in Language, Script and Speech*, New Delhi, India, 1994, pp. 7–38.

**Ujjwal Bhattacharya** received the M.Sc. and M. Phil., both in pure mathematics, in 1989 and 1990, respectively, from the Calcutta University. He obtained his post-graduate diploma in computer applications in 1991 from the Regional Computer Center, Calcutta. In 1991, he joined the Indian Statistical Institute, Calcutta, as a research scholar. Currently, he is working as a programmer in the Computer Vision and Pattern Recognition Unit of this Institute. In 1995, he received the Young Scientist Award in computer science from the Indian Science Congress Association. He is a member of the IEEE. He visited the Inter-University Center for Astronomy and Astrophysics, Pune and Institute of Biomathematics and Biometry, GSF, Munich during the year 1996 and 1998–1999, respectively. He has several research publications in international journals and conference proceedings.

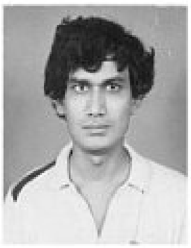
His research interests include pattern recognition, image processing, artificial neural networks and genetic algorithms.

**Amitava Datta** received his Master's degree in statistics from the Indian Statistical Institute in 1977. After working for a few years in industries, he joined the Indian Statistical Institute as a system analyst in 1988. Since then, he has been working in image processing and pattern recognition. From 1991 to 1992, he visited GSF, Munich, as a Guest Scientist and worked on a query-based decision support system. He received his Ph.D. degree from the Indian Statistical Institute in 2000.

His current research interests are in neural network based image processing and pattern recognition.

**Tanmoy Kanti Das** graduated with honours in physics from Calcutta University, India in 1993. He obtained his Master's degree in computer applications in 1999 from the Indira Gandhi National Open University, India. In that year, Mr. Das joined Indian Statistical Institute as a project supported Research Personnel. Since then, Mr. Das is working in the same institute under different projects. He also has several research publications.

His research interests include data mining, digital watermarking, optical character recognition, etc.



**Swapan Kumar Parui** received his Master's degree in statistics and his Ph.D. degree from the Indian Statistical Institute in 1975 and 1986, respectively. From 1985 to 1987, he held a post-doctoral position in Leicester Polytechnic, England, working on an automatic industrial inspection project, and from 1988 to 1989, he was a visiting scientist in GSF, Munich working on biological image processing. He has been with the Indian Statistical Institute from 1987, and is now a Professor.

Prof. Parui has published over 60 papers in journals and conference proceedings.

His current research interests include shape analysis, statistical pattern recognition, neural networks and computational geometry.

**Bidyut Baran Chaudhuri** received his B.Sc. (Hons.), B.Tech. and M.Tech. degrees from Calcutta University, India in 1969, 1972 and 1974, respectively and Ph.D. degree from Indian Institute of Technology, Kanpur in 1980. He joined the Indian Statistical Institute in 1978 where he served as the Project Co-ordinator and Head of National Nodal Center for Knowledge Based Computing. Currently, he is the Head of Computer Vision and Pattern Recognition Unit of the institute.

He has published 150 research papers in reputed international journals and has authored the books entitled *Two Tone Image Processing and Recognition* (Wiley Eastern, 1993) and *Object Oriented Programming: Fundamentals and Applications* (Prentice Hall, 1998). He was awarded Sir J. C. Bose Memorial Award for best Engineering Science oriented paper in 1986, M. N. Saha Memorial Award (twice) for best application oriented papers in 1989 and 1991, the Homi Bhabha Fellowship award in 1992 for OCR of the Indian Languages and computer communication for the blind, Dr. Vikram Sarabhai Research Award in 1995 for his outstanding achievements in the fields of Electronics, Informatics, and Telematics, and C. Achuta Menon Prize in 1996 for computer based Indian language processing. He worked as a Leverhulme visiting fellow at Queen's University, UK, in 1981–1982, a visiting scientist at GSF, Munich and guest faculty at the Technical University of Hanover during 1986–1988. He is a Fellow of IEEE, member secretary (Indian Section) of International Academy of Sciences, Fellow of International Association of Pattern Recognition, Fellow of National Academy of Sciences (India) and Fellow of the Indian National Academy of Engineering. He is serving as an associate editor of the journals, *Pattern Recognition*, *Pattern Recognition Letters* and *Vivek*. He acted as guest editor of a special issue on Fuzzy Systems for *IETE*.

His research interests include pattern recognition, image processing, computer vision, natural language processing and digital document processing including OCR.