Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals

Sung-Bae Cho

Abstract— **Artificial neural networks have been recognized as a powerful tool for pattern classification problems, but a number of researchers have also suggested that straightforward neural-network approaches to pattern recognition are largely inadequate for difficult problems such as handwritten numeral recognition. In this paper, we present three sophisticated neural-network classifiers to solve complex pattern recognition problems: multiple multilayer perceptron (MLP) classifier, hidden Markov model (HMM)/MLP hybrid classifier, and structure-adaptive self-organizing map (SOM) classifier. In order to verify the superiority of the proposed classifiers, experiments were performed with the unconstrained handwritten numeral database of Concordia University, Montreal, Canada. The three methods have produced 97.35%, 96.55%, and 96.05% of the recognition rates, respectively, which are better than those of several previous methods reported in the literature on the same database.**

Index Terms—**Handwritten numeral recognition, multiple neural networks, hidden Markov models, hybrid classifiers, self-organizing feature maps.**

I. Introduction

UNTIL today, a wide variety of methods have been proposed to realize the perfect recognizer of handwritten numerals by computer. Many systems have been developed, but more work is still required to be able to match human performance [1]. Recently, on the other hand, the emerging technology of neural networks has largely exploited to implement a system toward a pattern recognizer of such level.

Among several models, the multilayer perceptron (MLP) and Kohonen's self-organizing map (SOM) have been most frequently used as a powerful tool for pattern classification problems. Their strength is in the discriminative power and the capability to learn and represent implicit knowledge, but they also have faced to several difficulties in real-world problems.

Once one fixes the structure of the network, the network adjusts its weights via the learning rule until the optimal weights are obtained. The corresponding weights along with the structure of the network create the decision boundaries in the feature space. In many practical pattern recognition problems, however, this usual neural-network classifier tends not to converge to its solution state. Even if the network converges, the time required for convergence may be too prohibitive for practical purposes.

In this paper, we present three sophisticated neural-network classifiers to recognize the totally unconstrained handwritten numerals. Two of them are based on the MLP classifiers [multiple MLP classifier and hidden Markov model (HMM)/MLP hybrid classifier], and another, the structure-adaptive SOM classifier, based on the SOM classifier, which can adapt its structure as well as its weights.

The rest of this paper is organized as follows. In Section II, we give some background information on this work, such as the related works on the handwritten numeral recognition, the database used for the experiments, and the feature extraction methods used. Section III shows that the MLP can be formulated as a Bayesian framework, thereby making the connection to the statistical pattern classification. And then we present what the multiple MLP classifier is along with the possible combination methods. In Sections IV and V we illustrate the HMM/MLP hybrid classifier and the structure-adaptive SOM classifier. In order to investigate the performance of the presented classifiers, experimental results with the unconstrained handwritten numeral database of Concordia University, Montreal, Canada, are provided in Section VI.

II. Backgrounds

A. Related Works

In the past several decades, a wide variety of approaches have been proposed to attempt to achieve the recognition system of handwritten numerals. These approaches generally fall into two categories: statistical method and syntactic method [1]. First category includes techniques such as template matching, measurements of density of points, moments, characteristic loci, and mathematical transforms. In the second category, efforts are aimed at capturing the essential shape features of numerals, generally from their skeletons or contours. Such features include loops, endpoints, junctions, arcs, concavities and convexities, and strokes.

Table I shows the performances of some of the most reliable handwritten numeral recognition systems found in the literature. It also provides information about the size of the data sets used for training and testing along with the scanning resolution in PPI (pixels per inch). It is important to realize that recognition systems cannot be compared simply by their reported performances since most systems are still tested on databases with very different characteristics.

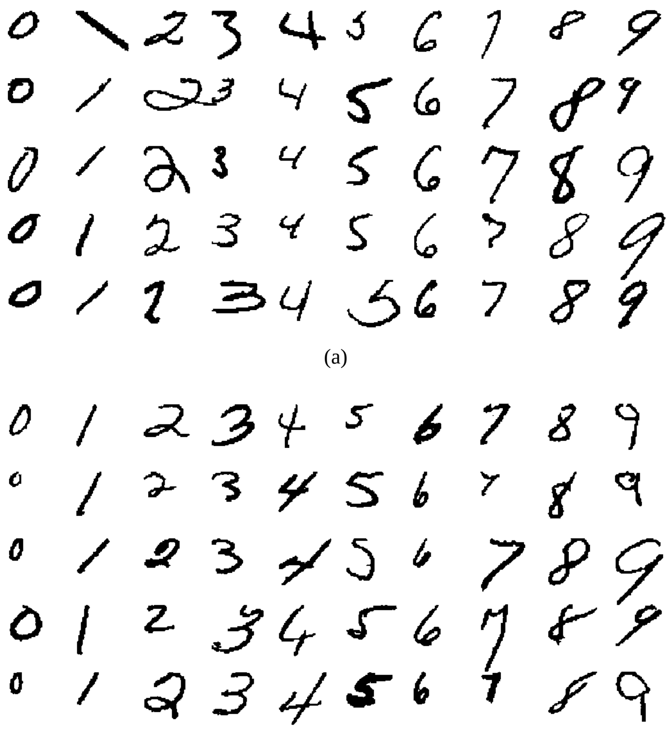
Manuscript received January 20, 1996; revised June 19, 1996. This work was supported in part by Grant 961–0901–009–2 from the Korean Science and Engineering Foundation (KOSEF).

The author is with the Department of Computer Science, Yonsei University, Seoul 120-749, Korea.

Publisher Item Identifier S 1045-9227(97)00239-7.

TABLE I
COMPARISONS OF THE BEST RESULTS IN LITERATURE (%)

Methods	Correct	Error	Training	Testing	PPI
Ahmed [2]	87.85	12.15	5000	3540	166
Beun [3]	90.87	9.13	15000	10000	
Cohen [4]	95.54	4.46		2711	300
Cohen [4]	97.10	2.90		1762	300
Duerr [5]	99.50	0.50	5000	5000	
Gader [6]	96.35	3.65		6000	166
Gader [6]	98.20	1.80		2219	300
Knerr [8]	90.30	9.70	7200	1800	
Krzyzak [9]	86.40	13.60	4000	2000	166
Krzyzak [9]	94.85	5.15	4000	2000	166
Kuan [10]	93.30	6.70	1820	7100	300
Lam [11]	93.10	6.90	4000	2000	166
Le Cun [12]	90.00	10.00	7291	2007	300
Le Cun [12]	92.00	8.00	7291	2007	300
Kim [7]	95.40	4.60	4000	2000	166
Kim [7]	95.85	4.15	4000	2000	166
Legault [13]	93.90	6.10	4000	2000	166
Lemarie [14]	97.97	2.03	8783	7394	
Mai [15]	92.95	7.05	4000	2000	166
Mitchell [16]	87.95	12.05		2103	
Nadal [17]	86.05	13.95	4000	2000	166
Stringa [18]	92.60	7.40	19377	19377	
Suen [19]	93.05	6.95	4000	2000	166



(a)

(b)

Fig. 1. Sample data for (a) training and (b) test.

B. Database Used

In this paper, we have used the handwritten numeral database of Concordia University, Montreal, Canada, which consists of 6000 unconstrained numerals originally collected from dead letter envelopes by the U.S. Postal Service at different locations in the United States. The numerals of this database were digitized in bilevel on a 64 × 224 grid of 0.153 mm square elements, giving a resolution of approximately 166 PPI [19]. Among the data, 4000 numerals were used for training and 2000 numerals for testing. Fig. 1 shows some representative samples taken from the database. We can see that many different writing styles are apparent, as well as numerals of different sizes and stroke widths.

A_0	A_1	A_2
A_7	(i, j)	A_3
A_6	A_5	A_4

Fig. 2. Definition of eight neighbors A_k $(k = 0, 1, \cdots, 7)$ of pixel (i, j).

$$\begin{array}{|c|c|c|}\hline 5 & 5 & 5 \\ \hline -3 & 0 & -3 \\ \hline -3 & -3 & -3 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -3 & -3 & -3 \\ \hline -3 & 0 & -3 \\ \hline 5 & 5 & 5 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline -3 & 5 & 5 \\ \hline -3 & 0 & 5 \\ \hline -3 & -3 & -3 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -3 & -3 & -3 \\ \hline 5 & 0 & -3 \\ \hline 5 & 5 & -3 \\ \hline \end{array}$$

(a) (c)

$$\begin{array}{|c|c|c|}\hline -3 & -3 & 5 \\ \hline -3 & 0 & 5 \\ \hline -3 & -3 & 5 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 5 & -3 & -3 \\ \hline 5 & 0 & -3 \\ \hline 5 & -3 & -3 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline -3 & -3 & -3 \\ \hline -3 & 0 & 5 \\ \hline -3 & 5 & 5 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 5 & 5 & -3 \\ \hline 5 & 0 & -3 \\ \hline -3 & -3 & -3 \\ \hline \end{array}$$

(b) (d)

Fig. 3. Kirsch masks used for extracting four directional features: (a) horizontal direction, (b) vertical direction, (c) right-diagonal direction, and (d) left-diagonal direction.

C. Feature Extraction

Numerals, whether handwritten or typed, are essentially line drawings, i.e., one-dimensional structures in a two-dimensional space. Thus, local detection of line segments seems to be an adequate feature extraction method. For each location in the image, information about the presence of a line segment of a given direction is stored in a feature map [8]. Especially, in this paper Kirsch masks have been used for extracting directional features [7].

Kirsch defined a nonlinear edge enhancement algorithm as follows [20]:

$$G(i, j) = \max\{1, \max_{k=0}^{7}[|5S_k - 3T_k|]\} \tag{1}$$

where

$$S_k = A_k + A_{k+1} + A_{k+2} \tag{2}$$

$$T_k = A_{k+3} + A_{k+4} + A_{k+5} + A_{k+6} + A_{k+7}. \tag{3}$$

Here, $G(i, j)$ is the gradient of pixel (i, j), the subscripts of A are evaluated modulo 8, and A_k $(k = 0, 1, \cdots, 7)$ is eight neighbors of pixel (i, j) defined as shown in Fig. 2.

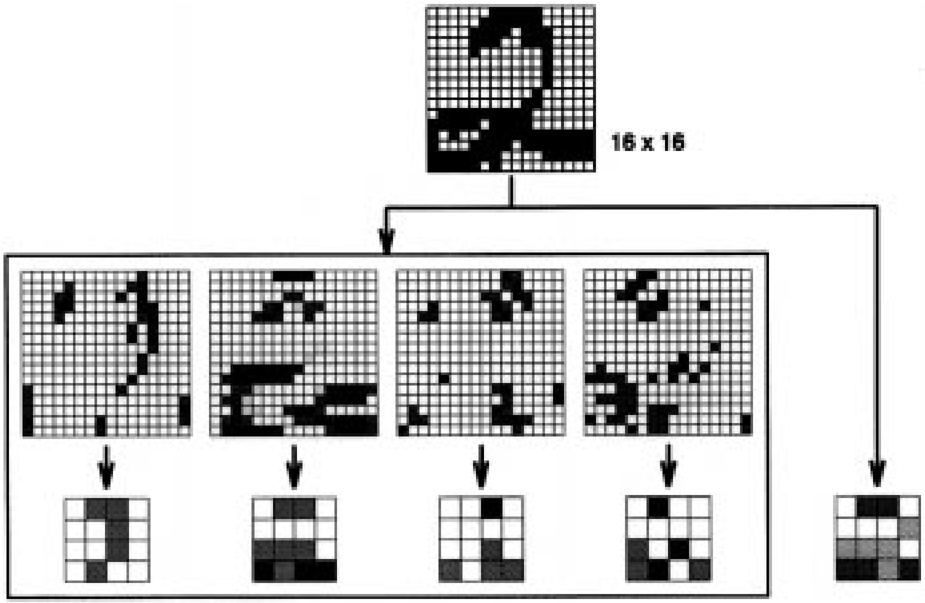


Fig. 4. Overall process of extracting features: four 4 × 4 local features from Kirsch masks and one 4 × 4 global feature from compressed image.

In this paper, input pattern is size-normalized by 16 × 16 and then directional feature vectors for horizontal (H), vertical (V), right-diagonal (R), and left-diagonal (L) directions are calculated from the size-normalized image as follows:

$$\begin{aligned} G(i,j)_{\mathrm{H}} &= \max(|5S_0 - 3T_0|, |5S_4 - 3T_4|) \\ G(i,j)_{\mathrm{V}} &= \max(|5S_2 - 3T_2|, |5S_6 - 3T_6|) \\ G(i,j)_{\mathrm{R}} &= \max(|5S_1 - 3T_1|, |5S_5 - 3T_5|) \\ G(i,j)_{\mathrm{L}} &= \max(|5S_3 - 3T_3|, |5S_7 - 3T_7|). \end{aligned} \tag{4}$$

As a final step in extracting directional features, each 16 × 16 directional feature vector is compressed to 4 × 4 feature vector. Fig. 3 shows the Kirsch masks used for calculating directional feature vectors.

Moreover, 4 × 4 compressed image can be considered as a good candidate for global features. In addition to those two features, we have also used a contour feature: 15 complex Fourier descriptors from the outer contours and simple topological features from the inner contours.

As a result, available features include five 4 × 4 features (four 4 × 4 local features and one 4 × 4 global feature) and structural features extracted from the contours of the numerals. Fig. 4 shows the schematic diagram of the steps for extracting the former features.

III. Multiple MLP Classifier

There has been a tremendous growth in the complexity of the recognition, estimation, and control problems expected from neural networks. In solving these problems, we are faced with a large variety of learning algorithms and a vast selection of possible network architectures. After all the training, we choose the best network with a winner-take-all cross-validatory model selection. However, recent theoretical and experimental work indicates that we can improve performance by considering methods for combining multiple neural networks [21]–[25]. In the following, we shall briefly introduce the MLP as a pattern classifier and describe how to boost the performance by combining them.

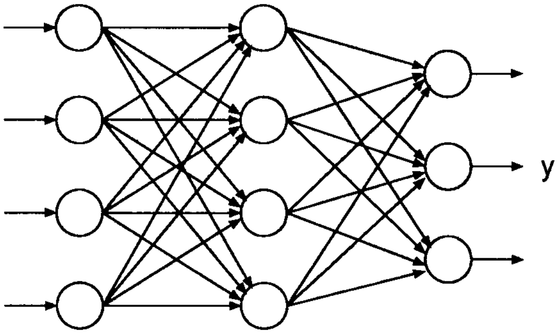


Fig. 5. A two-layered MLP architecture.

A. MLP Classifier

Fig. 5 shows a two-layered neural network. The network is fully connected between adjacent layers. The operation of this network can be thought of as a nonlinear decision-making process. Given an unknown input $X = (x_1, x_2, \cdots, x_T)$ and the output set $\Omega = \{\omega_1, \omega_2, \cdots, \omega_c\}$, each output node yields the output y_i of belonging to this class by

$$y_i = f\left\{\sum_k w_{ik}^{om} f\left(\sum_j w_{kj}^{mi} x_j\right)\right\} \tag{5}$$

where w_{kj}^{mi} is a weight between the jth input node and the kth hidden node, w_{ik}^{om} is a weight from the kth hidden node to the ith class output, and f is a sigmoid function such as $f(x) = 1/(1+e^{-x})$. The node having the maximum value is selected as the corresponding class.

The outputs of the MLP as shown in the above are not just likelihoods or binary logical values near zero or one. Instead, they are estimates of Bayesian *a posteriori* probabilities [26], [27]. With a squared-error cost function, the network parameters are chosen to minimize the following:

$$E\left[\sum_{i=1}^{c}(y_i(X)-d_i)^2\right] \tag{6}$$

where $E[\cdot]$ is the expectation operator, $\{y_i(X)\colon i=1,\cdots,c\}$ the outputs of the network, and $\{d_i\colon i=1,\cdots,c\}$ the desired outputs for all output nodes. Performing several treatments in this formula allows it to cast in a form commonly used in statistics that provides much insight as to the minimizing values for $y_i(X)$ [26]

$$E\left[\sum_{i=1}^{c}(y_i(X)-E[d_i|X])^2\right]+E\left[\sum_{i=1}^{c}\text{var}\,[d_i|X]\right] \tag{7}$$

where $E[d_i|X]$ is the conditional expectations of d_i and $\text{var}[d_i|X]$ is the conditional variance of d_i.

Since the second term in (7) is independent of the network outputs, minimization of the squared-error cost function is achieved by choosing network parameters to minimize the first expectation term. This term is simply the mean-squared error between the network outputs $y_i(X)$ and the conditional expectation of the desired outputs. For a "1" of M problem, d_i equals one if the input X belongs to class ω_i and zero otherwise. Thus, the conditional expectations are the following:

$$\begin{aligned}E[d_i|X] &= \sum_{j=1}^{c} d_i P(\omega_j|X)\\ &= P(\omega_i|X)\end{aligned} \tag{8}$$

which are the Bayesian probabilities. Therefore, for a "1" of M problem, when network parameters are chosen to minimize a squared-error cost function, the outputs estimate the Bayesian probabilities so as to minimize the mean-squared estimation error.

B. Multiple Classifier

The networks train on a set of example patterns and discover relationships that distinguish the patterns. A network of a finite size does not often load a particular mapping completely or it generalizes poorly. Increasing the size and number of hidden layers most often does not lead to any improvements. Furthermore, in complex problems such as character recognition, both the number of available features and the number of classes are large. The features are neither statistically independent nor unimodally distributed. Therefore, if we could make the network consider the only specific part of the complete mapping, it would perform its job better.

The basic idea of the multiple network classifier is to develop n independently trained neural networks with particular features, and to classify a given input pattern by obtaining a classification from each copy of the network and then using a consensus scheme to decide the collective classification by utilizing combination methods [21] (see Fig. 6). Two general approaches, one based on fusion techniques and the other on voting techniques, form the basis of the methods presented.

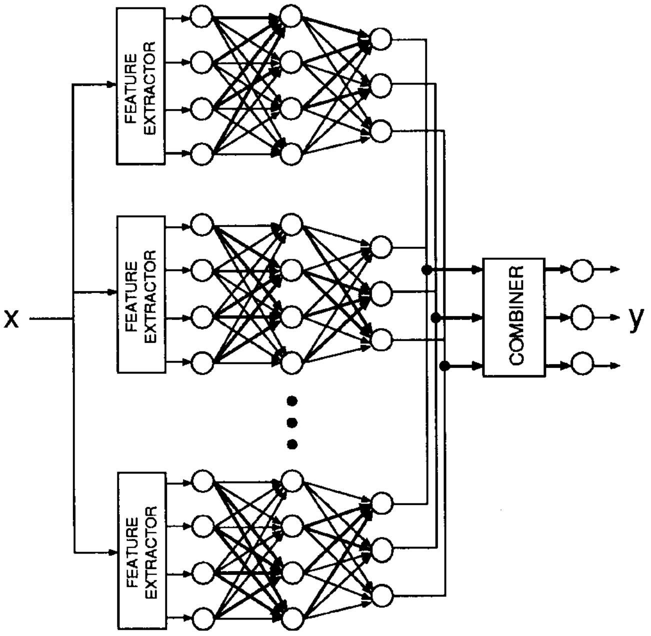


Fig. 6. The multiple MLP classifier with consensus scheme. n independently trained neural networks classify a given input pattern by using a consensus method to decide the collective classification.

There have been proposed various neural-network optimization methods based on combining estimates, such as boosting, competing experts, ensemble averaging, metropolis algorithms, stacked generalization, and stacked regression. A general result from the previous works is that averaging separate networks improves generalization performance for the mean squared error. If we have networks of different accuracy, however, it is obviously not good to take their simple average or simple voting.

To give a solution to the problem, we have developed a fusion method that considers the difference of performance of each network in combining the networks, which is based on the notion of fuzzy logic, especially the fuzzy integral [28], [29]. This method combines the outputs of separate networks with importance of each network, which is subjectively assigned as the nature of fuzzy logic.

The fuzzy integral introduced by Sugeno and the associated fuzzy measures provide a useful way for aggregating information. Using the notion of fuzzy measures, Sugeno developed the concept of the fuzzy integral, which is a nonlinear functional that is defined with respect to a fuzzy measure, especially g_λ-fuzzy measure.

Definition 1: Let X be a finite set and $h\colon X\rightarrow[0,1]$ be a fuzzy subset of X. The fuzzy integral over X of the function h with respect to a fuzzy measure g is defined by

$$h(x)\circ g(\cdot)=\max_{E\subseteq X}[\min(\min_{x\in E}h(x),g(E))]. \tag{9}$$

The calculation of the fuzzy integral with respect to a g_λ-fuzzy measure would only require the knowledge of the density function, where the ith density g^i is interpreted as the degree of importance of the source y_i toward the final evaluation. These densities can be subjectively assigned by an expert or can be generated from data. The value obtained from comparing the evidence and the importance in terms of the $\min$ operator is interpreted as the grade of agreement between real possibilities $h(y)$ and the expectations g. Hence, fuzzy integration is interpreted as searching for the maximal grade of agreement between the objective evidence and the expectation. For further information, see [29].

IV. HMM/MLP Hybrid Classifier

Though MLP has been recognized as powerful for pattern classification problems, current neural-network topologies are inefficient in modeling temporal structures. An alternative approach to sequence recognition is to use HMM's. The HMM provides a good probabilistic representation of temporal sequences having large variations and has been widely used for automatic speech recognition.

The main drawback of an HMM-based recognizer trained independently, however, is the weak discriminative power. The maximum likelihood estimation procedures typically used for training HMM can be suitable to model the time sequential order and variability of input observation sequences, but the recognition task requires more powerful discrimination.

This section presents a classifier in which HMM's provide an MLP with input vectors through which the temporal variations are filtered. This classifier takes the likelihoods inside the HMM's of all class models and presents them to an MLP to estimate posterior probabilities better. To evaluate the performance of the hybrid classifier, we utilize the contour features of handwritten numerals.

A. Hidden Markov Models

An HMM can be thought of as a directed graph consisting of N nodes (states) and arcs (transitions) representing the relationships between them. We denote the state at time t as q_t, and an observation sequence as $X = (x_1, x_2, \cdots, x_T)$, where each observation x_t is one of the observation symbols and T is the number of observations in the sequence. Each node stores the initial state probability $\pi = \{\pi_i | \pi_i = P(q_1 = i), i = 1, 2, \cdots, N\}$ and the observation symbol probability distribution $B = \{b_j(X_t) | b_j(X_t) =$ *a posteriori* probability of observation X_t given $q_t = j\}$, and each arc contains the state transition probability distribution $A = \{a_{ij} | a_{ij} = P(q_{t+1} = j | q_t = i), i, j = 1, 2, \cdots, N\}$. Using these parameters, the observation sequence can be modeled by an underlying Markov chain whose state transitions are not directly observable.

Given a model $\lambda_i = (A, B, \pi)$ and an unknown input sequence $X = (x_1, x_2, \cdots, x_T)$, the matching score is obtained by summing the probability of a sequence of observation X generated by the model over all possible state sequences giving

$$P(X|\lambda_i) = \sum_{q_1, q_2, \cdots, q_T} \pi_{q_1} b_{q_1}(x_1) \prod_{t=2}^{T} a_{q_{t-1}q_t} b_{q_t}(x_t). \tag{10}$$

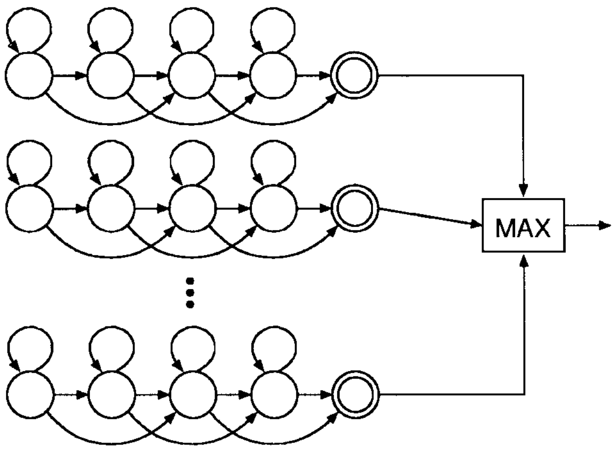


Fig. 7. Schematic diagram of an HMM-based recognizer.

Then, we select the maximum as

$$i^* = \arg \max_i P(X|\lambda_i), \qquad 1 \le i \le c \tag{11}$$

and classify the input sample as class i^*. A schematic diagram of the HMM based recognizer is shown in Fig. 7.

For a given λ_i, an efficient method for computing (10), known as the forward–backward algorithm, is as follows.

- Initialization:

$$\alpha_1(k) = \pi_k b_k(x_1), \qquad 1 \le k \le N. \tag{12}$$

- Induction:

$$\alpha_{t+1}(j) = \left[\sum_{k=1}^{N} \alpha_t(k) a_{kj}\right] b_j(x_{t+1}), \qquad 1 \le t \le T-1. \tag{13}$$

Then the matching score can be calculated by

$$P(X|\lambda_i) = \sum_{k=1}^{N} \alpha_T(k). \tag{14}$$

Notice that in this equation the score for a model is computed as a sum over all states of the model, but it is usual to specify distinguished final states for each model. In that case, the score is amount to the sum of the forward variables $\alpha_T(k)$ at the final states.

B. The Hybrid Classifier

The key idea in the proposed HMM/MLP classifier is 1) to convert a dynamic input sample to a static pattern sequence by using HMM-based recognizer and 2) to recognize the sequence by using an MLP-trained classifier. A block diagram of the hybrid classifier is shown in Fig. 8.

A usual HMM-based recognizer assigns one Markov model for each class. Recognition with HMM's involves accumulating scores for an unknown input across the nodes in each class model, and selecting that class model which provides the maximum accumulated score. On the contrary, the proposed classifier replaces the maximum-selection part with an MLP classifier.

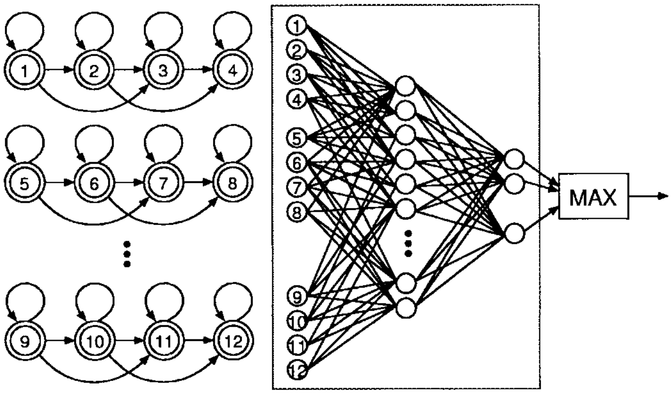


Fig. 8. The HMM/MLP hybrid classifier.

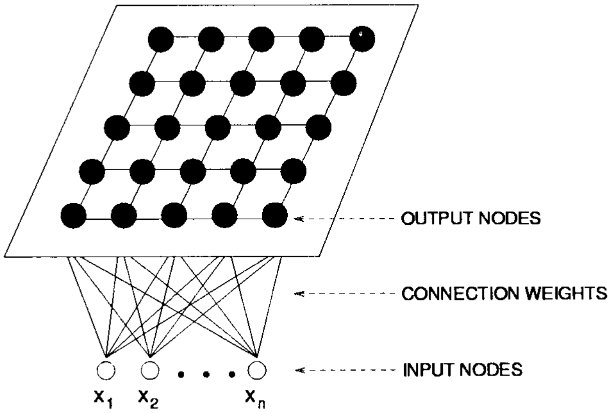


Fig. 9. Kohonen's self-organizing map.

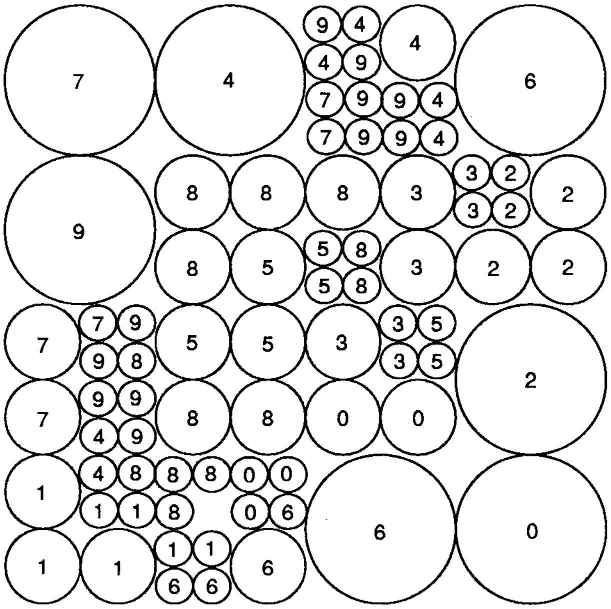


Fig. 10. Doubly self-organizing neural network. The figure in each circle means the class that the corresponding node represents.

The hybrid classifier takes the likelihood patterns inside the HMM's and presents them to an MLP to estimate posterior probabilities of class ω_i as follows:

$$P(\omega_i|X) \approx f\left\{\sum_{k=1}^{H} w_{ik}^{om} f\left(\sum_{j=1}^{T}\sum_{l=1}^{N} w_{kjl}^{mi}\alpha_T(j,l)\right)\right\} \tag{15}$$

where the w_{kjl}^{mi} is a weight from the jth input node at the lth state to the kth hidden node, w_{ik}^{om} is a weight from the kth hidden node to the ith class output, and f is a sigmoid function such as $f(x) = 1/(1+e^{-x})$. Here, $\alpha_T(j,l)$ is the value of the forward variable $\alpha_T(l)$ at the jth HMM class model. Rather than simply selecting the model producing the maximum value of $P(X|\lambda_j)$, the proposed classifier have an MLP perform additional classification with all the likelihood values inside HMM's. In this classifier, the HMM yields a kind of static pattern of which the inherent temporal variations have been processed and the MLP classifier discriminates them as belonging to one particular class.

The hybrid classifier automatically focuses on those parts of the model which are important for discriminating between sequentially similar patterns. In the conventional HMM-based approach, only the patterns in the specified class are involved in the estimation of parameters; there is no role for any patterns in the other classes. The hybrid classifier uses more information than the conventional approach; it uses knowledge of the potential confusions in the particular training data to be recognized. Since it uses more information, there are certainly reasons to suppose that the hybrid classifier will prove superior to the conventional approach. In this classifier, the MLP will learn prior probabilities as well as to correct the assumptions made on the probability density functions used in the HMM's.

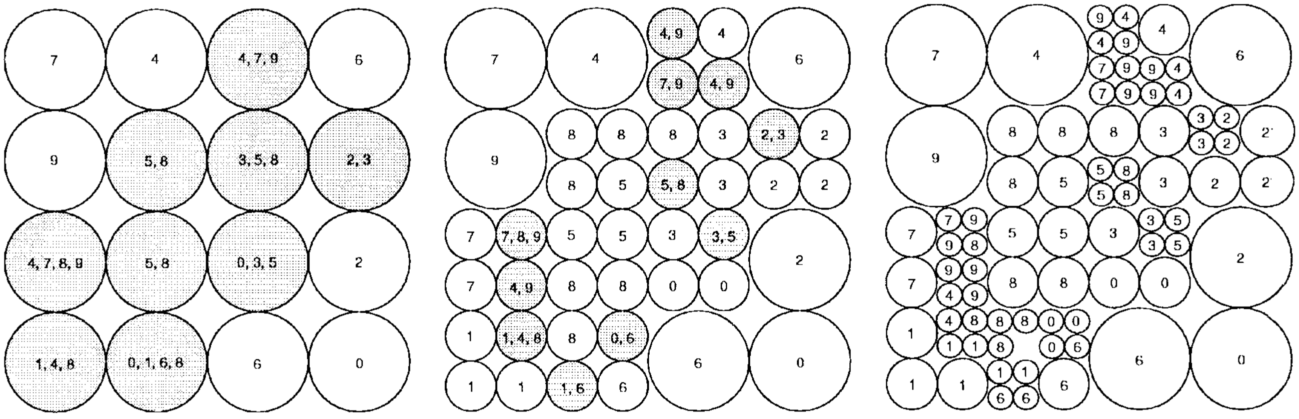
V. Structure-Adaptive SOM Classifier

A. K-Means Algorithm

Assume a sequence of samples of a vectorial observable $x = x(t) \in R^n$, where t is the time coordinate, and a set of variable reference vectors $\{w_i(t)\colon w_i \in R^n, i = 1, 2, \cdots, k\}$. If the $w_i(0)$ have been initialized in some proper way and $x(t)$ can somehow be simultaneously compared with each $w_i(t)$ at each successive instant of time, the best-matching $w_i(t)$ is to be updated to match even more closely the current $x(t)$. In this way the different reference vectors tend to become specifically tuned to different domains of the input variable x.

In general, however, no closed-form solution for the optimal placement of the w_i is possible and iterative approximation schemes must be used. By the way, it often turns out to be more economical to first observe a number of training samples $x(t)$, which are labeled according to the closest vectors w_i, and then to perform the updating operation in a single step. For the new vector w_i, the average is taken of those $x(t)$ that were identified with vector i. This algorithm, termed the *k-means algorithm*, is widely used in pattern recognition, especially for pattern clustering.

It has been pointed out in several previous works that Kohonen's SOM is an iterative version of the *k-means algorithm*, although SOM has a lot of intrinsic merits that a



(a) (b) (c)

Fig. 11. Map configurations changed through learning. (a) Initial status. (b) Intermediate status. (c) Final status.

neural-network model usually possesses. Therefore, it is not appropriate to use the SOM for classification problems because decision accuracy cannot be fine-tuned with the conventional SOM. Also, it is quite difficult to determine the size and structure of the network. To overcome these difficulties, several approaches based on the structure adaptation of the networks have been recently proposed [30]–[32].

B. Structure-Adaptive SOM

In this section we present a structure-adaptive self-organizing neural network which is able to simultaneously determine a suitable number of nodes and the connection weights between input and output nodes. The basic idea is very simple.

1) Start with a basic neural network (in our case, 4 × 4 map of which each node is fully connected to all input nodes).
2) Train the current network with the Kohonen's algorithm [33].
3) Calibrate the network using known input–output patterns to determine
   a) which node should be replaced with a submap of several nodes (in our case, 2 × 2 map) and
   b) which node should be deleted.
4) Unless every node represents a unique class, go to 2).

Note that step 3) positions the node in regions where the current network does not produce a unique label for the classification. In our model, the weights of new nodes are interpolated from those of neighboring nodes.

C. Network Structure and Adaptation

The structure of the network is very similar to Kohonen's SOM shown in Fig. 9 except the irregular connectivity in the map. Fig. 10 shows an instance of the network where each node represents a unique class. Every node is connected to all the input nodes with corresponding weights. (Actually, this is the final network structure obtained for recognizing the handwritten numerals in our simulation.) The initial map of the network consists of 4 × 4 nodes. The weight vector of node i shall be denoted by $w_i \in R^n$.

The simplest analytical measure for the match of x with the w_i may be the inner product $x^T w_i$, which is based on the *Euclidean distance* between x and w_i. The minimum distance defines the winner w_c. If we define a neighborhood set N_c around node c, at each learning step all the nodes within N_c are updated, whereas nodes outside N_c are left intact. This neighborhood is centered around that node for which the best match with input x is found as

$$\|x - w_c\| = \min_i \{\|x - w_i\|\}. \tag{16}$$

The width or radius of N_c can be time-variable. For a good global ordering, it is advantageous to let N_c be very wide in the beginning and shrink monotonically with time [33].

The updating process may read

$$w_i(t+1) = \begin{cases} w_i(t) + \alpha(t)[x(t) - w_i(t)] & \text{if } i \in N_c(t) \\ w_i(t) & \text{if } i \notin N_c(t) \end{cases} \tag{17}$$

where $\alpha(t)$ is a learning rate $0 < \alpha(t) < 1$.

D. Insertion of New Nodes

After a constant number of adaptation steps, a node representing more than one class is replaced with several nodes. (In our case, we have used a submap of 2 × 2 nodes.) Obviously, this node lies in a region of the input vector space where many misclassifications occur. If input patterns from different classes are covered by the same local node and activate this node to about the same degree, it might be the case where their vectors of local node activations are nearly identical.

Fig. 11 shows how the network structure changes as some nodes representing duplicated classes are replaced by several nodes having finer resolution.

E. Deletion of Nodes

The previous section gives us the way how to extend the network structure. A necessary consequence thereof is that all

TABLE II
THE RESULT OF RECOGNITION RATES (%)

Methods	Recognized	Substituted	Rejected	Reliability
MLP_1	89.05	7.00	3.95	92.71
MLP_2	95.40	3.75	0.85	96.22
MLP_3	93.95	4.10	1.95	95.82
Voting	96.70	3.05	0.25	96.94
Average	97.15	2.35	0.50	97.64
Fuzzy	97.35	2.30	0.35	97.69

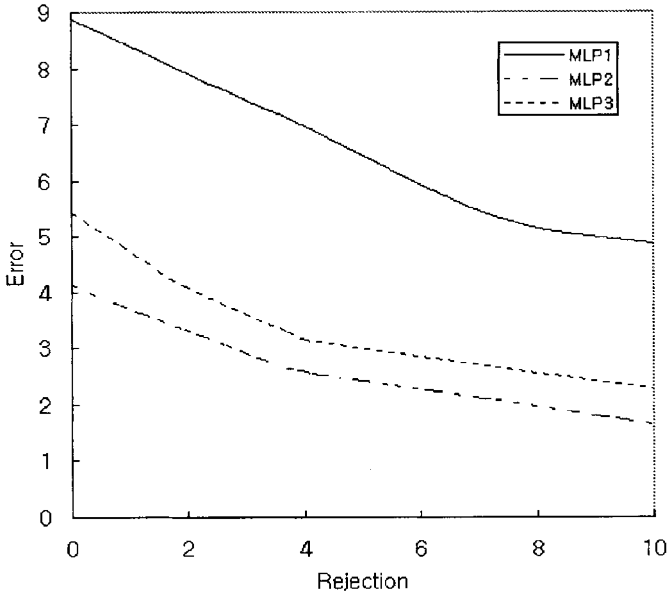


Fig. 12. Rejection-versus-error curves (1).

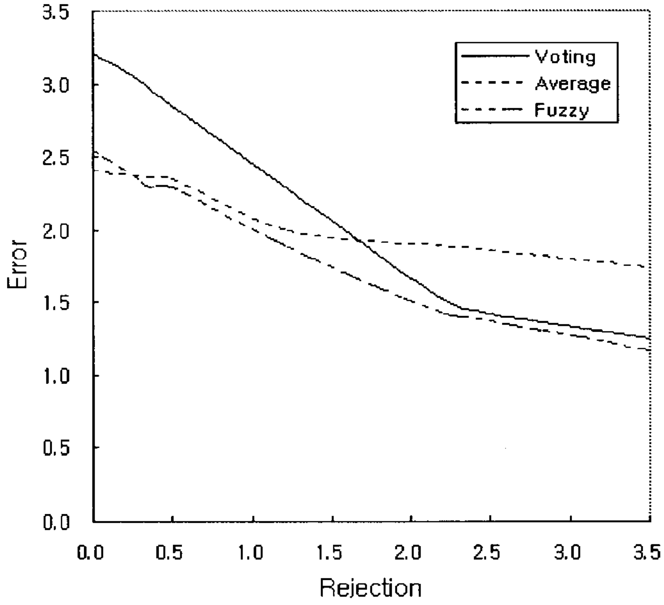


Fig. 13. Rejection-versus-error curves (2).

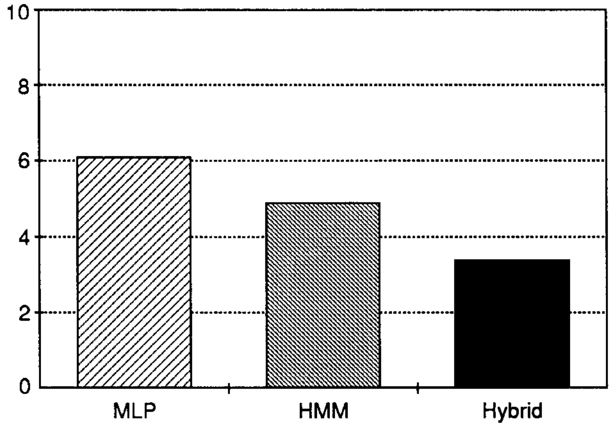


Fig. 14. A comparison of the error rates of MLP, HMM, and the hybrid classifier.

the nodes are connected directly or indirectly to each other. However, a problem may occur if the pattern space we try to discriminate has some disconnected regions. A solution can be found by introducing the deletion of nodes from the structure. An obvious criterion for a node to be deleted would be that it has a position in an area of the R^n where the probability density is zero. For this purpose, we delete some nodes that do not activate for a long while. In our example, only one node is deleted at the final map. [See Fig. 11(c).]

VI. EXPERIMENTAL RESULTS

A. Multiple MLP Classifier

To evaluate the performance of the multiple MLP classifier, we have implemented three different networks, each of which is a two-layer neural network using different features. MLP_1, MLP_2, and MLP_3 have used the normalized image, Kirsch features, and the sequence of contour features, respectively. In this fashion each network makes the decision through its own criterion. Each of the three networks was trained with 4000 samples and tested on 2000 samples from the Concordia database.

The error backpropagation algorithm was used for the training and the iterative estimation process was stopped when an average squared error of 0.9 over the training set was obtained, or when the number of iteration reaches 1000, which was adopted mainly for preventing networks from overtraining. The parameter values used for training were: learning rate is 0.4 and momentum parameter is 0.6. An input vector is classified as belonging to the output class associated with the highest output activation.

Table II shows the recognition rates with respect to the three different networks and their combinations by utilizing consensus methods like majority voting, average, and the fuzzy integral. The reliability in the table is computed as the following equation:

$$\text{reliability} = \frac{\text{recognition rate}}{\text{recognition rate} + \text{error rate}} \times 100 \tag{18}$$

where the error rate is the portion of patterns which are classified incorrectly by the method. As can be seen, every method of combining multiple MLP produces better results than individual networks, and the overall classification rate for the fuzzy integral is higher than those for other consensus

TABLE III
COMPARISONS OF THE PRESENTED METHOD WITH THE RELATED (%)

Methods	Recognized	Substituted	Rejected	Reliability
Kim [7]	95.40	4.60	0.00	95.40
Kim [7]	95.85	4.15	0.00	95.85
Krzyzak [9]	86.40	1.00	12.60	98.85
Krzyzak [9]	94.85	5.15	0.00	94.85
Lam [11]	93.10	2.95	3.95	96.98
Legault [13]	93.90	1.60	4.50	98.32
Mai [15]	92.95	2.15	4.90	97.74
Nadal [17]	86.05	2.25	11.70	97.45
Suen [19]	93.05	0.00	6.95	100.00
Proposed method	96.05	3.95	0.00	96.05

TABLE IV
CONFUSION MATRIX FOR THE PROPOSED METHOD

Class	0	1	2	3	4	5	6	7	8	9	Recognized
0	197	0	1	0	0	0	1	0	1	0	98.5%
1	0	192	4	1	0	0	2	0	1	0	96.0%
2	0	0	189	3	2	1	0	1	3	1	94.5%
3	0	2	2	190	0	0	0	1	3	2	95.0%
4	0	3	0	0	193	0	1	1	1	1	96.5%
5	1	0	0	1	0	193	2	1	2	0	96.5%
6	2	1	0	0	0	2	191	0	4	0	95.5%
7	0	6	0	0	0	0	0	193	1	0	96.5%
8	1	0	2	2	0	2	5	0	188	0	94.0%
9	1	0	1	0	0	1	1	1	0	195	97.5%
Average											96.05%

methods. Fig. 12 and 13 provide rejection-versus-error curves that compare the results at the same levels of rejection.

B. HMM/MLP Hybrid Classifier

The next experiment is to recognize the same data set by HMM and the hybrid classifiers. For the HMM, we have implemented left–right model in which no transitions are allowed to states whose indexes are lower than that of the current state. It was composed of the ten nodes and the eight observation symbols in each node. The ten nodal matching scores of all models provided as inputs to the neural network part of the hybrid classifier.

In order to apply the presented hybrid classifier for the numeral recognition, we have implemented another two-layered MLP which has 100 input nodes, 20 hidden nodes, and ten output nodes. The input was provided by the ten HMM models consisting of ten nodes.

Fig. 14 compares the error rates of all the three methods. The overall recognition rate for the ten classes with hybrid classifier is 96.55%. This is a significant improvement over the performance obtained with the HMM trained with maximum likelihood (ML) optimization (93.95% recognition rate), as well as with the MLP using the direction sequences of digit contour as inputs (95.10% recognition rate).

In summary, the hybrid classifier gave a better discriminative capability over the conventional HMM classifiers. We may thus assert that these improvements are mainly due to the excellent discriminative capability of MLP. In the problem of recognizing off-line characters, however, it has to be considered seriously to devise a robust method for extracting sequential features before attempting to use the HMM-based method.

C. Structure-Adaptive SOM Classifier

Table III shows the performance of the presented method along with the results produced by some previous methods reported on the same database. Even though some of the previous methods produce relatively higher reliability, it must be acknowledged that it uses a highly tuned architecture. The error rate of the proposed classifier is 3.95%, which is a big improvement compared with those of the previous methods, but in terms of the reliability this result cannot be said as excellent. Further work is in progress toward emphasizing this aspect by introducing the reject criteria to the decision process.

Table IV reports the confusion matrix for the proposed method with respect to the data set. It can be seen from this table that most of the confusion makes sense: For example, "0" has three instances of misclassification, as "2," "6," and "8," respectively, all of which are just neighbors to the correct node in the map produced by the classifier obtained in the simulation. [See Fig. 11(c).]

This is a strong evidence that the classifier made by the proposed neural network preserves the topological ordering of the input patterns, the handwritten numerals. In order to improve the performance in this point, we are attempting to incorporate the concept of k-nearest neighbor rule into the decision of the final class.

VII. Concluding Remarks

In this paper, we have presented three sophisticated neural-network classifiers to recognize the totally unconstrained handwritten numerals: multiple MLP classifier, HMM/MLP hybrid classifier, and structure-adaptive SOM classifier. All of them have produced better results than several previous methods reported in the literature on the same database. Actually, the proposed methods have a small, but statistically significant $(p > 0.995)$, advantage in recognition rates obtained by the conventional methods. We have found that the proposed neural-network classifiers might solve the complex classification problem.

Although the multiple MLP classifier was the best in this simulation, each classifier has its own merits and gives some possibility to enlarge the conventional neural-network classifiers for real-world problems. The multiple MLP classifier leads to a reliable recognizer without great effort to fine-tune the individual MLP classifiers. Also, the HMM/MLP hybrid classifier complements each method for improving the overall performance, and the structure-adaptive SOM classifier automatically finds a network structure and size suitable for the classification of complex patterns through the ability of structure adaptation.

Even though our work to date is concentrated on handwritten numeral recognition, we believe that the methods presented can be easily generalized to more difficult problems, such as handwritten Roman character recognition and Hangul (Korean script) recognition. The further works are under going with the more difficult task of recognizing handwritten Hangul.

Acknowledgment

The author would like to thank J. H. Baik, K. Lee, and S.-I. Lee, graduate students in the AI laboratories at Yonsei University, for their support of the implementation of the algorithms and the simulation performed for this research.

References

[1] C. Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam, "Computer recognition of unconstrained handwritten numerals," *Proc. IEEE*, vol. 80, pp. 1162–1180, 1992.

[2] P. Ahmed and C. Y. Suen, "Computer recognition of totally unconstrained handwritten ZIP codes," *Int. J. Pattern Recognition Artificial Intell.*, vol. 1, nol. 1, pp. 1–15, 1987.

[3] M. Beun, "A flexible method for automatic reading of handwritten numerals," *Philips Tech. Rev.*, vol. 33, pp. 89–101, 130–137, 1973.

[4] E. Cohen, J. J. Hull, and S. N. Srihari, "Understanding handwritten text in a structured environment: Determining ZIP codes from addresses," *Int. J. Pattern Recognition Artificial Intell.*, vol. 5, nos. 1 and 2, pp. 221–264, 1991.

[5] B. Duerr, W. Haettich, H. Tropf, and G. Winkler, "A combination of statistical and syntactical pattern recognition applied to classification of unconstrained handwritten numerals," *Pattern Recognition*, vol. 12, pp. 189–199, 1980.

[6] P. D. Gader, D. Hepp, B. Forester, T. Peurach, and B. T. Mitchell, "Pipelined systems for recognition of handwritten digit in USPS ZIP codes," in *Proc. U.S. Postal Service Advanced Technol. Conf.*, pp. 539–548, 1990.

[7] Y. J. Kim and S. W. Lee, "Off-line recognition of unconstrained handwritten digits using multilayer backpropagation neural network combined with genetic algorithm," (in Korean), in *Proc. 6th Wkshp. Image Processing Understanding*, 1994, pp. 186–193.

[8] S. Knerr, L. Personnaz, and G. Dreyfus, "Handwritten digit recognition by neural networks with single-layer training," *IEEE Trans. Neural Networks*, vol. 3, pp. 962–968, 1992.

[9] A. Krzyzak, W. Dai, and C. Y. Suen, "Unconstrained handwritten character classification using modified backpropagation model," in *Proc. 1st Int. Wkshp. Frontiers Handwriting Recognition*, Montreal, Canada, 1990, pp. 155–166.

[10] C. L. Kuan and S. N. Srihari, "A stroke-based approach to handwritten numeral recognition," in *Proc. U.S. Postal Service Advanced Technol. Conf.*, 1988, pp. 1033–1041.

[11] L. Lam and C. Y. Suen, "Structural classification and relaxation matching of totally unconstrained handwritten ZIP-code numbers," *Pattern Recognition*, vol. 21, no. 1, p. 19–31, 1988.

[12] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, and H. S. Baird, "Constrained neural network for unconstrained handwritten digit recognition," in *Proc. 1st Int. Wkshp. Frontiers Handwriting Recognition*, Montreal, Canada, 1990, pp. 145–154.

[13] R. Legault and C. Y. Suen, "Contour tracing and parametric approximations for digitized patterns," in *Computer Vision and Shape Recognition*, A. Krzyzak, T. Kasvand, and C. Y. Suen Eds. Singapore: World Scientific, 1989, pp. 225–240.

[14] B. Lemarie, "Practical implementation of a radial basis function network for handwritten digit recognition," in *Proc. 2nd Int. Conf. Document Anal. Recognition*, Tsukuba, Japan, 1993, pp. 412–415.

[15] T. Mai and C. Y. Suen, "A generalized knowledge-based system for the recognition of unconstrained handwritten numerals," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 835–848, 1990.

[16] B. T. Mitchell and A. M. Gillies, "A model-based computer vision system for recognizing handwritten ZIP codes," *Machine Vision Applicat.*, vol. 2, pp. 231–243, 1989.

[17] C. Nadal and C. Y. Suen, "Recognition of totally unconstrained handwritten digit by decomposition and vectorization," Concordia Univ., Montreal, Canada, Tech. Rep., 1988.

[18] L. Stringa, "A new set of constraint-free character recognition grammars," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 1210–1217, 1990.

[19] C. Y. Suen, C. Nadal, T. Mai, R. Legault, and L. Lam, "Recognition of handwritten numerals based on the concept of multiple experts," in *Proc. 1st Int. Wkshp. Frontiers Handwriting Recognition*, Montreal, Canada, 1990, pp. 131–144.

[20] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1978.

[21] L. K. Hansen and P. Salamon, "Neural-network ensembles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 993–1001, 1990.

[22] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computa.*, vol. 3, pp. 79–87, 1991.

[23] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[24] S.-B. Cho and J. H. Kim, "Strategic application of feedforward neural networks to large-scale classification," *Complex Syst.*, vol. 6, no. 4, pp. 363–389, 1992.

[25] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," *Neural Networks for Speech and Image Processing*, R. J. Mammone, Ed. London: Chapman-Hall, 1993.

[26] M. D. Richard and R. P. Lippmann, "Neural network classifiers estimate Bayesian *a posteriori* probabilities," *Neural Computa.*, vol. 3, pp. 461–483, 1991.

[27] D. J. C. MacKay, "A practical Bayesian framework for backprop networks," *Neural Computa.*, vol. 4, no. 3, pp. 448–472, 1992.

[28] S.-B. Cho and J. H. Kim, "A multiple network architecture combined by fuzzy integral," in *Proc. IEEE/INNS Int. Joint Conf. Neural Networks*, vol. II, Nagoya, Japan, 1993, pp. 1373–1376.

[29] ——, "Multiple network fusion using fuzzy logic," *IEEE Trans. Neural Networks*, vol. 6, pp. 497–501, 1995.

[30] T. D. Sanger, "A tree-structured adaptive network for function approximation in high-dimensional spaces," *IEEE Trans. Neural Networks*, vol. 2, pp. 285–293, Mar. 1991.

[31] B. Friztke, "Growing cell structures—a self-organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, no. 9, pp. 1441–1460, 1994.

[32] T. Li, Y. Y. Tang, and L. Y. Fang, "A structure-parameter-adaptive (SPA) neural tree for the recognition of large character set," *Pattern Recognition*, vol. 28, no. 3, pp. 315–329, 1995.

[33] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, pp. 1464–1480, 1990.

Sung-Bae Cho received the B.S. degree in computer science from Yonsei University, Seoul, Korea, in 1988 and the M.S. and Ph.D. degrees in computer science from KAIST (Korea Advanced Institute of Science and Technology), Taejeon, Korea, in 1990 and 1993, respectively.

He worked as a Member of the Research Staff at the Center for Artificial Intelligence Research at KAIST from 1991 to 1993. He was an Invited Researcher of Human Information Processing Research Laboratories at ATR (Advanced Telecommunications Research) Institute, Kyoto, Japan from 1993 to 1995. Since 1995, he has been an Assistant Professor in the Department of Computer Science, Yonsei University. His research interests include neural networks, pattern recognition, intelligent man-machine interfaces, evolutionary computation, and artificial life.

Dr. Cho was awarded outstanding paper prizes from the IEEE Korea Section in 1989 and 1992, and another one from the Korea Information Science Society in 1990. He was also the recipient of the Richard E. Merwin prize from the IEEE Computer Society in 1993. He was listed in *Who's Who in Pattern Recognition* from the International Association for Pattern Recognition in 1994, and nominated for biographical inclusion in the Fifth Edition of *Five Thousand Personalities of the World* from the American Biographical Institute in 1995. He is a Member of the Korea Information Science Society, INNS, the IEEE Computer Society, and the IEEE Systems, Man, and Cybernetics Society.