A Self-Organizing Neural Tree for Large-Set Pattern Classification

Hee-Heon Song, *Member, IEEE,* and Seong-Whan Lee, *Senior Member, IEEE*

***Abstract*—Neural networks have been successfully applied to various pattern classification problems in terms of their learning ability, high discrimination power, and excellent generalization ability. However, for the case of classifying large-set and complex patterns, the greater part of conventional neural networks suffer from several difficulties such as the determination of the structure and size of the network, the computational complexity, and so on. In this paper, to cope with these difficulties, we propose a structurally adaptive intelligent neural tree (SAINT). The basic idea is to partition hierarchically input pattern space using a tree-structured network which is composed of subnetworks with topology-preserving mapping ability. The main advantage of SAINT is that it attempts to find automatically a network structure and size suitable for the classification of large-set and complex patterns through structure adaptation. Experimental results reveal that SAINT is very effective for the classification of large-set real world handwritten characters with high variations, as well as multilingual, multifont, and multisize large-set characters.**

***Index Terms*—Large-set pattern classification, parameter adaptation, structure adaptation, structurally adaptive intelligent neural tree, topology-preserving mapping.**

I. Introduction

CLASSIFICATION is to assign a physical object or event into one of several prespecified classes based on the extraction of significant features and the processing or analysis of these features [1]. Over the last decades, extensive study on classification problems has led to various mathematical models that provide the theoretical basis for classifier design. It has many applications in various areas such as robot control, remote sensing, speech recognition, character recognition, and so on. The methodology frequently used for pattern classification is statistical approach, in which decision theory derived from statistics of input patterns is used for designing a classifier [2], [3]. Though this approach has been successfully applied to solve various problems in pattern classification [4], [5], it has a difficulty in expressing structural information unless a suitable choice of features is possible. Furthermore, this approach requires much heuristic information to design a classifier.

Manuscript received May 22, 1996; revised January 2, 1998. This work was supported by the Directed Basic Research Fund of Korea Science and Engineering Foundation under Grant 95-0100-06-01-3 and Creative Research Initiatives of the Korean Ministry of Science and Technology.

H.-H. Song is with the Department of Computer Engineering Education, Andong National University, songchon-dong, Andong, Kyongbuk 760–749, Korea.

S.-W. Lee is with the Center for Artificial Vision Research, Department of Computer Science and Engineering, Korea University, 1 5-ka, Anam-dong, Seongbuk-ku, Seoul 136-701, Korea.

Publisher Item Identifier S 1045-9227(98)02774-X.

As a new means for implementing various classifiers based on statistical and structural approaches, neural networks have emerged. A neural network is a collection of interconnecting units with an ability to store information in a distributed manner and in the form of interconnecting weights between the units. In this model which is often called "connectionist" architecture, processing occurs in a highly interconnected network of simple processing units. Each unit determines its activation value as a function of the signals it receives from other units. When input patterns are presented to the network, every unit in the network is involved in simple calculation using the weights and the input values to produce an output. This process resembles the interactions between biological networks of neurons but some networks do not precisely correspond to the way biologocal networks of neurons work. However, neural networks appear to capture some important properties of how actual neural networks operate.

The field of neural networks is highly interdisciplinary, and has been approached by neurobiologists, physicists, electrical engineers, and computer scientists with each of their respective domains having viewpoints on topics throughout the field. Several models have been proposed for classification problem [6], such as restricted Coulomb energy [7], neocognitron [8], multilayer perceptrons (MLP's) using error backpropagation [9], adaptive resonance theory [10], self-organizing feature map (SOFM) [11] and a number of its variants. These neural networks have been successfully applied to various problems in pattern classification owing to their learning ability, high discrimination power, and excellent generalization ability. However, for the case of classifying patterns which are large-set and require complex decision boundaries in high-dimensional pattern space, the greater part of conventional neural networks suffer from some of difficult problems to solve, such as the structure and size of the network, the computational complexity, and so on.

For example, even though it has been reported that MLP using error backpropagation is adequate for tough pattern classification problems owing to its high discrimination power and excellent generalization ability [12]–[14], the number of classes to be allowable is too small to apply it directly to large-set classification problems without preclassification. In other words, as the number of classes increases, the computational complexity of the learning process quickly reaches unmanageable proportions. Furthermore, it is very difficult to determine the structure and size of the network for the classification of large-set and complex patterns.

In recent years, in order to overcome the difficulties of determining the structure and size of the network, several ap-

proaches on the basis of the structure adaptation of the network havebeen proposed, which may be divided into two categories; the structure adaptation in single-stage network [15]–[17] and the structure adaptation in multistage network [18]–[20]. However, the greater part of them also are inadequate for classifying patterns which are large-set and require complex decision boundaries in high-dimensional pattern space.

In this paper, to cope with these difficulties, we propose a structurally adaptive intelligent neural tree (SAINT). The basic idea is to partition hierarchically input pattern space using a tree-structured neural network which is composed of subnetworks with topology-preserving mapping ability. SAINT is a multilevel and competitive neural network suitable for the classification of large-set patterns. Moreover, since a fixed topology of network cannot properly model a given inputdistribution due to the difficulties of predicting the statistics of complex input pattern space, SAINT attempts to find automatically a network structure and size suitable for the classification of large-set and complex patterns through structurally adapting ability. To evaluate the performance of SAINT, we have performed experiments with large-set real world handwritten characters including high variations, as well as multilingual, multifont, and multisize large-set characters.

This paper is organized as follows. In the next section, we briefly review some of structurally adapting techniques proposed in the literature and discuss the considerations in adapting the network structure. Section III presents the problem formulation and the motivation for hierarchically partitioning the input pattern space. In Section IV, we propose SAINT and its learning algorithm for solving not only the complex input environment but also large-set classification problem. In Section V, we give the results and analysis of our experiments performed to evaluate SAINT. Next, the relationship to MLP and several methods which are closely related to SAINT will be discussed in Section VI. Finally, we summarize this work with a brief discussion of the scope of future work.

II. Structure Adaptation in Neural Networks

In general, the mapping capability of neural network is dependent on their structure, that is, the number of neurons [6]. A small network might not be adequate for a complex problem, and a large network might lead to poor generalization power. Furthermore, a fixed topology of network cannot properly model a given input distribution due to the difficulties of predicting the statistics of complex input pattern space. Currently, there is no formal way for computing the network structure as a function of the complexity of the problem. It is usually selected by trial-and-error that can be rather time consuming.

Thus, in case that the structure of the network is fixed in the beginning of the training process, we must think of the following two questions, what is called the frame problem of neural networks [15]. One is how to decide the initial structure of the network, and the other is how to modify the structure of the network to reflect the complex input environment. To tackle the frame problem of neural networks, it is necessary to adapt the structure of the network by modifying both the number of neurons and the structural relationship between neurons in the network.

Several methods have been proposed to solve the frame problem of neural networks. These methods may be divided into structure adaptation in single-stage network and structure adaptation in multistage network. In this section, we review some of the structurally adapting techniques proposed in the literatures, and discuss the considerations in adapting the network structure.

A. Structure Adaptation in the Single-Stage Networks

Restricted Columb energy (RCE) [7] is an incremental algorithm in which each unit has a number of prototypes and each one has its domination region. A new prototype is created if an input pattern does not fall into the domination region of any of the existing ones.

Lee *et al.* [15] proposed a self-development neural network, called space partition network (SPAN), which is able to adapt its structure by adding neurons, deleting neurons, and modifying the structural relationships between neurons in the network. SPAN has two different levels of adaptation abilities. One is parameter level adaptation which takes the structure of the network as fixed and adapts the synaptic weights of the neurons in the network. The other is structure level adaptation which adapts the structure of the network by changing the number of neurons and the structural relationship between neurons in the network based on homogeneous neighborhood systems.

Blackmore *et al.* [16] proposed an incremental feature map algorithm. In this network, mapping of high-dimensional structure onto two-dimensional feature map is carried out based on Kohonen's network and learning algorithm [11]. Moreover, nodes and connections are added to or deleted from the map according to the input distribution. New nodes are inserted based on distortion (or quantization) error of the node and some nodes are deleted according to the referencing frequency.

Fritzke [17] proposed a method to construct two-dimensional cell structures which are special feature map with problem-dependent cell structure. All nodes in the network form n-dimensional topological neighborhoods. Kohonen's learning algorithm [11] was used for the adaptation of network parameters. In structure adaptation, the insertion of a node based on distortion error of the network is occurred, and the deletion of the node according to the referencing frequency of a node is occurred.

B. Structure Adaptation in the Multistage Networks

A single-stage classifier makes class assignment based on a single decision using a fixed set of the features. In this case, discriminant function is determined for each class and a pattern is assigned to the class with the maximum discriminant function value. In a single-stage classifier, a common set of features for all classes is used jointly in a single one-shot decision step. Consequently, it is difficult to evaluate the contribution of a given feature to the classification decision. And a large number of features should be required in order

to achieve a satisfactory separation of large-set patterns. An alternative classification structure is a multistage classifier where the most obvious or natural discriminations are done first and the more subtle distinctions are postponed until a later stage [3]. The rationale for hierarchical classifiers is that by using different subsets of n features at the various decision levels a better performance may be achieved than by employing a single best set of n features in a one step decision [23].

Sanger [18] proposed a tree-structured adaptive network for approximating function on the basis of the hope that for most of the data only a few dimensions of the input pattern may be necessary to compute the desired output function. In his network, a function is decomposed first along one dimension of the function to form one level of the tree, and then decomposed along other dimensions to form other levels of the tree. The structure of the network is dynamically changed during learning and determined by the data distribution and output function to be approximated. The network structure has n-ary tree topology and its parameters are updated using Widrow–Hoff's learning rule. And in structure adaptation, new neurons are inserted minimizing the mean-squared error. But, unlike other approaches, there is no successive approximation property, that is, clustering property.

Li *et al.* [20] proposed a structure-parameter-adaptive (SPA) neural tree which is multilevel competitive neural network with tree topology. In this network, Hebbian's learning algorithm was used for the adaptation of the network parameters, and used three operations for the structure adaptation: the creation of new nodes, the splitting of nodes into more nodes, and the deletion of nodes from the network. If the distortion error of a leaf node exceeds the predefined values, a sibling node is created. When the accumulated distortion error of a leaf node exceeds thresholds over a period of time, this node is split into a few child nodes. And when a leaf node has not been used frequently over a period of time, the leaf node is deleted.

C. Considerations in Adapting the Network Structure

In this section, we discuss some of considerations in adapting the network structure. In order to design a classifier suitable for the classification of large-set and complex patterns, the following properties should be considered:

1) a hierarchical structure suitable for the classification of large-set patterns;
2) successive approximation property to behave as a classifier;
3) automatic adaptation of the network parameters and structure to overcome the complex input environment.

However, the conventional approaches reviewed in the previous sections may be inadequate for classifying patterns which are large-set and require complex decision boundaries in several aspects. First of all, the greater part of single-stage networks based on Kohonen's network have a great difficulty in solving the classification problem of large-set patterns due to their inherent limitations in structural aspect of the network. Furthermore, as the network size becomes larger, the computational complexity of the learning process quickly reaches unmanageable proportions. Next, the unbalance of computational load is occurred in most multistage networks. In other words, the resultant network tends to be taken the form of the skewed-tree structure. It may be attributed to the following two facts: 1) weight adjustment of only a single winner node at each level from the root to the leaf and 2) undesirable adaptation of network structure.

Another aspect to be considered is overfitting problem. In the work of Li *et al.* [20], overfitting may be occurred when the vigilance value to control the creation of its sibling node and the threshold value to control the creation of its child nodes are inharmonious, the decreasing rule of these values is inadequate, or total number of iterations is excessive. As a result, the total size of the network becomes larger and overfitting problem at nodes around leaves may be occurred. Unfortunately, it is very difficult and heuristic to establish these values appropriately.

Finally, a part of these methods are data-dependent approaches. They have the difficulty of initializing the weight of the new node and require much heuristic information. Moreover, in view of convergence property, the difficulty of determining the stopping condition remains heuristic and results in underfitting problem.

III. Problem Formulation

In this section, we define the problem that SAINT is supposed to solve. Under the assumption that $\mathbf{x}^i = (x_1, x_2, \cdots, x_n)' \in R^n$ is a n-dimensional pattern whose component $\{x_k, 1 \le k \le n\}$ are real-valued and continuous random variables, consider a set of input patterns, $\mathbf{x} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots\}$. In the case that *a priori* probabilities $P(\omega_j)$ and the probability density functions $p(\mathbf{x} \mid \omega_j)$ are known, the problem of pattern classification can be solved using Bayes decision rule

$$P(\omega_j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_j) P(\omega_j)}{p(\mathbf{x})} \tag{1}$$

where

$$p(\mathbf{x}) = \sum_{j=1}^{M} p(\mathbf{x} \mid \omega_j) P(\omega_j). \tag{2}$$

However, in the absence of such knowledge, the problem may be directly estimated from the given input patterns. A neural network can be considered as a mapping machine between input and output sets. Mathematically speaking, a neural network represents a function F that maps input I into output O

$$F : I \mapsto O \quad \text{or} \quad \mathbf{y} = F(\mathbf{x}) \tag{3}$$

where $\mathbf{y} \in O$ and $\mathbf{x} \in I$. In our problem, we define that a classification is a mapping from a feature space to some set of output classes. Thus, to represent a given feature space accurately, its topological properties should be captured. Let $V = R^n$ denote the feature space from which feature vectors corresponding to input patterns stem. We are interested in the mapping of V onto the network with tree topology which

is composed of subnetworks with two-dimensional lattice topology. This mapping should have the following properties.

1) Similar input patterns are mapped onto topologically close nodes in the corresponding subnetwork of overall tree-structured network.
2) Regions where the probability density of the input distribution is high should be represented by a number of nodes in lower subnetworks connected to their child nodes.
3) Overlap between nodes in a subnetwork as well as subnetworks should be allowed for reflecting the complex input environment.

We expect that if it is possible to preserve the similarity relation among feature vectors in spite of reducing dimensionality (that is, $n > k$), then the complexity of the feature vectors is reduced without the loss of information. The mapping of the input feature space $V \in R^n$ onto the two-dimensional lattice L with respect to the synaptic weights of a node can be defined as follows:

$$F_{\mathbf{w}} : V \mapsto L, \quad \mathbf{x} \in V \mapsto F_{\mathbf{w}}(\mathbf{x}) \in L \tag{4}$$

where $\mathbf{w}$ is the weight vector of a pattern on n-dimensional pattern space, and $F_{\mathbf{w}}(\mathbf{x})$ is defined through the following condition:

$$\|\mathbf{w}_{F_{\mathbf{w}}(\mathbf{x})} - \mathbf{x}\| = \min_{i \in L} \|\mathbf{w}_i - \mathbf{x}\|. \tag{5}$$

We want to distribute hierarchically all the two-dimensional lattice of subnetwork in the tree topology of the overall network in R^n so that the resulting mapping is topology-preserving and distribution-preserving with respect to the n-dimensional probability density $p(\mathbf{x})$ of the feature space. Fortunately, SOFM with a topology-mapping ability was proposed by Kohonen [11]. In this network, the weights of nodes are organized so that neighboring nodes on the network tend to have similar weight vectors which represent the neighboring regions in the input pattern space.

One of the simplest systems is a linear array of functional units, each of which receives the same set of input patterns in parallel. Assume that such a unit i produces a different response to the different input patterns: $o_i(x_1), o_i(x_2), \cdots, o_i(x_k)$. And assume for simplicity that a set of input patterns $\{x_i : i = 1, 2, \cdots, k\}$ can be ordered in some topologic way such that $x_1 R x_2 R x_3 \cdots R x_k$, where R stands for a simple ordering relation. Obviously, topology-preserving mapping is to generate feature map preserving the relationship of input patterns, which may be defined as follows.

Definition 1: The system is said to produce a one-dimensional topology-preserving mapping if for $i_1 > i_2 > i_3 > \cdots > i_k$

$$\begin{aligned} o_{i_1}(x_1) &= \max_i \{o_i(x_1) : i = 1, 2, \cdots, n\}, \\ o_{i_2}(x_2) &= \max_i \{o_i(x_2) : i = 1, 2, \cdots, n\} \\ o_{i_3}(x_3) &= \max_i \{o_i(x_3) : i = 1, 2, \cdots, n\} \\ &\vdots \\ o_{i_k}(x_k) &= \max_i \{o_i(x_k) : i = 1, 2, \cdots, n\}. \end{aligned} \tag{6}$$

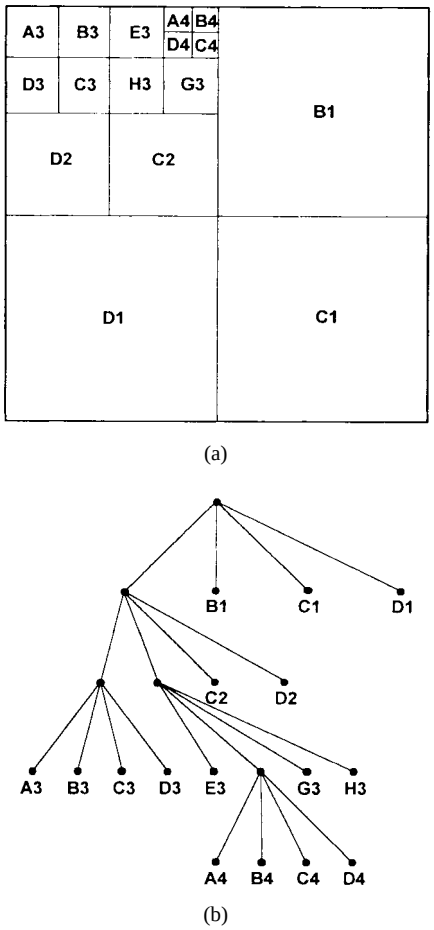


Fig. 1. Geometrical interpretation of hierarchical partitioning. (a) Hierarchically decomposed feature map. (b) Quadtree representation of (a).

Fig. 1 illustrates the geometrical interpretation of hierarchical partitioning. For simplicity, we consider the subnetwork composed of four nodes. Fig. 1(a) represents feature map hierarchically decomposed into two-dimensional lattice topology. It can be hierarchically represented by the well-known quadtree structure as shown in Fig. 1(b).

IV. SAINT: Structurally Adaptive Intelligent Neural Tree

A. Architecture of SAINT

SAINT is a tree-structured neural network with two-dimensional lattice-structured subnetworks. To preserve the lattice topology of each subnetwork, we considered the homogeneous neighborhood system which can be defined as follows.

Definition 2: If and only if the neighborhood of node i, η_i is such that:

1) $i \in \eta_i$;
2) if $j \in \eta_i$, then $i \in \eta_j$ for any $i \in \mathcal{T}_s$;

then homogeneous neighborhood system η is defined as follows:

$$\eta = \{\eta_i \mid i \in \mathcal{T}_s\}, \quad \eta_i = \{j \mid \|l_j - l_i\| \le r, j \in \mathcal{T}_s\} \tag{7}$$

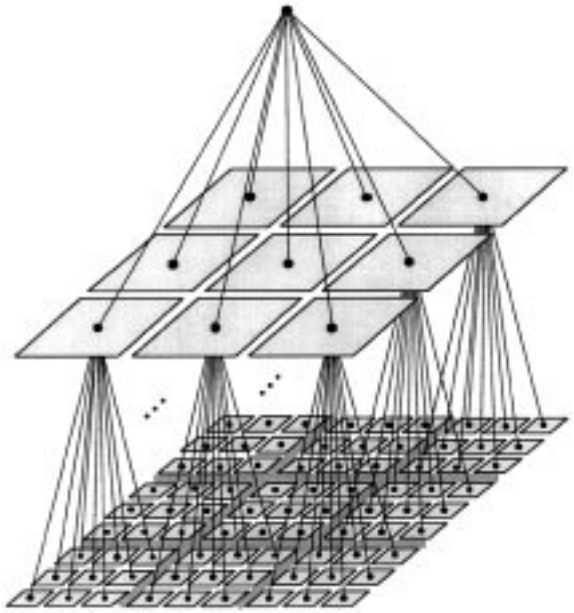


Fig. 2. Architecture of SAINT.

where $\mathcal{T}_s$ is the set of nodes of a subtopology, l_i and l_j are the lattice position for node i and j, respectively, and r is the order of neighborhood system.

Based on these concept, the architecture of SAINT is illustrated in Fig. 2.

In this figure, a node i has its neighbors η_i which are composed of a group of nodes around i in the lattice topology of the subnetwork. The size of the neighborhoods decreases with time. Every node i in this figure has n-dimensional synaptic weight vector $\mathbf{w}_i$. This weight vector may be regarded as the position of i in the input pattern space. Each node i computes the squared error between the input pattern $\mathbf{x}$ and its synaptic weight $\mathbf{w}_i$. The squared error of node i is calculated as follows:

$$\mathrm{err}^i = \sum_{j=0}^{n-1} \left|\left|\mathbf{x}_j - \mathbf{w}_j^i\right|\right|^2. \tag{8}$$

To adjust the weight vectors of nodes in the network, we adopted Kohonen's learning algorithm [11] which is designed to create a vector quantizer. It can be summarized as follows.

1) Initialize all weight vectors.
2) Select the node with minimum squared error (MSE) with respect to the input pattern $\mathbf{x}$

$$||\mathbf{x}(t) - \mathbf{w}_c(t)|| = \min_i\{||\mathbf{x}(t) - \mathbf{w}_i(t)||\}. \tag{9}$$

3) Update weight vectors of the nodes that lie within the neighborhoods of the node

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \alpha(\mathbf{x}(t) - \mathbf{w}_c(t)) \quad \text{for all } c \in \eta_c. \tag{10}$$

4) Decrease the order of neighborhood system in (7) as follows:

$$r = r_0 \cdot \left(1 - \frac{\text{No-Of-Iteration}}{\text{Total-No-Of-Iteration}}\right). \tag{11}$$

5) Repeat 1) through 4) for all training vectors until stopping criterion is satisfied.

After enough input training patterns have been presented, the weight vectors of nodes will specify clusters or vector centers. Thus the point density function of the vector centers tends to approximate the probability density function of the input patterns. In addition, the weight vectors will be organized so that topologically close nodes are sensitive to input patterns that are physically similar in Euclidean space. Fig. 3 shows the adaptation process of the network parameters using this learning algorithm. In this figure, the winner node A for the input pattern $\mathbf{x}$ and its three neighboring nodes are adapted.

B. Adaptive Characteristics of SAINT

The network dynamics are characterized by the structure adaptation of all nodes in the network. Through structure adaptation, SAINT can dynamically modify the frame that nodes reside in, so that the structure of the network can always reflects the statistics of the input pattern space.

We used the average distortion error as the measure for adapting the network structure. Before the description of the structure adaptation, we first define this measure. To define the distortion error more formally, the following notations are introduced.

D	Average distortion.
$\mathbf{m}$	Reference vector.
$d(\mathbf{x}, \mathbf{m})$	Distortion measure (Euclidean distance).
$P(\mathbf{m}_i)$	Probability of the reference vector $\mathbf{m}_i$.
$f(\mathbf{x} \mid \mathbf{m}_i)$	Multidimensional probability density function of $\mathbf{x}$ given $\mathbf{m}_i$.

Based on these notations, the average distortion error can be defined as follows:

$$\begin{aligned} D &= E[d(\mathbf{x}, \mathbf{m})] \\ &= \sum_{i=1}^{L} E[d(\mathbf{x}, \mathbf{m}_i) \mid \mathbf{x} \in V_i] P(\mathbf{m}_i) \\ &= \sum_{i=1}^{L} P(\mathbf{m}_i) \int_{\mathbf{x} \in V_i} d(\mathbf{x}, \mathbf{m}_i) f(\mathbf{x} \mid \mathbf{m}_i)\, d\mathbf{x} \\ &= \sum_{i=1}^{L} P(\mathbf{m}_i) d_i(\mathbf{x}, \mathbf{m}_i). \end{aligned} \tag{12}$$

There is a tradeoff between distortion error and representation complexity in SAINT. Since increasing the number of neurons in SAINT will decrease the distortion error of the network, we can use this distortion error as a measure to determine the structure adaptation of the network. If a node i contributes too much to the average distortion of the network, that is, its Voronoi region V in the input pattern space is underrepresented by neuron i, then a new node should be generated to share some of the representation load of node i. As a result, the average distortion can be decreased.

Based on the average distortion, we considered three operations for the structure adaptation: node generation, deletion, and merging. After level-wise parameter adaptation for the current level of the network to represent accurately the distribution of the input pattern space, nodes to contribute too

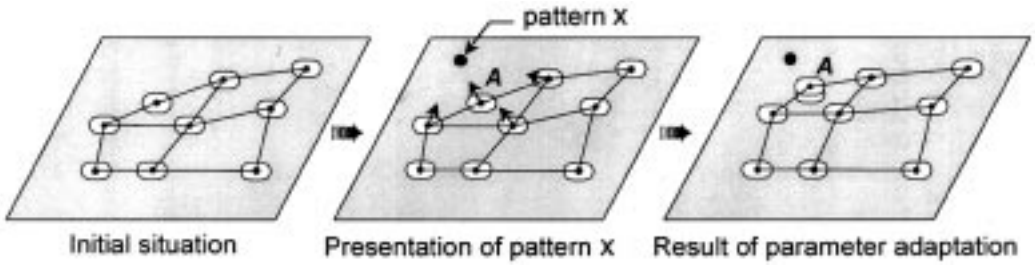


Fig. 3. A typical example of parameter adaptation.

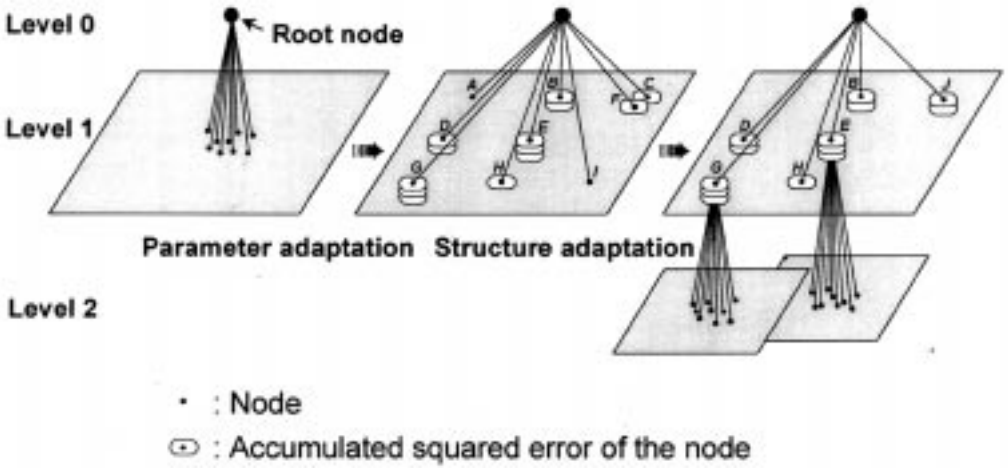


Fig. 4. A typical example of structure adaptation.

much to the average distortion of the network are selected and the subnetworks to share some of representation load of the nodes are generated at the next level of the network. In this case, the newly generated subnetworks have the same lattice-topology as that of the upper level in the network. Moreover, the overall structure of the network preserves the tree topology. As a result, this adaptation has an effect of fine coding.

And a node which is not active for a long period of time is deleted from the subnetwork at the current level in the network. To determine whether a node is active or not, we define an activity measure for nodes which is an operational definition of the average output activity of the node. As a result, this adaptation has an effect of preventing the overgrowing problem of the network.

We also considered the merging of two nodes similar to one another. Through this adaptation, we may get the effect of preventing the network from overtraining for training data.

Fig. 4 illustrates a typical example of structure adaptation which includes all the process of generating new nodes, deleting nodes, and merging two nodes.

In this figure, the height of the node indicates the representation load of thenode. As shown in this figure, after parameter adaptation, node A and I are deleted from the current network and node C an F are merged to J. Also, node E and G generate their subnetworks as their child nodes at next level of tree-structured network. Each subnetwork has the predefined number of nodes and in order to preserve the topology of each subnetwork, all nodes in the subnetwork are adapted through the parameter adaptation process in the next iteration step.

C. Learning Algorithm

We recall that our goal is to design a neural network suitable for solving the two difficult problems of pattern classification, that is, the complex input environment and a large-set classification problem. To realize these objective, the learning algorithm for SAINT proposed in the previous sections should have the following properties; 1) level-wise parameter adaptation should be completely carried out before the structure adaptation of the network, and 2) the network should have the successive approximation property according to the tree topology of overall network. Based on these properties, learning algorithm for SAINT is as follows:

1) Initialize variables.
   - $k \leftarrow 1$; // k is a level number of tree-structured network.
   - $\theta_k \leftarrow \theta_0$; // threshold for creating or deleting nodes.
2) WHILE (stopping criterion is satisfied?) DO
   - $r \leftarrow r_0$; $\alpha \leftarrow \alpha_0$; // r: the order of neighborhood system, α: the learning rate.
   - WHILE (epoch?) DO.
     - –Calculate the distortion error between input and nodes at last level, and select the winner node n_c with the minimum error

$$c = \arg\min_i \{||\mathbf{x}(t) - \mathbf{w}_i(t)||\}.$$

–Update the weight of the winner node and its neighboring nodes

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \alpha(\mathbf{x}(t) - \mathbf{w}_c(t)) \quad \text{for all } c \in \eta_c.$$

–Accumulate the distortion error of the winner node

$$d_{n_c}^{\text{acc}}(t+1) = d_{n_c}(t) + \frac{d_{n_c}^{\text{acc}}(t_L) \cdot (w - t + t_L) \cdot \delta(w - t + t_L)}{w}$$

where $d_{n_c}^{\text{acc}}$ is defined over a time window w of finite size [20], $d_{n_c}(t)$ is a distortion error at time t, t_L denotes the last time when node c was selected as winner, and $\delta(\cdot)$ is a step function.

–Increase the reference counter of the winner node.

–Decrease the size of neighborhood r and the learning rate α

$$r = r_0 \cdot \left(1 - \frac{\text{Number-Of-Iteration}}{\text{Total-Number-Of-Iteration}}\right)$$
$$\alpha = \alpha_0 \cdot \left(1 - \frac{\text{Number-Of-Iteration}}{\text{Total-Number-Of-Iteration}}\right).$$

END-WHILE.

- Delete nodes which are not active for a long period of time.
- Merge nodes in each subnetwork at level k.

—Select nodes N_i and N_j satisfying the following condition:

$$||\mathbf{w}_i - \mathbf{w}_j||^2 \le \frac{\epsilon}{M} \sum_{j=1, j\neq i}^{M} ||\mathbf{w}_i - \mathbf{w}_j||^2 \quad \text{for } i = 1, 2, \cdots, M$$

where ϵ is $0 < \epsilon \ll 1$ and M is the number of nodes in a subnetwork.

—Calculate the initial weight of node to be merged

$$\mathbf{w}_{\text{new}} = 0.5(\mathbf{w}_i + \mathbf{w}_j).$$

—Delete node N_i and N_j.

- Create a subnetwork at the node with larger accumulated distortion error than θ_k.
- Decrease θ

$$\theta_{k+1} = \theta_0 \cdot \beta^k$$

where $0 < \beta < 1$.
END-WHILE.

3) Optimize networks.
4) Assign the corresponding class into the leaf nodes.

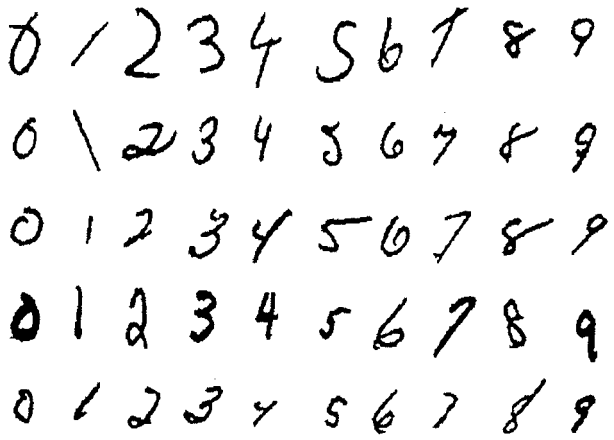


Fig. 5. Representative samples from handwritten numeral database of Concordia University.

Since the upper level nodes represent a large number of patterns by virtue of the property of tree-structured network, they have larger representation load than the lower level nodes. Thus, the upper level nodes need larger threshold values and these values should be reduced in each level further down.

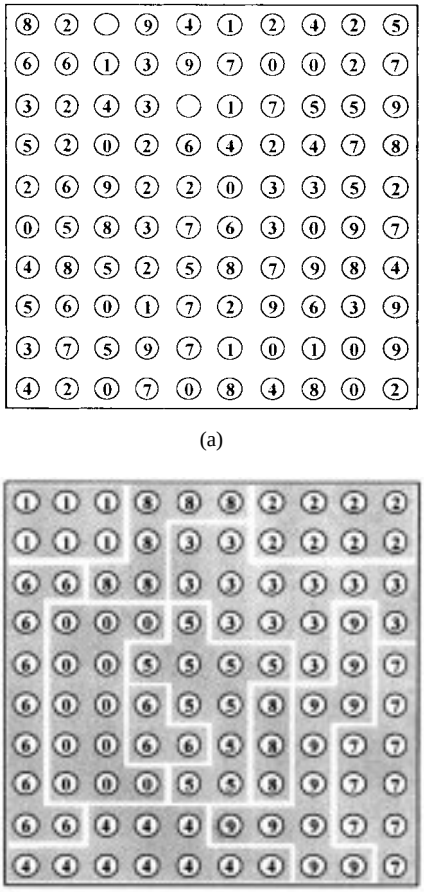
After parameter adaptation, the merging process of nodes as well as the deletion of node was performed based on the similarity between neighboring nodes in a subnetwork. In the merging process, if the synaptic weights between neighboring nodes are too close, one of two nodes is eliminated and the remaining one has the synaptic weight set to be the average of two nodes. Through this merging process, we are able to keep the overall network size small and have the effect that minimize the overfitting problem of the network. After then, if necessary, a new subnetwork is generated at the next level of the current leaf nodes. These adaptation processes are continued until the stopping criterion is satisfied.

V. Experimental Results and Analysis

A. Experiments on the Ability of Topology-Preserving Mapping

To illustrate the topology-preserving mapping ability of a subnetwork in SAINT, experiment with unconstrained handwritten numeral database of Concordia University of Canada was performed. This database consists of 6000 unconstrained numerals originally collected from dead letter envelopes by the U.S. Postal Services at different locations in the U.S. The numerals of this database were digitized in bilevel on a 64 $\times$ 224 grid of 0.153-mm-square elements, giving a resolution of approximately 166 PPI [24]. Fig. 5 shows some representative samples taken from this numeral database used in this experiment.

The subnetwork used in this experiment consists of 10 $\times$ 10 lattice, and 64 dimensions of directional features extracted from each numeral using Kirsch masks [13] were used as an input to the sunbetwork. Fig. 6 shows the result of topology-preserving mapping in the subnetwork. In this figure, a circle represents one node in the subnetwork and the number labeled in a circle indicates one of the ten class information from 0 to 9. This label is assigned by the result of the majority voting



(a) (b)

Fig. 6. Result of topology-preserving mapping. (a) Initial state of the subnetwork. (b) Final state of the subnetwork after parameter adaptation.

according to the classification of training patterns. Fig. 6(a) stands for initial state of the subnetwork. The weights of each node is randomly initialized. Unlabeled nodes indicates that these nodes does not selected at all in classification phase. Fig. 6(b) represents the final state of the subnetwork after parameter adaptation during 300 epoches.

As shown in this figure, the central result in this experiment reveals that the weights of nodes are organized so that the neighboring nodes on the subnetwork tend to have similar weight vectors which represent the neighboring regions in the input pattern space.

B. Experiments on the Classification of Large-Set Real World Patterns

1) Databases: In order to verify the performance of SAINT, several experiments have been conducted with large-set real world characters, on CRAY Y-MP Supercomputer. Two kinds of database were used in these experiments. One is PE92 database (type I) containing 100 sets of KS (Korean Standards) 2350 handwritten Korean characters written by more than 500 different writers [25]. And the other is a database (type II) containing 30 sets of 7320 multilingual, multifont, and multisize characters which are composed of 2350 Korean characters, 4888 Chinese characters, 62 alphanumeric characters, and 20 special

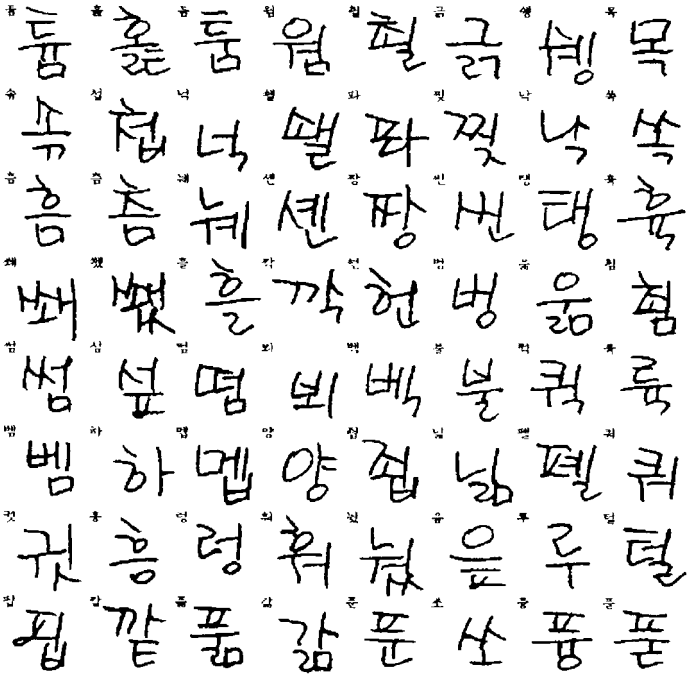


Fig. 7. Representative samples of handwritten Korean characters from PE92 database.

characters, respectively [26]. In Korea, these 7320 kinds of characters are daily used in newspapers, official document, and so on. The training set and testing set of each database consisted of the following:

- type I database (2350 handwritten Korean characters)
  - —Training data: 90 set (211 500 characters) written by 500 different scripters
  - —Testing data: 10 set (23 500 characters) written by another 23 scripters
- type II database (7320 multilingual, multifont, and multisize characters)
  - —Training data: 23 set (168 360 characters) scanned by Microtek ScanmakerII scanner
  - —Testing data: 7 set (51 240 characters) scanned by Microtek ScanmakerII scanner

144 dimensions of the dynamic mesh features [27] extracted from each character were normalized [28] and then used as an input to SAINT.

2) Experiments with the Type I Database: First of all, we performed experiments with the type I database. Fig. 7 shows some representative samples taken from PE92 database used in this experiment.

Three experiments were conducted using the most frequently used 520, 990, and the entire 2350 handwritten Korean characters. Table I shows the experimental results for each character set.

Next, we carried out experiments to compare SAINT with two-layer perceptron using backpropagation learning algorithm in view of the network size and the recognition rates with respect to the number of pattern classes. Two-layer perceptron used in these experiments has 144 input units, and 289 hidden units (this number is induced in Kolmogorov's

TABLE I
NETWORK SIZE AND RECOGNITION RATES FOR THE TYPE I DATABASE

Number of classes	Tree size	Recognition rates	Average distortion
520	1,128	90.7%	0.0021
990	2,446	89.1%	0.0024
2,350	6,183	85.3%	0.0032

TABLE II
COMPARATIVE RESULTS OF SAINT AND MLP

Criteria		SAINT	MLP
# of pattern classes = 50	# of connections	37,152[1)]	56,066[2)]
	Recognition rates	94.4%	95.6%
# of pattern classes = 90	# of connections	47,232	67,626
	Recognition rates	93.7%	92.8%
# of pattern classes = 130	# of connections	59,040	79,186
	Recognition rates	92.4%	88.3%

1) # of input dimension × # of nodes
2) (# of input dimension × # of hidden units) + (# of hidden units × # of output units)

TABLE III
COMPARISON OF SAINT WITH SPA NEURAL TREE ON THE ENTIRE SET OF THE TYPE I DATABASE

Criteria	SAINT	SPA neural tree
Number of internal nodes	735	1,779
Number of terminal nodes	5,448	5,601
Average height of terminals	6	16
Recognition rates	85.3%	73.9%

TABLE IV
COMPARISON OF SAINT WITH LVQ4SA ON THE ENTIRE SET OF THE TYPE I DATABASE

Criteria	SAINT	LVQ4SA
Number of internal nodes	735	0
Number of terminal nodes	5,448	5,448
Average height of terminals	6	0
Recognition rates	85.3%	81.5%
Recognition speed	19 mschar	363 mschar

theorem [29]), and the same number as class number to be classified in output units. In this experiments, we have seen that two-layer perceptron shows high recognition rate in the case of the small-set pattern classification, but as the number of pattern classes becomes larger, the network size and the training time exponentially increase. Especially, for the case in which the number of pattern classes is more than 150, we came to the conclusion that it reached to an unmanageable state. Thus, to extend two-layer perceptron to meet large-set pattern classification, some additional efforts should be required, such as preclassification which necessarily deteriorates the performance of the network. In this aspect, it is obvious that SAINT is adequate for the classification of large-set and complex patterns. Table II shows the comparative results for MLP and SAINT.

Next, we compared SAINT with SPA neural tree [20] which has similar structure with ours except that each subnetwork of SAINT has a lattice topology. Table III shows the comparative results for SPA neural tree and SAINT with respect to the number of pattern classes. As shown in this table, the total connections of SAINT are significantly less than those of SPA neural tree in views of the total number of nodes and the average height in the overall networks. It may be attributed to the fact SAINT is composed of subnetworks with topology-preserving ability. Especially, SAINT was shown to be superior to SPA neural tree for the case in which patterns have much more variations.

Finally, we compared SAINT with the improved LVQ3 (Learning Vector Quantization 3) combined with Simulated Annealing (SA), called LVQ4SA, which had been known as the method for the optimal or near-optimal design of large-set reference model [30]. In this method, SA, which is well known to provide good solutions to several problems including the optimization of multivariate functions, was used in order to overcome the drawback of LVQ-based algorithms due to a brute force acceptance of new reference models in each iterations. In this experiment, 5448 terminal nodes were also used in LVQ4SA in order to compare both methods on the basis of the recognition performance and speed. Table IV shows the comparative results for LVQ4SA and SAINT. As shown in this table, SAINT outperforms LVQ4SA. Especially, SAINT was shown to be superior to LVQ4SA with respect to the recognition speed.

3) Experiments with the Type II Database: In order to confirm the performance of SAINT, we also performed experiments with very large-set real world characters, that is, multilingual, multifont, and multisize large-set characters. Fig. 8 shows a typical example of the scanned characters used in this experiment.

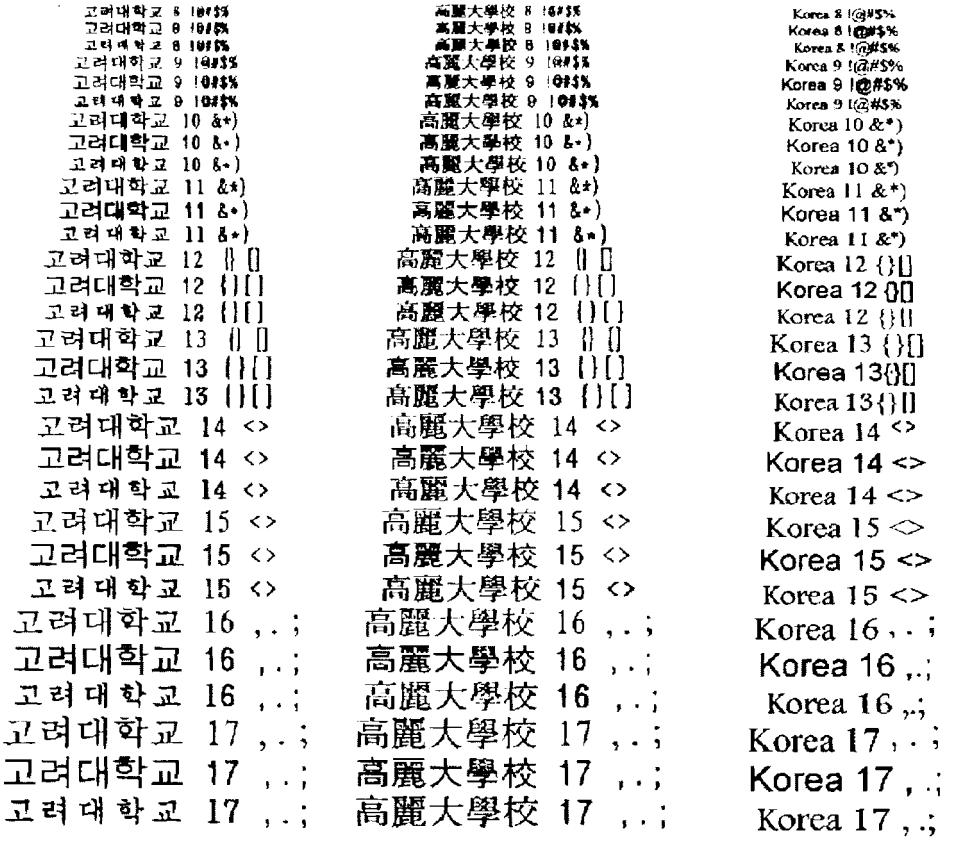


Fig. 8. A typical example of multilingual, multifont, and multisize character set.

TABLE V
NETWORK SIZE AND RECOGNITION RATE FOR THE TYPE II DATABASES

Number of classes	Tree size	Recognition rates	Average distortion
2,350(Korean)	6,183	98.6%	0.0020
4,888(Chinese)	9,254	98.1%	0.0031
7,320(the entire set)	16,084	97.5%	0.0046

Three experiments were conducted using KS 2350 printed Korean characters, 4888 Chinese characters, and the entire 7320 multilingual characters including Korean, Chinese, alphanumeric, and special characters, respectively. Of course, three different fonts and various sizes were contained in each character set.

First of all, we performed experiments to evaluate the performance of SAINT with respect to the number of pattern classes. Table V shows the experimental results for each character set.

Table VI shows the comparative results for SPA neural tree and SAINT with respect to the entire set of the type II database. Similarly to the experimental results for set of the type I database, the total connections of SAINT are significantly less than those of SPA neural tree in views of the number of nodes and the average height in the overall networks.

TABLE VI
COMPARISON OF SAINT WITH SPA NEURAL TREE ON THE ENTIRE SET OF THE TYPE II DATABASE

Criteria	SAINT	SPA neural tree
Number of internal nodes	1,313	1,751
Number of terminal nodes	16,771	18,124
Average height of terminals	5	12
Recognition rates	97.5%	88.6%
Average distortion	0.001	0.024

VI. DISCUSSIONS

In this section, we discuss important differences between MLP with backpropagation and SAINT. And then, we present the relationship on several networks which are closely related to SAINT.

As shown in the previous section, except the training speed, MLP showed as well as or better performance than SAINT for the case of small-set patterns. However, as the number of pattern classes becomes larger, the computational complexity of the learning process in MLP quickly reaches unmanageable proportions. MLP is very different from SAINT in several aspects. In the MLP, a single point in the input pattern space is distributed into all units of the network. On the other hands,

TABLE VII
COMPARISON OF SAINT WITH MLP

Property	SAINT	MLP
Decision region	Voronoi region	Convex and non-convex region
Convergence properties	Local or global MSE	Local MSE
Learning method	Unsupervised learning (Kohonen's learning algorithm)	Supervised learning (error backpropagation)
Method to overcome over/under-fitting	Adjustment of the network size	Adjustment of the number of hidden units
Activation function	Linear	Nonlinear
Weight update	Update of synaptic weights of winner and its neighbors	Update of all weights
Representation capacity	Large-set patterns	Small-set patterns

in SAINT, a single point in the input pattern space is mapped onto one node of the network. We summarized the different properties of MLP and SAINT in Table VII.

There are several neural network models which are closely related to SAINT. The tree-structured adaptive network [18] differs from SAINT in that it is intended for function approximation rather than classification tasks. In this network, a function is decomposed first along one dimension of the function to form one level of the tree, and then decomposed along other dimensions to form further levels of the tree. It also has not the successive approximation property. There exists the weight between parent and child nodes in which LMS (Least Mean Square) learning rule for updating the weight is used. Thus, it may generate the complex decision boundaries, but be inadequate for classifying large-set patterns.

SPA neural tree [20] is closely related to SAINT in that it hierarchically partitions the n-dimensional input pattern space into successive subregion. However, as mentioned in Section IV, SAINT differs from SPA neural tree in that each subnetwork has its topology.

VII. CONCLUDING REMARKS

In this paper, we have proposed a Structurally Adaptive Intelligent Neural Tree (SAINT) and itslearning algorithm. The basic idea is to partition hierarchically input pattern space using the tree-structured network with topology-preserving mapping ability. On the basis of this idea, we adopted the tree-structured network which is composed of the subnetwork with a lattice-topology, and also adopted the structure adaptation of the network in order to solve the network structure and size problems. The main advantage of SAINT is that it attempts to find automatically a network structure and size suitable for the classification of large-set and complex patterns through structurally adapting ability. By hierarchically and dynamically modeling its distribution according to the statistics of a given input pattern space, SAINT has been found:

1) to solve the large-set classification problem;
2) to overcome the complex input environment problem.

Experimental results revealed that SAINT is very effective for the classification of large-set real world patterns with high variations.

Our future works will be concerned with the more difficult tasks of dynamically generating only the required neighboring node in each subnetwork preserving its topology and of enhancing generalization ability.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers for their helpful comments leading to improvements of the earlier draft of this paper.

REFERENCES

[1] R. Schalkoff, *Pattern Recognition: Statistical, Structural, and Neural Approaches*. New York: Wiley, 1992.
[2] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[3] A. K. Jain, "Advances in statistical pattern recognition," in *Pattern Recognition Theory and Applications*, P. A. Devijver and J. Kittler, Eds. Berlin, Germany: Springer-Verlag, pp. 1–19, 1986.
[4] Q. R. Wang and C. Y. Suen, "Large tree classifier with heuristic search and global training," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 91–102, Jan. 1987.
[5] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of OCR research and development," *Proc. IEEE*, July 1992, vol. 80, no. 7, pp. 1029–1058.
[6] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, pp. 4–22, Apr. 1987.
[7] D. L. Reilly, L. N. Cooper, and C. Elbaum, "A neural model for category learning," *Biol. Cybern.*, vol. 45, pp. 35–41, 1982.
[8] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 826–834, 1983.
[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williarms, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Eds., vol. I. Cambridge, MA: MIT Press, 1986, pp. 318–364.
[10] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54–115, 1987.
[11] T. Kohonen, *Self-Organizing and Associative Memory*, 3rd ed. New York: Springer-Verlag, 1989.
[12] Y. Le Cun *et al.*, "Constrained neural network for unconstrained handwritten digit recognition," in *Proc. 1st Int. Wkshp. Frontiers in Handwriting Recognition*, Montreal, Canada, 1990, pp. 145–154.
[13] S.-W. Lee, "Multilayer cluster neural network for totally unconstrained handwritten numeral recognition," *Neural Networks*, vol. 8, no. 5, pp. 783–792, 1995.
[14] J. Moody and C. Darken, "Learning with localized receptive fields," in *Proc. 1988 Connectionist Models Summer School*, pp. 133–143.
[15] T. C. Lee and A. M. Peterson, "Adaptive vector quantization using a self-development neural network," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 1458–1471, Oct. 1990.

[16] J. Blackmore and R. Miikkulainen, "Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map," Dept. Computer Sci., Univ. Texas-Austin, Tech. Rep. AI92-192, Dec. 1992.
[17] B. Fritzke, "Growing cell structures—A self-organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, no. 9, pp. 1441–1460, 1994.
[18] 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.
[19] L. Fang, A. Jennings, W. X. Wen, K. Q.-Q. Li, and T. Li, "Unsupervised learning for neural tree," in *Proc. 1991 Int. Joint Conf. Neural Networks*, Singapore, vol. 1, Nov. 1991, pp. 2709–2715.
[20] 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.
[21] P. Patrick and M. Hasler, "Self-organization of a one-dimensional Kohonen network with quantized weights and inputs," *Neural Networks*, vol. 7, no. 9, pp. 1427–1439, 1994.
[22] J. Koh, M. Suk, and S. M. Bhandarkar, "A multilayer self-organizing feature map for range image segmentation," *Neural Networks*, vol. 8, no. 1, pp. 67–86, 1995.
[23] E. M. Rounds, "A combined nonparametric approach to feature selection and binary decision tree design," *Pattern Recognition*, vol. 12, pp. 313-317, 1980.
[24] C. Y. Suen, C. Nadal, T. A. Mai, R. Legault, and L. Lam, "Recognition of handwritten numerals based on the concept of multiple experts," in *Proc. 1st Int. Wkshp Frontiers in Handwriting Recognition*, Montreal, Canada, 1990, pp. 131–144.
[25] D. H. Kim, Y. S. Hwang, S. T. Park, E. J. Kim, S. H. Paek, and S. Y. Bang, "Handwritten Korean character image database PE92," in *Proc. 2nd Int. Conf. Document Anal. Recognition*, Tsukuba, Japan, 1993, pp. 470–473.
[26] S.-W. Lee and J.-S. Kim, "Multilingual, multifont and multisize large-set character recognition using self-organizing neural network," in *Proc. 3rd Int. Conf. Document Anal. Recognition*, Montreal, Canada, 1995, pp. 28–33.
[27] S.-W. Lee and J.-S. Park, "Nonlinear shape normalization methods for the recognition of large-set handwritten characters," *Pattern Recognition*, vol. 27, no. 7, pp. 895–902, 1994.
[28] P. Demartines and F. Blayo, "Kohonen self-organizing maps: Is the normalization necessary," *Complex Syst.*, vol. 6, pp. 105–123, 1992.
[29] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," *Trans. Amer. Math. Soc.*, vol. 28, pp. 55–59, 1963.
[30] H.-H. Song and S.-W. Lee, "LVQ Combined with simulated annealing for optimal design of large-set reference models," *Neural Networks*, vol. 9, no. 2, pp. 329–336, 1996.

Hee-Heon Song (M'96) received the B.S. degree in computer science from Dongkook University, Seoul, Korea, in 1986, the M.S. degree in computer science from Chungnam National University, Taejon, Korea, in 1992, and the Ph.D. degree in computer science from Chungbuk National University, Cheongju, Korea, in 1995.

From 1986 to 1998, he worked as a Senior Research Engineer at Electronics and Telecommunications Research Institute, Taejon, Korea. In March 1998, he joined the faculty of the Department of Computer Engineering Education, Andong National University, Andong, Korea. His research interests include neural networks, pattern recognition, and intelligent networks.

Dr. Song is a member of the IEEE Computer Society and the Korea Information Science Society.

Seong-Whan Lee (S'87–M'91–SM'96) received the B.S. degree in computer science and statistics from Seoul National University, Seoul, Korea, in 1984 and the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology in 1986 and 1989, respectively.

In 1987, he worked as a Visiting Researcher at the Pattern Recognition Division, Delft University of Technology, Delft, the Netherlands. He was a Visiting Scientist at the Centre for Pattern Recognition and Machine Intelligence, Concordia University, Montreal, Canada, during the winter of 1989 and the summer of 1990. From 1989 to 1994, he was an Assistant Professor in the Department of Computer Science, Chungbuk National University, Cheongju, Korea. In March 1995, he joined the faculty of the Department of Computer Science and Engineering, Korea University, Seoul, Korea, as an Associate Professor. He has more than 120 publications on pattern recognition and neural networks in International Journals and Conference Proceedings, and authored two Korean books: *Theory and Practice of Character Recognition* (Seoul, Korea: Hongneung Press, 1993) and *Principles of Pattern Recognition* (Seoul, Korea: Hongneung Press, 1994). His research interests include pattern recognition, document image analysis, computer vision, and neural networks.

Dr. Lee was the winner of the Annual Best Paper Award of the Korea Information Science Society in 1986. He obtained the First Outstanding Young Researcher Award at the Second International Conference on Document Analysis and Recognition in 1993, and the First Distinguished Research Professor Award from Chungbuk National University in 1994. In 1996, he also obtained the Outstanding Research Award from the Korea Information Science Society. He is the Coeditor-in-Chief of the *International Journal of Document Analysis and Recognition*, and the Associate Editor of the *Pattern Recognition Journal*, *International Journal of Pattern Recognition and Artificial Intelligence*, and *International Journal of Computer Processing of Oriental Languages*. He was the Program Chairman of the 17th International Conference on Computer Processing of Oriental Languages and the Sixth International Workshop on Frontiers in Handwriting Recognition. He is the General Cochairman of the Third International Workshop on Document Analysis Systems, and the Program Cochairman of the Fifth International Conference on Document Analysis and Recognition and Second International Conference on Multimodal Interface. He served on the program committees of several well-known international conferences. He is a senior member of the IEEE Computer Society and a member of the Korea Information Science Society, the Pattern Recognition Society, the International Neural Network Society, and the Oriental Language Computer Society.