An Improved Algorithm for Learning Long-Term Dependency Problems in Adaptive Processing of Data Structures

Siu-Yeung Cho, *Member, IEEE*, Zheru Chi, *Member, IEEE*, Wan-Chi Siu, *Senior Member, IEEE*, and Ah Chung Tsoi, *Senior Member, IEEE*

***Abstract*—For the past decade, many researchers have explored the use of neural-network representations for the adaptive processing of data structures. One of the most popular learning formulations of data structure processing is backpropagation through structure (BPTS). The BPTS algorithm has been successful applied to a number of learning tasks that involve structural patterns such as logo and natural scene classification. The main limitations of the BPTS algorithm are attributed to slow convergence speed and the long-term dependency problem for the adaptive processing of data structures. In this paper, an improved algorithm is proposed to solve these problems. The idea of this algorithm is to optimize the free learning parameters of the neural network in the node representation by using least-squares-based optimization methods in a layer-by-layer fashion. Not only can fast convergence speed be achieved, but the long-term dependency problem can also be overcome since the vanishing of gradient information is avoided when our approach is applied to very deep tree structures.**

***Index Terms*—Adaptive processing of data structures, backpropagation through structure (BPTS), least-squares method, long-term dependency.**

I. Introduction

MOST connectionist models assume that data are organized by relatively poor structures, such as arrays or sequences. It is a common opinion in the scientific community that quite a wide variety of real-world problems require hybrid solutions, i.e., solutions combining techniques based on neural networks, fuzzy logic, genetic algorithms, probabilistic networks, expert systems, and other natural or artificial techniques. In recent years, machine learning models conceived for dealing with sequences can be straightforwardly adapted to process data structures. The tree-structure processing by these specified models can carry out on sequential representations based upon the construction of the trees. However, this approach has two major drawbacks. First, the sequential mapping of data structures, which are necessary to break some regularities inherently associated with the data structure, will yield poor generalizations. Second, since the number of nodes grows exponentially with the depth of the trees, a lot of parameters need to be learned, which makes learning difficult and inefficient.

Neural networks for adaptive processing of data structure are of paramount importance for structural pattern recognition and classification [1]. The main motivation of this adaptive processing is that neural networks are able to classify static information or temporal sequences and to perform automatic inferring or learning [2], [3]. Sperduti and Starita have proposed supervised neural networks for the classification of data structures [4]. This approach is based on using generalized recursive neurons and a backpropagation through structure (BPTS) algorithm [5], [1]. Most recently, some advances in this area have been presented and some preliminary results have been obtained [6]–[8]. The basic idea of this algorithm is to extend a backpropagation through time (BPTT) algorithm [9] to encode data structures by recursive neurons. By recursive here means that a copy of the same neural network is used to encode every node of the tree. In the BPTT algorithm, the gradients of the weights to be updated can be computed by backpropagating the error through the time sequence. Similarly, if learning is performed on a data structure such as a directed acyclic graph, the gradients can be calculated by backpropagating the error through the data structures, which is known as the BPTS algorithm. However, this gradient-based learning algorithm has several shortcomings. First, the rate of convergence is slow so that the learning process cannot be guaranteed to be completed within a reasonable time for most complicated data structures. Although the algorithm can be accelerated simply by using a larger learning rate, this would probably introduce oscillation and might result in a failure in finding an optimal solution. Second, gradient-based algorithms are usually prone to local minima [10]. From a theoretical point of view, we believe that gradient-based learning is not very reliable for rather complex error surfaces formulated in the data structure processing. Third, it is extremely difficult for the gradient-based BPTS algorithm to learn a very deep tree structure because of the problem of long-term dependency [11], [12]. Indeed, the gradient contribution disappears at a certain tree level when the error backpropagates through the deep tree structure (the learning information is latched) to the terminal nodes. This is because the decreasing gradient terms tend to be zero since the backpropagating error is recursively multiplied by the derivative (between 0 and 1) of the sigmoidal function in each neural node.

Manuscript received October 29, 2001; revised August 5, 2002. This paper was supported by Hong Kong Polytechnic University under an ASD research grant (Project A408).

S.-Y. Cho, Z. Chi, and W.-C. Siu are with the Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: enzheru@polyu.edu.hk).

A. C. Tsoi is with Information Technology Services, University of Wollongong, Wollongong NSW 2522, Australia.

Digital Object Identifier 10.1109/TNN.2003.813831

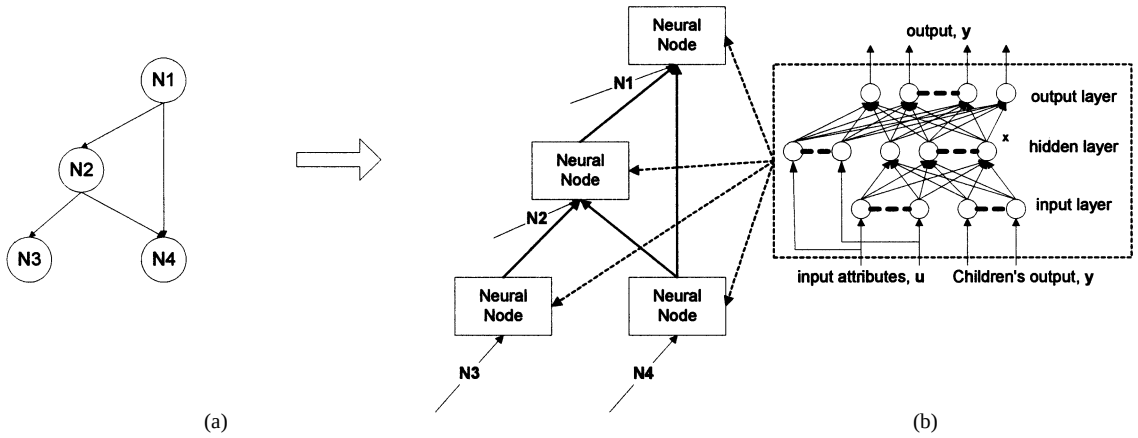


Fig. 1. Illustration of a data structure with its nodes encoded by a single-hidden-layer neural network. (a) Directed acyclic graph (DAG). (b) Encoded DAG.

This results in convergence stalling and yields a poor generalization.

In view of the rather complex error surfaces formulated by the adaptive processing of data structures, we need more sophisticated learning schemes to replace the gradient-based algorithm so as to avoid the learning parameters being converged to suboptimal solutions. These algorithms may include simulated annealing, genetic algorithms, and more specific heuristic learning algorithms [13]–[16]. Although they are able to tackle the local minima problem, the computational cost is relatively high. In this paper, an improved learning algorithm for adaptive processing of data structures is developed to reduce the aforementioned problems. This proposed learning scheme is modified from the derived methods reported in [15] and [16] and implemented into the recursive neural representation. The optimum values of the learning parameters in the recursive neural representation for tree structures are determined by the modified linear least squares method in a layer-by-layer fashion so that the convergence of the algorithm can be accelerated. The updated parameters do not need to be evaluated by the total gradients through data structures so that the problem of long-term dependency can be eliminated. The proposed algorithm has been validated by different simulation problems in structural representation, including a traffic policeman signaling simulation problem [1], a full binary tree structural problem, and a natural scene classification [18]. For the scene classifications, image contents can be represented by a tree-structure representation that can characterize the image features at multiscales as beneficial to image classification by using a small number of simple features. Experimental results illustrate that our proposed algorithm outperforms the BPTS algorithms. By using this improved algorithm, the learning performance of the tree structure data can be enhanced significantly in terms of convergence speed, quality of solution, and avoidance of the long-term dependency problem in the structural processing.

This paper is organized as follows: the basic idea of the adaptive processing of data structures is presented in Section II. A detailed discussion of the long-term dependency problem is given in Section III. The derivations of our proposed algorithm for the adaptive processing of data structure is given in Section IV. In Section V, the results of the simulations for tree structure tasks are discussed and we show that our proposed methodology is able to overcome the limitations of the BPTS algorithm. Finally, the conclusion is drawn in Section VI.

II. Adaptive Processing of Data Structures

In this paper, the problem of devising neural network architectures and learning algorithms for the adaptive processing of data structure is addressed for the classification of structured patterns. The encoding method by recursive neural networks was based on and modified by the research works of [1] and [4]. We consider that a structured domain D and all graphs (Trees are a special case of graphs. In the following discussion, we will use either graph or tree when it is appropriate.) G form learning set representing the task of the adaptive processing of data structures. This representation by the recursive neural networks is shown in Fig. 1.

As shown in Fig. 1, a copy of the same neural network [shown on the right-hand side of Fig. 1(b)] is used to encode every node in a graph G. Such an encoding scheme is flexible enough to allow the model to deal with DAGs of different internal structures and with a different number of nodes. Moreover, the model can also naturally integrate structural information into its processing. In a DAG shown in Fig. 1(a), the operation is run forward for each graph, i.e., from terminals nodes (N3 and N4) to the root node (N1). A maximum number of children for a node (i.e., the maximum branch factor c) is assumed for a task. For instance, a binary tree (each node has two children only) has a maximum branch factor c equal to two. At the terminal nodes, there will no inputs from children. Therefore, the terminal nodes are known as frontier nodes. The forward recall is in the direction from the frontier nodes to the root in a bottom up fashion. The bottom–up processing from a child node to its parent node can be denoted by an operator q^{-1}. Suppose that a maximum branch factor of c has been predefined, each of the form q_i^{-1}, $i = 1, 2, \ldots, c$, denotes the input from the ith child into the current node. This operator is similar to the shift operator used in the time series representation. Thus, the recursive network for the structural processing is formed as

$$\mathbf{x} = F_n\left(\mathbf{A}q^{-1}\mathbf{y} + \mathbf{B}\mathbf{u}\right) \tag{1}$$

$$\mathbf{y} = F_p\left(\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}\right) \tag{2}$$

where $\mathbf{x}$, $\mathbf{u}$, and $\mathbf{y}$ are the n-dimensional output vector of the n hidden layer neurons, the m-dimensional inputs to the neurons, and the p dimensional outputs of the neurons, respectively. q^{-1} is a notation indicating that the input to the node is taken from its child so that

$$q^{-1}\mathbf{y} = \begin{pmatrix} q_1^{-1}\mathbf{y} \\ q_2^{-1}\mathbf{y} \\ \vdots \\ q_c^{-1}\mathbf{y} \end{pmatrix}. \tag{3}$$

The parametric matrix $\mathbf{A}$ is defined as follows:

$$\mathbf{A} = (\,\mathbf{A}^1 \quad \mathbf{A}^2 \quad \cdots \quad \mathbf{A}^c\,) \tag{4}$$

where c denotes the maximum number of children in the graph. $\mathbf{A}$ is a $n \times (c \times p)$ matrix such that each $\mathbf{A}^k, k = 1, 2, \ldots, c$ is a $n \times p$ matrix, which is formed by the vectors $\mathbf{a}_j^i, j = 1, 2, \ldots, n$. The parameters $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are respectively $(n \times m)$, $(p \times n)$, and $(p \times m)$-dimensional matrices. $F_n(\cdot)$ and $F_p(\cdot)$ are n and p dimensional vectors, respectively, given as follows:

$$F_n(\alpha) = \begin{pmatrix} f(\alpha_1) \\ f(\alpha_2) \\ \vdots \\ f(\alpha_n) \end{pmatrix} \tag{5}$$

where $f(\alpha)$ is the nonlinear function defined as $f(\alpha) = 1/(1 + e^{-\alpha})$.

In accordance with the research work by Hammer and Sperschnedier [19], based on the theory of the universal approximation of the recursive neural networks, a single hidden layer is sufficient to approximate any complicated mapping problems. The input–output learning task can be defined by estimating the parameters $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ in the parameterization from a set of input–output examples. Each input-output example can be formed in a tree data structure consisting of a number of nodes with their inputs and target outputs. Each node's inputs are described by a set of attributes $\mathbf{u}$. The target output is denoted by $\mathbf{t}$, where $\mathbf{t}$ is a p-dimensional vector. So, the cost function is defined as a total sum-squared-error function

$$J = \frac{1}{2}\sum_{i=1}^{N_T} \left(\mathbf{t}_i - \mathbf{y}_i^R\right)^T \left(\mathbf{t}_i - \mathbf{y}_i^R\right) \tag{6}$$

where N_T is the total number of the learning data structures. $\mathbf{y}^R$ denotes the output at the root node. Note that in the case of structural learning processing, it is often assumed that the attributes $\mathbf{u}$ are available at each node of the tree. The main step in the learning algorithm involves the following gradient learning step:

$$\theta(k+1) = \theta(k) - \eta \left.\frac{\partial J}{\partial \theta}\right|_{\theta=\theta(k)} \tag{7}$$

where $\theta(k)$ denotes the free learning parameters $\theta : \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ at the kth iteration and η is a learning rate. $\partial J/\partial\theta|_{\theta=\theta(k)}$ is the partial derivative of the cost function with respect to θ evaluated at $\theta = \theta(k)$. The derivation of the learning algorithm involves the evaluation of the partial derivative of the cost function with respect to the parameters in each node. Thus, the general form of the derivatives of the cost functon with respect to the parameters is given by

$$\frac{\partial J}{\partial \theta} = -\sum_{i=1}^{N_T} \left(\mathbf{t} - \mathbf{y}_i^R\right)^T \Lambda\left(\mathbf{y}_i^R\right) \delta\left(\nabla_\theta \mathbf{x}_i\right) \tag{8}$$

where $\Lambda(\mathbf{y})$ is a $p \times p$ diagonal matrix defined by the first derivative of the nonlinear activation function. δ is defined as n-dimensional vector which is the function of the derivative of $\mathbf{x}$ with respect to the parameters. It can be evaluated as

$$\nabla_\theta \mathbf{x} = \Lambda(\mathbf{x})\,\mathbf{A} q^{-1} \frac{\partial \mathbf{y}}{\partial \theta} \tag{9}$$

where $\Lambda(\mathbf{x})$ is a $n \times n$ diagonal matrix defined in a similar manner as $\Lambda(\mathbf{y})$. It is noted that $q^{-1}\partial\mathbf{y}/\partial\theta$ essentially repeats the same computation such that the evaluation depends on the structure of the tree. This is so called either the folding architecture algorithm [5] or BPTS [4].

In the formulation of the learning structural processing task, it is not required to assume *a priori* knowledge of any data structures or any *a priori* information concerning the internal structures. However, we have assumed that the maximum number of children for each node in the tree is predefined. The parameterization of the structural processing problem is said to be an over-parameterization if the predefined maximum number of children is so much greater than that of real trees, i.e., there are many redundancy parameters in the recursive network than required to describe the behavior of the tree. The over-parameterization may give rise to the problem of local minima in the BPTS learning algorithm. Moreover, the long-term dependency problem may also affect the learning performance of BPTS approach due to the vanishing gradient information in learning deep trees. The learning information may disappear at a certain level of the tree before it reaches at the frontier nodes so that the convergence of the BPTS stalls and a poor generalization results. A detailed analysis of this problem will be given in the next section.

III. Long-Term Dependency Problems

For backpropagation learning of multilayer perceptron (MLP) networks, it is well known that if there are too many hidden layers, the parameters at very deep layers are not updated. This is because backpropagating errors are multiplied by the derivative of the sigmoidal function, which is between zero and one and, hence, the product of the derivatives and the gradient for very deep layers could become very small. Bengio *et al.* [11], [20] have analytically explained why backprop learning problems with the long-term dependency are difficult. They stated that the recurrent MLP network is able to robustly store information for an application of long temporal sequences when the states of the network stay within the vicinity of a hyperbolic attractor, i.e., the eigenvalues of the Jacobian are within the unit circle. However, Bengio *et al.* have shown that if its eigenvalues are inside the unit circle, then the Jacobian at each time step is an exponentially decreasing function. This implies that the portion of gradients becomes insignificant. This behavior is called the effect of vanishing gradient or forgetting

behavior [11]. In this section, we briefly describe some of the key aspects of the long-term dependency problem learning in the processing of data structures. The gradient based learning algorithm updates a set of parameters $\theta : \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ in the recursive neural network for node representation defined in (1) and (2) such that the updated parameter can be denoted as

$$\Delta\theta = \eta \nabla_\theta J \tag{10}$$

where η is a learning rate and ∇_θ is the matrix defined as

$$\nabla_\theta = \left[\frac{\partial}{\partial\theta_1} \quad \frac{\partial}{\partial\theta_2} \quad \cdots \quad \frac{\partial}{\partial\theta_n} \right]. \tag{11}$$

By using the chain rule, the gradient can be expressed as

$$\nabla_\theta J = -\sum_{i=1}^{N_T} \left(\mathbf{t}_i - \mathbf{y}_i^R\right)^T \nabla_{\mathbf{x}^R} \mathbf{y}_i^R \nabla_\theta \mathbf{x}^R. \tag{12}$$

If we assume that computing the partial gradient with respect to the parameters of the node representation at different levels of a tree is independent, the total gradient is then equal to the sum of these partial gradients as

$$\nabla_\theta J = -\sum_{i=1}^{N_T} \left(\mathbf{t}_i - \mathbf{y}_i^R\right)^T \nabla_{\mathbf{x}^R} \mathbf{y}_i^R \cdot \left(\sum_{l=1}^{R} J_{\mathbf{x}}^{R,R-l} \nabla_{\theta^l} \mathbf{x}^l \right) \tag{13}$$

where $l = 1, \ldots, R$ represents the levels of a tree and $J_{\mathbf{x}}^{R,R-l} = \nabla_{\mathbf{x}^l} \mathbf{x}^R$ denotes the Jacobian of (1) expanded over a tree from level R (root node) to l backwardly. Based on the idea of Bengio *et al.* [11], the Jacobian $J_{\mathbf{x}}^{R,n}$ is an exponentially decreasing function of n since the backpropagating error is multiplied by the derivative of the sigmoidal function which is between zero and one, so that $\lim_{n\to\infty} J_{\mathbf{x}}^{R,n} = 0$. This implies that the portion of $\nabla_\theta J$ at the bottom levels of trees is insignificant compared to the portion at the upper levels of trees. The effect of vanishing gradients is the main reason why the BPTS algorithm is not sufficiently reliable for discovering the relationships between desired outputs and inputs, which we term the problem of long-term dependency. Therefore, we are now proposing a layer-by-layer least-squares based method to avoid this effect of vanishing gradients by the BPTS algorithm so that the evaluation for updating the parameters becomes more robust in the problem of deep tree structures. The detailed deviations of our proposed algorithm will be described in the next section.

IV. Our Proposed Learning Algorithm

We consider that a multilayer preceptron network with a single hidden layer network is adopted to encode each node in a tree shown by (1)–(5). The free learning parameters of $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ are optimized to generalize the data structures. In our study, the cost function of (6) can be rewritten as

$$J = \frac{1}{2} \sum_{i=1}^{N_T} \left(\mathbf{d}_i^R - [\mathbf{C} \quad \mathbf{D}] \cdot \begin{bmatrix} \mathbf{x}_i^R \\ \mathbf{u}_i^R \end{bmatrix} \right)^T \cdot \left(\mathbf{d}_i^R - [\mathbf{C} \quad \mathbf{D}] \cdot \begin{bmatrix} \mathbf{x}_i^R \\ \mathbf{u}_i^R \end{bmatrix} \right) \tag{14}$$

where $\mathbf{d}_i^R = F_p^{-1}(\mathbf{t}_i)$ which is the inverse function of the target output at the root node. The task of the learning algorithm is to optimize the parameters of the network by minimizing the cost function as follows:

$$\min J = \min \frac{1}{2} \left\| \mathbf{d}^R - \mathbf{V} \cdot \mathbf{X}^R \right\|^2 \tag{15}$$

where $\|\cdot\|$ denotes the Euclidean norm. $\mathbf{d}^R = \left(\mathbf{d}_1^R, \ldots, \mathbf{d}_{N_T}^R\right)$ and $\mathbf{X}^R = \begin{pmatrix} \mathbf{x}_1^R, \ldots, \mathbf{x}_{N_T}^R \\ \mathbf{u}_1^R, \ldots, \mathbf{u}_{N_T}^R \end{pmatrix}$ represent the matrices of inversed function of the target output and the input patterns of the output layer, respectively. The matrix $\mathbf{V} = [\mathbf{C} \quad \mathbf{D}]$ denotes the parameters of the output layer of the network. All the derivatives of J with respect to $\mathbf{V}$ are

$$\frac{\partial J}{\partial \mathbf{V}} = -\left(\mathbf{d}^R \mathbf{X}^{R^T} - \mathbf{V} \cdot \mathbf{X}^R \mathbf{X}^{R^T} \right). \tag{16}$$

The optimal values of the parameters of the output layer of the network can be obtained by solving $\partial J / \partial \mathbf{V} = \mathbf{0}$

$$\mathbf{V} = [\mathbf{C} \quad \mathbf{D}] = \left(\mathbf{X}^R \mathbf{X}^{R^T} \right)^{-1} \left(\mathbf{d}^R \mathbf{X}^{R^T} \right) \tag{17}$$

where "-1" represents the pseudoinverse operation.

As the cost function in (14) is convex with respect to all parameters, the optimum values of these parameters can be determined by the linear least squares method so that the convergence of the algorithm is accelerated. Similarly, the parameters matrix of $\mathbf{W} = [\mathbf{A} \quad \mathbf{B}]$ is also updated by the layer-by-layer least squares method. The optimum values are again evaluated by taking the derivatives of the cost function J with respect to the parameters matrix $\mathbf{W}$ as

$$\frac{\partial J}{\partial \mathbf{W}} = \left[\frac{\partial J}{\partial \mathbf{A}} \quad \frac{\partial J}{\partial \mathbf{B}} \right] = \mathbf{0}. \tag{18}$$

The derivatives of J with respect to $\mathbf{A}$ and B are, respectively

$$\frac{\partial J}{\partial \mathbf{A}} = \begin{bmatrix} \frac{\partial J}{\partial \mathbf{a}_1} \\ \vdots \\ \frac{\partial J}{\partial \mathbf{a}_n} \end{bmatrix}, \text{ and } \quad \frac{\partial J}{\partial \mathbf{B}} = \begin{bmatrix} \frac{\partial J}{\partial \mathbf{b}_1} \\ \vdots \\ \frac{\partial J}{\partial \mathbf{b}_n} \end{bmatrix}. \tag{19}$$

By the chain rule

$$\frac{\partial J}{\partial \mathbf{a}_j} = \left(\sum_{i=1}^{N_T} \frac{\partial \mathbf{x}_i}{\partial \mathbf{a}_j} \cdot \frac{\partial J}{\partial \mathbf{x}_i} \right)^T. \tag{20a}$$

$$\text{and} \quad \frac{\partial J}{\partial \mathbf{b}_j} = \left(\sum_{i=1}^{N_T} \frac{\partial \mathbf{x}_i}{\partial \mathbf{b}_j} \cdot \frac{\partial J}{\partial \mathbf{x}_i} \right)^T \quad j = 1 \ldots n. \tag{20b}$$

We can obtain the following equations:

$$\frac{\partial J}{\partial \mathbf{x}_i} = -\mathbf{C}^T \left(\mathbf{d}_i^R - \mathbf{C}\mathbf{x}_i \right) \tag{21}$$

$$\frac{\partial \mathbf{x}_i}{\partial \mathbf{a}_j} = \left[\Lambda(\mathbf{x}) \left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) + \mathbf{A} q^{-1} \frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_j} \right) \right]^T \tag{22}$$

$$\text{and} \quad \frac{\partial \mathbf{x}_i}{\partial \mathbf{b}_j} = \left[\Lambda(\mathbf{x}) \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A} q^{-1} \frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_j} \right) \right]^T \tag{23}$$

where $\Lambda(\mathbf{x})$ is a $n \times n$ diagonal matrix defined by the first derivative of the sigmoid activation function. $\mathbf{H}\left(q^{-1}\mathbf{y}_i\right) = \mathbf{Q}_y \cdot \operatorname{diag}\left(q^{-1}\mathbf{y}_i\right)$ and $\mathbf{H}(\mathbf{u}_i) = \mathbf{Q}_u \cdot \operatorname{diag}(\mathbf{u}_i)$, where $\mathbf{Q}_y$ and $\mathbf{Q}_u$ denote $n \times (c \times p)$ and $n \times m$ matrices, respectively, with all

elements being "1." $\mathrm{diag}\left(q^{-1}\mathbf{y}_i\right)$ is a $(c\times p)\times(c\times p)$ diagonal matrix of $q^{-1}\mathbf{y}_i$ and $\mathrm{diag}\left(\mathbf{u}_i\right)$ is a $m\times m$ diagonal matrix of $\mathbf{u}_i$.

Let $\mathbf{e}_i^R=\left(\mathbf{d}_i^R-\mathbf{C}\mathbf{x}_i\right)$ which defines the linear error signal between the outputs of desired state and the root node, (20a) and (20b) become, respectively

$$\frac{\partial J}{\partial \mathbf{a}_j}=-\sum_{i=1}^{N_T}\mathbf{e}_i^{R^T}\mathbf{C}\Lambda\left(\mathbf{x}\right)\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_j}\right) \tag{24a}$$

$$\text{and } \frac{\partial J}{\partial \mathbf{b}_j}=-\sum_{i=1}^{N_T}\mathbf{e}_i^{R^T}\mathbf{C}\Lambda\left(\mathbf{x}\right)\left(\mathbf{H}\left(\mathbf{u}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_j}\right). \tag{24b}$$

If

$$\mathbf{e}_i^{hid^T}=\mathbf{e}_i^{R^T}\mathbf{C}\Lambda\left(\mathbf{x}\right) \tag{25}$$

denotes the error signal between the desired and the output states at the hidden layer, then the desired state at the hidden layer can be defined as $\mathbf{z}_i=\left(\mathbf{x}_i+\mathbf{e}_i^{hid}\right)$. In our study, $\mathbf{z}_i$ can be approximated by a first-order Taylor series as

$$\begin{aligned}\mathbf{z}_i\left(\mathbf{A},\mathbf{B}\right)\approx&\mathbf{z}_i\left(\mathbf{A}_0,\mathbf{B}_0\right)+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{a}_j}\right)^T\Delta\mathbf{a}_j\\&+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{b}_j}\right)^T\Delta\mathbf{b}_j+\cdots\\&\approx\left(\mathbf{A}_0q^{-1}\mathbf{y}_i+\mathbf{B}_0\mathbf{u}_i+\mathbf{e}_i^{hid}\right)+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{a}_j}\right)^T\Delta\mathbf{a}_j\\&+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{b}_j}\right)^T\Delta\mathbf{b}_j+\cdots\\&\approx\left(\mathbf{A}_0q^{-1}\mathbf{y}_i+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{a}_j}\right)^T\Delta\mathbf{a}_j+\cdots\right)\\&+\left(\mathbf{B}_0\mathbf{u}_i+\sum_{j=1}^{n}\left(\frac{\partial \mathbf{z}_i}{\partial \mathbf{b}_j}\right)^T\Delta\mathbf{b}_j+\cdots\right)+\mathbf{e}_i^{hid}\\\approx&\mathbf{z}_i\left(\mathbf{A}\right)+\mathbf{z}_i\left(\mathbf{B}\right)+\mathbf{e}_i^{hid}.\end{aligned} \tag{26}$$

Therefore, the following two approximated functions can be obtained:

$$\mathbf{z}_i\left(\mathbf{A}\right)\approx\hat{\mathbf{A}}q^{-1}\mathbf{y}_i+\gamma\mathbf{e}_i^{hid} \tag{27a}$$

$$\mathbf{z}_i\left(\mathbf{B}\right)\approx\hat{\mathbf{B}}\mathbf{u}_i+\left(1-\gamma\right)\mathbf{e}_i^{hid} \tag{27b}$$

where γ denotes the portion of error contributions. In our experimental study, γ is set to 0.5.

Therefore, from (24a), (26), and (27a), the derivatives of J with respect to $\mathbf{A}$ can be written as

$$\frac{\partial J}{\partial \mathbf{A}}=-2\cdot\begin{bmatrix}\sum_{i=1}^{N_T}\left(\mathbf{z}_i\left(\mathbf{A}\right)-\hat{\mathbf{A}}q^{-1}\mathbf{y}_i\right)^T\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_1}\right)\\\vdots\\\sum_{i=1}^{N_T}\left(\mathbf{z}_i\left(\mathbf{A}\right)-\hat{\mathbf{A}}q^{-1}\mathbf{y}_i\right)^T\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_n}\right)\end{bmatrix}. \tag{28}$$

If we have (29)–(30), shown at the bottom of the page, where the desired state at the hidden layer is approximated as $\mathbf{z}_i\left(\mathbf{A}\right)=\mathbf{A}q^{-1}\mathbf{y}_i+0.5\mathbf{e}_i^R\mathbf{C}\Lambda\left(\mathbf{x}\right)$. The derivative $q^{-1}\partial\mathbf{y}_i/\partial\mathbf{a}_j=\begin{bmatrix}\partial q^{-1}\mathbf{y}_i/\partial\mathbf{a}_j^1\\\vdots\\\partial q^{-1}\mathbf{y}_i/\partial\mathbf{a}_j^c\end{bmatrix}$ is a $(c\times p)\times(c\times p)$ matrix with the output gradients of the child nodes with respect to the $\mathbf{A}$ parameters. Generally, if the number of tree level, $l=1,\ldots,L-1$, where L is the total tree level, the derivative

$$\frac{\partial q^{-l}\mathbf{y}_i}{\partial \mathbf{a}_j^c}=\mathbf{C}\Lambda\left(\mathbf{x}\right)\left(\mathbf{H}\left(q^{-(l+1)}\mathbf{y}_i\right)+\mathbf{A}q^{-(l+1)}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_j^c}\right). \tag{31}$$

$$\begin{bmatrix}\sum_{i=1}^{N_T}\left(\mathbf{z}_i\left(\mathbf{A}\right)-\hat{\mathbf{A}}q^{-1}\mathbf{y}_i\right)^T\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_1}\right)\\\vdots\\\sum_{i=1}^{N_T}\left(\mathbf{z}_i\left(\mathbf{A}\right)-\hat{\mathbf{A}}q^{-1}\mathbf{y}_i\right)^T\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_n}\right)\end{bmatrix}=\begin{bmatrix}\mathbf{0}\\\vdots\\\mathbf{0}\end{bmatrix} \tag{29}$$

$$\text{then}\quad\hat{\mathbf{A}}=\begin{bmatrix}\sum_{i=1}^{N_T}q^{-1}\mathbf{y}_i^T\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_1}\right)^T\\\vdots\\\sum_{i=1}^{N_T}q^{-1}\mathbf{y}_i^T\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_n}\right)^T\end{bmatrix}^{-1}\cdot\begin{bmatrix}\sum_{i=1}^{N_T}\mathbf{z}_i\left(\mathbf{A}\right)^T\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_1}\right)\\\vdots\\\sum_{i=1}^{N_T}\mathbf{z}_i\left(\mathbf{A}\right)^T\left(\mathbf{H}\left(q^{-1}\mathbf{y}_i\right)+\mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{a}_n}\right)\end{bmatrix} \tag{30}$$

Similarly, the derivatives of J with respect to the parameters $\mathbf{B}$ from (20b), (24b), and (27b)

$$\frac{\partial J}{\partial \mathbf{B}} = -2 \cdot \begin{bmatrix} \sum_{i=1}^{N_T} \left(\mathbf{z}_i(\mathbf{B}) - \hat{\mathbf{B}}\mathbf{u}_i\right)^T \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_1}\right) \\ \vdots \\ \sum_{i=1}^{N_T} \left(\mathbf{z}_i(\mathbf{B}) - \hat{\mathbf{B}}\mathbf{u}_i\right)^T \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_n}\right) \end{bmatrix}. \tag{32}$$

If

$$\begin{bmatrix} \sum_{i=1}^{N_T} \left(\mathbf{z}_i(\mathbf{B}) - \hat{\mathbf{B}}\mathbf{u}_i\right)^T \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_1}\right) \\ \vdots \\ \sum_{i=1}^{N_T} \left(\mathbf{z}_i(\mathbf{B}) - \hat{\mathbf{B}}\mathbf{u}_i\right)^T \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_n}\right) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \tag{33}$$

then

$$\hat{\mathbf{B}} = \begin{bmatrix} \sum_{i=1}^{N_T} \mathbf{u}_i^T \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_1}\right)^T \\ \vdots \\ \sum_{i=1}^{N_T} \mathbf{u}_i^T \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_n}\right)^T \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum_{i=1}^{N_T} \mathbf{z}_i(\mathbf{B})^T \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_1}\right) \\ \vdots \\ \sum_{i=1}^{N_T} \mathbf{z}_i(\mathbf{B})^T \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-1}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_n}\right) \end{bmatrix} \tag{34}$$

where the desired state at the hidden layer are approximated as $\mathbf{z}_i(\mathbf{B}) = \mathbf{B}\mathbf{u}_i + 0.5\mathbf{e}_i^R \mathbf{C}\Lambda(\mathbf{x})$. The derivative

$$q^{-1}\partial \mathbf{y}_i / \partial \mathbf{b}_j = \begin{bmatrix} \partial q^{-1}\mathbf{y}_i / \partial \mathbf{b}_j^1 \\ \vdots \\ \partial q^{-1}\mathbf{y}_i / \partial \mathbf{b}_j^c \end{bmatrix}$$

is a $(c \times p) \times m$ matrix with the output gradients of the child nodes with respect to the $\mathbf{B}$ parameters. Again, if the number of tree level, $l = 1, \ldots, L-1$, where L is the total tree level, the derivative

$$\frac{\partial q^{-l}\mathbf{y}_i}{\partial \mathbf{b}_j^c} = \mathbf{C}\Lambda(\mathbf{x}) \left(\mathbf{H}(\mathbf{u}_i) + \mathbf{A}q^{-(l+1)}\frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_j^c}\right). \tag{35}$$

Computationally, the evaluations of these two derivatives [in (31) and (35)] depend on the structure of the tree and essentially repeats the computations propagating through the structure backwardly. For instance, the total derivatives of the output nodes with respect to the $\mathbf{A}$ parameters for the whole structure are, generally, defined as

$$\frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{a}_j^c} = \mathbf{C}\Lambda(\mathbf{x})\mathbf{H}\left(q^{-2}\mathbf{y}_i\right) + \mathbf{C}\Lambda(\mathbf{x})\mathbf{A} \cdot \left(\mathbf{C}\Lambda(\mathbf{x})\left(\mathbf{H}\left(q^{-3}\mathbf{y}_i\right) + \cdots \mathbf{A} \cdot \left(\mathbf{C}\Lambda(\mathbf{x})\left(\mathbf{H}\left(q^{-(L-1)}\mathbf{y}_i\right) + \mathbf{A}\right)\right)\right)\right). \tag{36}$$

From the above discussion, suppose we are dealing with an extremely deep tree structure by this proposed algorithm, let $L \to \infty$, so the total derivatives can be simplified by taking

$$\lim_{L\to\infty} \frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{a}_j^c} \approx \mathbf{C}\Lambda(\mathbf{x})\mathbf{H}\left(q^{-2}\mathbf{y}_i\right) \tag{37}$$

as the expression at right-hand side tends to be zero because of infinite multiples of $\Lambda(\mathbf{x})$ (it is noted that all elements of $\Lambda(\mathbf{x})$ are between zero to one). Similarly, the total derivatives of the output nodes with respect to the $\mathbf{B}$ paratemetes for the whole structure are obtained by the same manner as

$$\lim_{L\to\infty} \frac{\partial q^{-1}\mathbf{y}_i}{\partial \mathbf{b}_j^c} \approx \mathbf{C}\Lambda(\mathbf{x})\mathbf{H}(\mathbf{u}_i). \tag{38}$$

Based on the above evaluations, the effect of vanishing gradients can be avoided so that the problem of long-term dependency can be eliminated for the processing in a deep tree structure.

In accordance with the above updated (17), (30), and (34), the parameters ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$) of the node representation can be evaluated by the least squares method and a suboptimal solution can be found in a few iterations. This is fairly efficient when operating under simple convex error surfaces with only a unique minimum. But under highly convoluted nonconvex surfaces encountered in practical problems of data structure representation, the least squares methods often lead to nonoptimal or unacceptable solutions. Once the algorithm is trapped into a nonoptimal state, the learning performance cannot be improved by increasing the number of iterations; hence the convergence stalls. Thus, in this paper, a heuristic mechanism is proposed to provide a force trajectory that will yield increasing opportunities of obtaining an optimal solution. Our proposed mechanism introduces a penalty term into the suboptimal solutions when the convergence stalls. The element of penalty term is generated by a Gaussian random generator to provide a random search in the corresponding parameters space. The search is performed repeatedly until the cost function converges again outside the suboptimal state. Afterwards, the learning procedure continues to search locally by using the least squares method. According to the above statements, the updated set of parameters $\theta : \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ formulation by the heuristic can be written as

$$\theta^{**} = \hat{\theta}(k-1) + \alpha\left(\theta^* - \hat{\theta}(k-1)\right) + \beta\Phi \tag{39}$$

where θ^{**} represents a new solution of the estimated parameters of the node representations. $\hat{\theta}(k-1)$ is the solution at the pervious iteration and θ^* represents the suboptimal state. α is a positive constant which is less than but close to unity. $\Phi = \mathbf{U} \cdot \|\partial J / \partial \theta\|$, where $\mathbf{U}$ denotes a matrix with the same size as the estimated parameters, which consists of a set of unit vectors generated by the Gaussian random generator to provide a random search direction to force trajectory out of the suboptimal state when the convergence stalls. $\|\partial J / \partial \theta\|$ is a norm of the total gradient matrices through the whole structure which is evaluated by the components of the derivatives shown in (37) and (38) approximately. The step-size of the search is denoted

by β of which its value increases gradually by a small fraction to enhance the searching ability. The increasing quantity of the step-size is a critical consideration according to the following conditions.

1) If the increase in β is too large, a significant change in the estimated parameters is possible because of the strong effect of the penalty term. In this case, the algorithm is unstable since it may result in an invalid solution.
2) If the increase in β is too small, only a small change in the parameters is allowed for the trajectory to escape because the penalty term is virtually redundancy.

As a result, in our experimental study, the quantity of step-size β is increased by 0.5% of the previous values until an escape from the suboptimal state is achieved.

The whole learning mechanism of our proposed algorithm for the data structure processing is summarized as follows.

Step 1: Algorithm initialization, set $n \leftarrow 0$. Randomize the parameters ($\theta : \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$) of the neural network in the node representation of data structure. Propagate the input attributes of each node through the data structure from frontier nodes to root forwardly and, hence, obtain the output of root node ($\mathbf{y}^R$).
Step 2: Define the stopping criterion for the learning convergence and set the values of the heuristic learning parameters α and β.
Step 3: While the stopping criterion is not met do
(a) $n \leftarrow n + 1$.
(b) For $i := 1$ to N_T do
Update the parameters $\theta : \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ through the whole structure by (17), (30) and (34).
(c) Calculate an error between the desired output and the root output of all learning examples by the new updated parameters.
(d) If the learning convergence stalls (i.e., the current value of the error is not less than the previous value), then use the heuristic shown in (39) in order to update the parameters until the convergence continues to converge.
Step 4: Store the updated parameters of the neural networks in the node representation.

V. Experimental Results and Discussions

This section presents the performance of our proposed algorithm for overcoming the long-term dependency problem in the adaptive processing of data structures. Several experiments were conducted for performance validation. In all cases, each node of data structures is encoded by a single-hidden-layer network. The initialization is performed by taking the free-learning parameters of the node randomly in the range of -1.0 to 1.0 with uniform distribution. The performance of our proposed algorithm is compared with different types of Backprop Through Structures (BPTS) algorithm reported in [1]. Mean-square error (MSE) is defined by

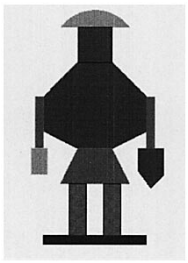


Fig. 2. Example of traffic policeman by combining primitives.

$$\text{MSE} = \frac{\sum_{i=1}^{N_T} \left(\mathbf{t}_i - \mathbf{y}_i^R\right)^T \left(\mathbf{t}_i - \mathbf{y}_i^R\right)}{p \cdot N_T} \tag{40}$$

that is used as a metric for the stopping criterion, where p is the number of outputs of the neural network and N_T is the size of learning samples.

A. Performance Evaluation on the Traffic Policeman Signaling Simulation Problem

This problem has been considered by the BPTS algorithm proposed in [1] which presented a well-controlled environment for validation. A number of primitives are adopted, which include hat, face, body, left arm, left hand object, right arm, right hand object, skirt or pants, and pedestal. These primitives can be described by a number of input attributes, for examples, color, shape, and position. These primitives can be joined together to form a policeman signifying a "go" traffic or a "stop" traffic concept. Fig. 2 presents an example of the traffic policeman composed of these primitives. A tree representation of this example is shown in Fig. 3. In this problem, 1057 samples are given and all the samples are varied by the different attributes of the primitives. Half of the samples represent the concept of "go" traffic and the remaining represent the concept of "stop" traffic.

In the learning task, a grammatical knowledge of the traffic policeman is not required and instead we only assume that the information about the policeman image representation is given. Basically, the image representation of the policeman composed of different parts is assumed to make use of *a priori* knowledge in the feature extraction. Each traffic policeman is divided into nine objects and 18 features are extracted by the x and y coordinates of the center of mass of these nine objects. These 18 features are used as the input to the learning algorithm. Five hundred structure patterns were generated for the learning set and the remaining 557 structure patterns were used for validation. A single hidden layer neural network has been used with only two inputs and two outputs. The number of hidden units can be varied and we use both four and six hidden units for our investigations. The performance of our proposed algorithm

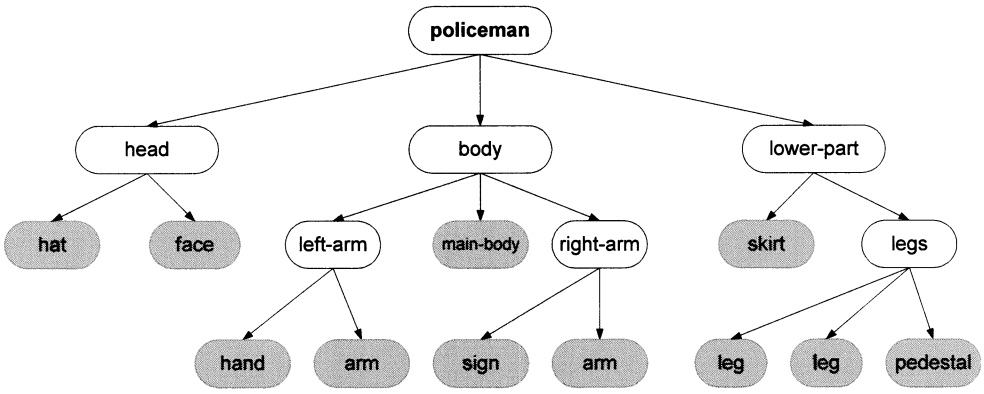
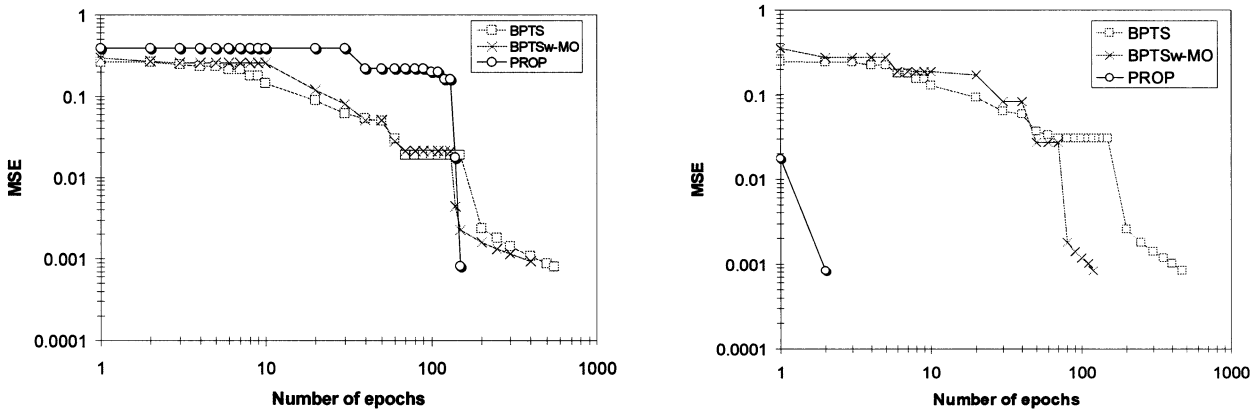


Fig. 3. Tree representation of the traffic policeman. (Shading blocks represent the primitives).

TABLE I
PERFORMANCE OF THE PROPOSED AND THE BPTS TYPE ALGORITHMS ON THE TRAFFIC POLICEMAN BENCHMARK PROBLEM

Algorithm	Number of hidden units	Number of epochs at the stopping criterion	CPU learning time required (sec)	Classification rate for validation
BackProp Through Structures (*BPTS*)	4	553	144	94.4%
	6	469	145	94.4%
BackProp Through Structures with momentum (*BPTSw-MO*)	4	140	43	92.2%
	6	72	22	89.4%
Proposed LS based with heuristic (*PROP*)	4	49	59	99.2%
	6	1	2	94.6%



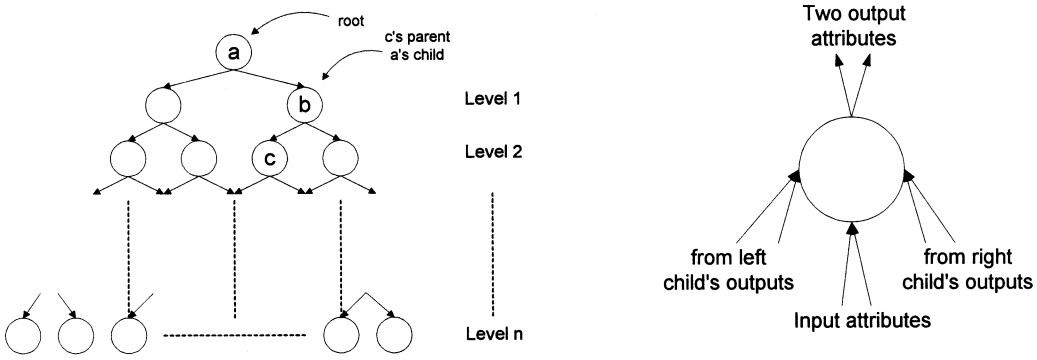
(a) (b)

Fig. 4. Convergence performances of the different algorithms for adaptive processing of data structures in traffic policemen signalling simulation. (a) Four hidden units used. (b) Six hidden units used.

is compared to the BPTS algorithms and their results are tabulated in Table I. Also, the convergence performances of all these compared algorithms are demonstrated in Fig. 4. It is noted that the proposed algorithm has the advantage of faster convergence speed with about 50 iterations being required. Although our proposed algorithm has a relative high computational cost for each iteration, the CPU time required for our algorithm is still much lower than that of the BPTS algorithms because of the faster rate of convergence yielded by the proposed learning algorithm. In terms of its generalization capability for the adaptive processing of data structures, our proposed algorithm produces the best solution with an accuracy of over 95% classification by overcoming the long-term dependency problem and by avoiding the local minima common to BPTS-type algorithms.

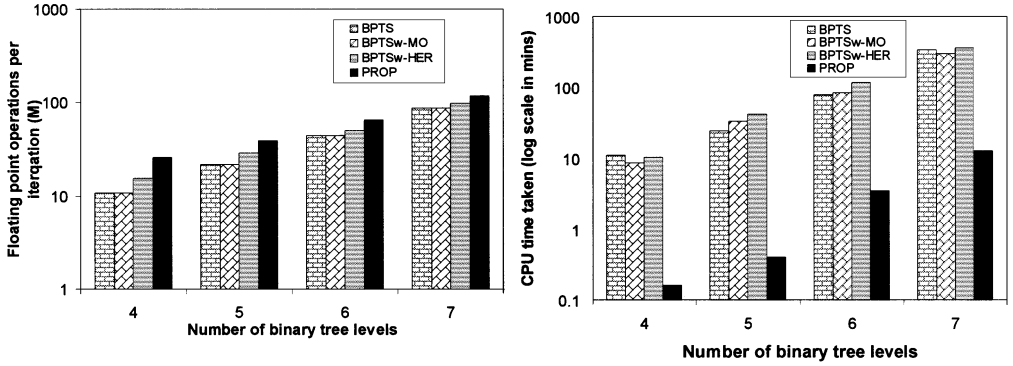
B. Performance Evaluation on Binary Tree Structures

A binary tree structure is essentially a generalization of an object-oriented representation, which has been extensively discussed in recent years [20]–[22]. Binary trees concentrate in a compact and structured representation for a set of meaningful objects that can be extracted from any complex domain. They offer a multiscale representation of the domain and define a translation invariant 2-connectivity rule among objects. This representation can be used for a large number of processing



(a) (b)

Fig. 5. Binary tree structure. (a) An example of n-level binary tree. (b) A single node representation in the binary tree.



(a) (b)

Fig. 6. Comparison results in computational demand of the different algorithms for adaptive processing of data structures in full binary tree structures. (a) Floating-point operation per iteration versus binary tree level. (b) CPU time taken versus binary tree level.

goals such as filtering, segmentation, information retrieval, and visual browsing. An example of a full binary tree is shown in Fig. 5(a). The figure illustrates graphically a full binary tree of level n with the maximum number of children for each node being equal to two. In an oriented tree, we will name one child the left child and the other the right child. It is known that there is a problem of long-term dependency if the tree is deep. In our experimental study, we generated full binary tree structures with different levels for performance evaluation. All nodes in the binary trees have two input attributes and two outputs. This node representation is shown in Fig. 5(b). These two attributes are generated by rounding-up values from a uniform random generator. In this simulation, we assume that each node performs a 6-bit parity checking. The desired outputs are either $[1 \quad 0]$ if odd parity is obtained or $[0 \quad 1]$ if even parity is obtained. Based on this representation, two classes can be produced at the root node which depend on the state of input attributes. It is very useful for evaluating algorithm performances and has been regarded as a challenging problem for the adaptive processing of data structures. The performance of our approach was compared with the BPTS algorithms, including the general BPTS, the BPTS with momentum term, and the BPTS integrated with our proposed heuristics. In all cases, 1000 samples were used for learning and other 1000 samples for testing. The results were averaged by running 50 trials independently and their classification rates are tabulated in Table II. It is noted that our proposed algorithm exhibits the best solution with an average accuracy rate of 90% whereas the BPTS algorithm cannot achieve the similar performance even if it is integrated with a heuristic as what we have done to our improved algorithm. This suggests that simply using a heuristic cannot solve the problem. Experimental results show that significant high classification rates can be obtained by our approach for deep tree structures, indicating that the problem of long-term dependency has been overcome. Moreover, the CPU time taken for these tested algorithms shown in Fig. 6 indicates that our approach provides a faster convergence speed although it has higher computational complexity per iteration than the BPTS type algorithms. The ratio of the numbers of floating point operations (Flops) per iteration between the BPTS and our proposed algorithm is 1: 1.3 for a deep tree (up to seven levels) implementation.

C. Performance Evaluation on Natural Scene Classification

Scene image content representation has been a popular research area in various images processing applications for the past few years. Most of the approaches represent the image content using only low-level visual features either globally or locally. It is noted that global features cannot characterize the image accurately; whereas, local features sometimes depend on error segmentation results. In our study, natural scene classification was performed by using a binary tree image representation method in which binary space partition (BSP) method [17], [18], [22], [23] was used to partition the image into subimages. The

TABLE II
CLASSIFICATION PERFORMANCES OF FULL BINARY TREE STRUCTURE REPRESENTATION

Full binary tree structure		Algorithms	Classification rate	
Number of levels n	Total number of nodes		Learning set Mean/Variance	Testing set Mean/Variance
4	31	BackProp Through Structures (*BPTS*)	85%/0.53%	86%/0.98%
		BackProp with momentum Through Structures (*BPTSw-MO*)	82%/0.67%	82.5%/1.01%
		BackProp Through Structures with heuristic (*BPTSw-HER*)	90%/0.94%	90%/1.12%
		Proposed LS based with heuristic (*PROP*)	**93%/0.91%**	**95%/1.05%**
5	63	BackProp Through Structures (*BPTS*)	78%/0.55%	49%/0.98%
		BackProp with momentum Through Structures (*BPTSw-MO*)	74%/0.71%	69%/1.00%
		BackProp Through Structures with heuristic (*BPTSw-HER*)	85%/0.98%	65%/1.16%
		Proposed LS based with heuristic (*PROP*)	**90%/0.62%**	**89%/0.71%**
6	127	BackProp Through Structures (*BPTS*)	76%/0.64%	59%/1.01%
		BackProp with momentum Through Structures (*BPTSw-MO*)	79%/0.66%	63%/1.00%
		BackProp Through Structures with heuristic (*BPTSw-HER*)	82%/0.88%	64%/1.09%
		Proposed LS based with heuristic (*PROP*)	**90%/0.73%**	**88%/0.77%**
7	255	BackProp Through Structures (*BPTS*)	76%/0.77%	51%/1.05%
		BackProp with momentum Through Structures (*BPTSw-MO*)	55%/0.81%	45%/1.09%
		BackProp Through Structures with heuristic (*BPTSw-HER*)	78%/1.02%	61%/1.19%
		Proposed LS based with heuristic (*PROP*)	**90%/0.84%**	**91%/0.91%**

BSP partitions the desired image recursively by a straight line in a hierarchical manner. First, an arbitrarily oriented line is selected, based on an appropriate criterion, to partition the whole image into two subimages. Using the same or another criterion, two lines are selected to split the two subimages resulting from the first partition. This procedure is repeated until a terminating criterion is reached. The outcome of this recursive partitioning is a set of unpartitioned regions that are referred to as the cells of the segmented image, and a binary tree that is referred to as the BSP tree representation of that image. The root node of the tree represents the whole image and the frontier nodes represent the cells of the image. A nonfrontier node of the BSP tree is associated with a subimage that are further partitioned by a partitioning line into two subregions each of which is associated with a child node. Each node of the BSP tree representation may have one or more attributes that describe the characteristics of the image region.

This section reports the scene classification performance of our proposed learning algorithm of BSP tree structures. This tree representation based image classification was tested on 40 scenery views of seven different categories, namely, apartments, beaches, campuses, flowers, playgrounds, grass, and trees, in which there were around three–seven samples taken from different perspectives for each scenery view. Fig. 7 shows samples of the BSP partitioned images and their BSP tree representations for different scene images. Each image was represented by a BSP tree constructed based on the BSP line selection and seven visual features were extracted as the attributes for each node of the BSP tree. As can be seen in Fig. 7, different BSP trees are used to represent the different image contents. Basically, the attributes associated with the root node are the global features of the whole images and the attributes of the child nodes characterize the local features of the subregions or objects. In our work, seven visual features including four color attributes, two texture attributes, and one area attribute contibuting to the image content were extracted. The color attributes include the number of quantized colors as well as the percentages of the three most dominant colors in the RGB color space. The average pixel value and the variance of the region pixels are used to roughly characterize the texture of the image region. The area of the partitioned region is used to provide some cue about the region or object. For our investigation in scene classification, a learning tree-structure set and a validation tree-structure set of each scenery image were generated. Each scenery image was used to generate four different trees by setting slightly different thresholds. Thus, there are 600 BSP trees for learning and another six hundred trees for validation. In the adaptive processing of data structures, a 7–10-7 neural network was used for our scene classification problem. The initialization is performed by setting the parameters ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$) of the network ran-

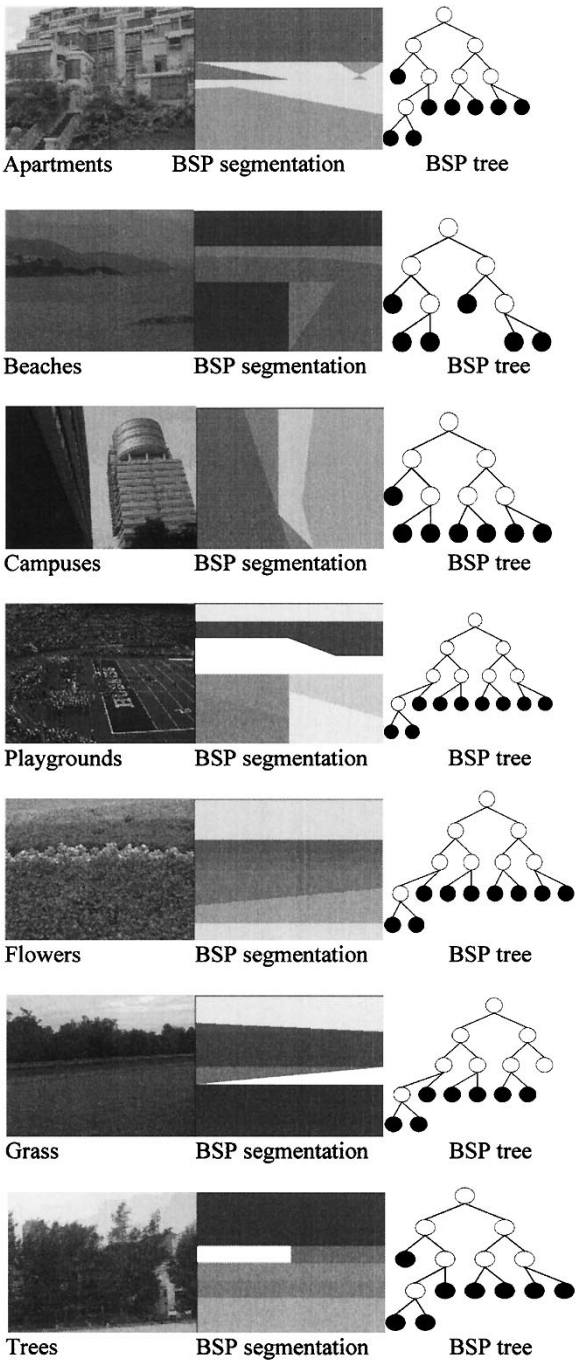


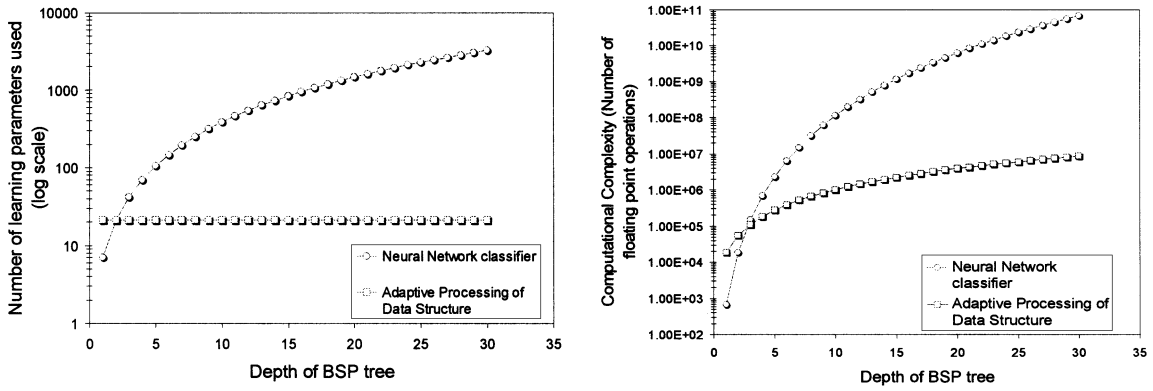
Fig. 7. Samples of natural scene images and their corresponding partitioned images and BSP trees. From top to bottom: apartments, beaches, campuses, playgrounds, flowers, grass, and trees. From left to right: Original images, the BSP segmented regions and the BSP trees. A dark node denotes a frontier (leaf) node of the trees.

domly in the range of -1 to 1.0 with a uniform distribution. The performance of our proposed algorithm for learning BSP tree structures is compared with the conventional neural-network classifier (feedforward type) using the backpropagation learning algorithm with a flat vector input converted from the tree representation, and the BPTS type algorithms.

In our investigation, a flat vector was constructed from each tree structure pattern by arranging the attributes of all nodes in the tree in sequence in the left-to-right and top-to-bottom order. A single-hidden-layer feedforward neural network was trained for scene classification using flat-vector patterns. The sequential mapping from the tree structure to the flat vector is, however, necessary to break some regularities inherently associated with the data structures, which will yield a poor generalization. This is the main motivation for developing an improved algorithm for the adaptive processing of data structures, which is able to classify structural information and to perform automatic inferring or learning for structural representations. Experimental results show that the classification performance of our proposed approach based on the adaptive processing of BSP tree structures outperforms the neural network classifier with the sequential representation. Moreover, since the number of nodes grows exponentially with the depth of the tree, a large number of parameters need to be learned for the sequential mapping by the neural-network classifier, which makes learning difficult and inefficient. Fig. 8(a) and (b) illustrates, respectively, the exponential increases in the number of learning parameters as well as an extremely high computational complexity (i.e., the number of floating point operations) for the neural-network classifier with sequential representation. On the contrary, a constant number of learning parameters and less floating-point operations (slightly more floating-point operations for a shallow tree) are required for the adaptive processing of the BSP structural representation. On the other hand, the scene classification performance of our proposed algorithm compares favorably with the general BPTS algorithms and the neural-network classifier. Encouraged results were obtained by the improved algorithm because the long-term dependency problems encountered by the BSP tree structural processing could be eliminated. To test whether the heuristic we added in our algorithm played a vital role, our proposed heuristic strategy is also integrated with the BPTS learning algorithm in this BSP scene classification. The resulting algorithm is called BPTSw-HER. Their performances are compared as shown in Fig. 9. It is noted that there is no significant improvement in the BPTS approach if only the heuristic strategy is applied. Overall classification accuracy rate is over 85% obtained by our proposed algorithm, whereas about 70% accuracy was obtained by BPTS type algorithms on the adaptive processing of BSP tree structures. Compared with the BPTS type algorithms and the neural-network classifier, our proposed algorithm exhibits consistently better performance over different image categories (especially for the categories of Flowers and Trees).

VI. Conclusion

In this paper, we present an improved algorithm for overcoming the problems of long-term dependency and slow convergence speed encountered in the BPTS-type algorithms in the adaptive processing of data structures. The rate of convergence is spectacular in our proposed algorithm because of the characteristic of fast convergence of the least squares method that we adopted for parameter optimization. Since the free learning parameters of the node representation in the tree are optimized in the layer-by-layer fashion, the total gradient information is not affected by the number of levels in tree structures. As a



(a) (b)

Fig. 8. Comparison of the scene classification performance between the sequential and structural representations. (a) Number of learning parameters used versus the depth of the BSP tree. (b) Computational complexity in the number of floating-point operations versus the depth of the BSP tree.

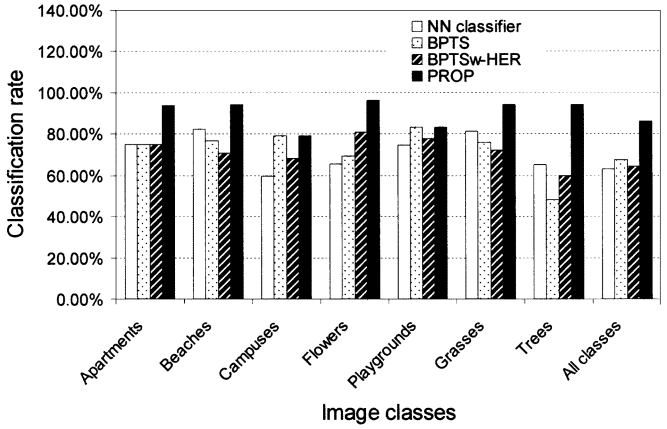


Fig. 9. Comparison of the scene classification performance by different approaches for seven categories of scene images.

result, the effect of gradient vanishing is insignificant so as to avoid the long-term dependency problem. We have also integrated a heuristic strategy in our algorithm to avoid the training stuck in local minima. Encouraging results are achieved by our proposed algorithm on three classification tasks we tested, including a synthesized policeman signaling problem, the parity checking of full binary tree structures, and scene classification from color images. Our improved algorithm outperforms BPTS type algorithms in the adaptive processing of data structures. Compared with a neural-network classifier with the flat vector input, our approach for processing data structures directly also shows much better results.

ACKNOWLEDGMENT

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

REFERENCES

[1] A. C. Tsoi, "Adaptive Processing of Data Structure: An Expository Overview and Comments," Faculty Informatics, Univ. Wollongong, Wollongong, Australia, 1998.

[2] ——, "Gradient based learning methods," in *Adaptive Processing of Sequences and Data Structures*, C. L. Giles and M. Gori, Eds. New York: Springer-Verlag, 1998, pp. 27–62.

[3] B. Hammer, *Learning with Recurrent Neural Networks*. New York: Springler-Verlag, 2000, vol. 254, Springer Lecture Notes in Control and Information Sciences.

[4] A. Sperduti and A. Starita, "Supervised neural networks for classification of structures," *IEEE Trans. Neural Networks*, vol. 8, pp. 714–735, May 1997.

[5] C. Goller and A. Kuchler, "Learning task-dependent distributed representations by back-propagation through structure," in *Proc. IEEE Int. Conf. Neural Networks*, 1996, pp. 347–352.

[6] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Trans. Neural Networks*, vol. 9, pp. 768–785, Sept. 1998.

[7] C. L. Giles and M. Gori, *Adaptive Processing of Sequences and Data Structures*. New York: Springer-Verlag, 1998.

[8] P. Fraconi, M. Gori, A. Kuchler, and A. Sperduti, "From sequences to data structures: Theory and applications," in *A Field Guild to Dynamical Recurrent Networks*, J. Kolen and S. Kremer, Eds. New York: IEEE Press, 2001, ch. 19, pp. 351–374.

[9] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press, 1986.

[10] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 76–86, Jan. 1992.

[11] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, pp. 157–166, Mar. 1994.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–80, 1997.

[13] R. Battiti and G. Tecchiolli, "Training neural nets with the reactive tabu search," *IEEE Trans. Neural Networks*, vol. 6, pp. 1185–1200, Sept. 1995.

[14] Y. Shang and B. W. Wah, "Global optimization for neural network training," *IEEE Computer*, vol. 29, pp. 45–54, Mar. 1996.

[15] S. Y. Cho and T. W. S. Chow, "A layer-by-layer least squares based recurrent networks: Stalling and escape," *Neural Processing Lett.*, vol. 7, no. 1, pp. 15–25, Feb. 1998.

[16] S. Y. Cho and T. W. S. Cho, "Training multilayer neural networks using fast global learning algorithm – Least squares and penalized optimization methods," *Neurocomput.*, vol. 25, pp. 115–131, 1999.

[17] Z. Wang, Z. Chi, D. Feng, and S. Y. Cho, "Adaptive processing of tree-structure image representation," in *Proc. 2nd IEEE Pacific-Rim Conf. Multimedia (PCM2001)*, Oct. 2001, pp. 989–995.

[18] S. Y. Cho, Z. Chi, and W. C. Siu, "Image classification with adaptive processing of BSP image representation," presented at the *ICSP2002*.

[19] B. Hammer and V. Sperschneider, "Neural networks can approximate mappings on structured objects," presented at the Proc. 2nd Int. Conf. Computational Intelligence Neuroscience, 1997.

[20] Y. Bengio and P. Frasconi, "Input-output HMMs for sequence processing," *IEEE Trans. Neural Networks*, vol. 7, pp. 1231–49, Sept. 1996.

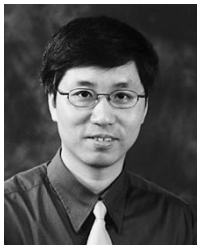
[21] X. Wu, "Image coding by adaptive tree-structured segmentation," *IEEE Trans. Inform. Theory*, vol. 38, pp. 1755–1767, Nov. 1992.

[22] H. Radha, M. Vetterli, and R. Leonardi, "Image compression using binary space partitioning tree," *IEEE Trans. Image Processing*, vol. 5, pp. 1610–1624, Dec. 1996.

[23] P. Salembier, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Trans. Image Processing*, vol. 9, pp. 561–576, Apr. 2000.

Siu-Yeung Cho (M'00) received the B.Eng. (Hons.) degree from the University of Brighton, Brighton, U.K., in 1994 and the Ph.D. degree from City University of Hong Kong, Hong Kong, in August 1999, both in electronic engineering.

From 1999 to 2001, he was a Senior Research Assistant and then Research Fellow with the Department of Electronic Engineering, City University of Hong Kong. From February 2001 to 2003, he was a Research Fellow with the Centre for Multimedia Signal Processing, Hong Kong Polytechnic University. In 2003, he joined the School of Computer Engineering, Nanyang Technological University of Singapore. His research interests include neural networks, pattern recognition, and 3-D computer vision. He has published more than 30 technical papers.



Zheru Chi (M'94) received the B.Eng. and M.Eng. degrees from Zhejiang University, Zhejiang, China, in 1982 and 1985, respectively, and the Ph.D. degree from the University of Sydney, Sydney, Australia, in March 1994, all in electrical engineering.

From 1985 to 1989, he was on the Faculty of the Department of Scientific Instruments at Zhejiang University. From April 1993 to January 1995, he was a Senior Research Assistant/Research Fellow in the Laboratory for Imaging Science and Engineering, University of Sydney. Since February 1995, he has been with the Hong Kong Polytechnic University, where he is now an Associate Professor in the Department of Electronic and Information Engineering. He was one of contributors to the *Comprehensive Dictionary of Electrical Engineering* (Boca Raton, FL; CRC Press and IEEE Press, 1999). His research interests include image processing, pattern recognition, neural networks, and fuzzy systems. He has co-authored one book and six book chapters, and published more than 100 technical papers.

Dr. Chi has served as Co-organizer of a Special Session/Session Chair/Area Moderator/Program Committee Member for a number of international conferences since 1997.

Wan-Chi Siu (S'77–M'77–SM'90) received the Associateship from The Hong Kong Polytechnic University, Hong Kong, (formerly called the Hong Kong Polytechnic), the M.Phil. degree from The Chinese University of Hong Kong, and the Ph.D. degree from Imperial College of Science, Technology, and Medicine, London, U.K., in 1975, 1977, and 1984, respectively.

From 1975 to 1980, he was with The Chinese University of Hong Kong. In 1980, he joined The Hong Kong Polytechnic University as a Lecturer and has been Chair Professor since 1992. He was Associate Dean of Engineering Faculty from 1992 to 1994 and Head of Department of Electronic and Information Engineering from 1994 to 2000. He has been Director of Centre for Multimedia Signal Processing and Dean of Engineering Faculty of the same university since September 1998 and September 2000, respectively. He has published more than 200 research papers, and his research interests include digital signal processing, fast computational algorithms, transforms, video coding, computational aspects of image processing and pattern recognition, and neural networks.

Dr. Siu is a Member of the Editorial Board of the *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, and an overseas member of the Editorial Board of the *IEE Review*. He was a Guest Editor of a Special Issue of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART II published in May 1998. He was also an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART II from 1995 to 1997. He was the General Chair or the Technical Program Chair of a number of international conferences. In particular, he was the General Chairman of the International Symposium on Neural Networks, Image and Speech Processing (ISSIPNN'94), and a Co-Chair of the Technical Program Committee of the IEEE International Symposium on Circuits and Systems (ISCAS'97), which were held in Hong Kong in April 1994 and June 1997, respectively. He is now the General Chair of the 2001 International Symposium on Intelligent Multimedia, Video & Speech Processing (ISIMVSP2001) and the 2003 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2003) which are to be held in Hong Kong. From 1991 to 1995, he was a Member of the Physical Sciences and Engineering Panel of the Research Grants Council (RGC), Hong Kong Government, and in 1994 he chaired the first Engineering and Information Technology Panel to assess the research quality of 19 Cost Centers (departments) from all universities in Hong Kong.

Ah Chung Tsoi (S'70–M'72–SM'90) was born in Hong Kong. He received the Higher Diploma in Electronic Engineering from Hong Kong Technical College, Hong Kong, in 1969, and the M.Sc. degree in electronic control engineering and Ph.D. degree in control engineering from University of Salford, Salford, U.K., in 1970 and 1972, respectively. He also received the B.D. degree from University of Otago, Otago, New Zealand, in 1980.

From 1972 to 1974, he was a Senior Research Fellow at the Inter-University Institute of Engineering Control, University College of North Wales, Bangor, U.K. From 1974 to 1977, he was a Lecturer at the Paisley College of Technology, Paisley, U.K. From 1977 to 1985, he was a Senior Lecturer at the University of Auckland, New Zealand. From 1985 to 1990, he was a Senior Lecturer at the University College, University of New South Wales, Australia. From 1990 to 1996, he was an Associate Professor, and then a Professor in Electrical Engineering at the University of Queensland, Australia. Since July 1996, he has been at the University of Wollongong, Wollongong, Australia, where he had been Dean, Faculty of Informatics from 1996 to 2000, and Director of Information Technology Services from 1999 to 2001. Since February 2001, he has been Pro Vice Chancellor (Information Technology) at the University of Wollongong. His research interests include aspects of neural networks and fuzzy systems and their application to practical problems, adaptive signal processing and adaptive control.

Dr. Thoi has been Chair of the ARC Expert Advisory Committee on Mathematics, Information and Communications Sciences since 2001. He received the Outstanding Alumni Award, Hong Kong Polytechnic University, in 2001.