Incremental Communication for Adaptive Resonance Theory Networks

Ming Chen, Ali A. Ghorbani, *Member, IEEE*, and Virendrakumar C. Bhavsar, *Member, IEEE*

***Abstract*—We have proposed earlier the incremental internode communication method to reduce the communication cost as well as the time of the learning process in artificial neural netwofrks (ANNs). In this paper, the limited precision incremental communication method is applied to a class of recurrent neural networks, the adaptive resonance theory 2 (ART2) networks. Simulation studies are carried out to examine the effects of the incremental communication method on the convergence behavior of ART2 networks. We have found that 7–13-b precision is sufficient to obtain almost the same results as those with full (32-b) precision conventional communication. A theoretical error analysis is also carried out to analyze the effects of the limited precision incremental communication. The simulation and analytical results show that the limited precision errors are bounded and do not seriously degrade the convergence of ART2 networks. Therefore, the incremental communication can be incorporated in parallel and special-purpose very large scale integration (VLSI) implementations of the ART2 networks.**

***Index Terms*—Adaptive resonance theory 2 (ART2) networks, artificial neural networks (ANNs), error analysis, finite precision computation, incremental communication.**

I. Introduction

THE COMMUNICATION complexity of artificial neural networks (ANNs) is directly proportional to the number of internode connections. To reduce the cost of interconnection as well as intercommunication, we have proposed the incremental internode communication method [10]. In the incremental internode communication, instead of communicating the full magnitude of a variable, only the increment or decrement to its previous value is sent on a communication link. For example, assume that node Y has to communicate the signal y to node Z at different time instants. Further, assume that $y(t)$ is the output of node Y at time t and $y(t+1)$ is its output at time $t+1$. In the conventional communication, the communication link will carry the value $y(t+1)$ at time $t+1$. In contrast, in the incremental communication method the communication link will carry the value $\Delta y(t+1)$, where $\Delta y(t+1) = y(t+1) - y(t)$. At the receiving end, the value $y(t+1)$ will be obtained by adding $\Delta y(t+1)$ to the previous value $y(t)$ stored at node Z. The incremental value Δy can be represented in either fixed- or floating-point format. The fixed-point value may be represented in the integer or fractional form using a fewer number of bits (i.e., limited precision) than full-precision used for the signal y. In the floating-point representation, few significant bits of the mantissa and full value of the exponent are often used. When the incremental value Δy is limited to a smaller precision than the precision of y, we denote the incremental value by $\overline{\Delta} y$. In this case, at the receiving end we get $y^*_{(t+1)}$ which represents an approximated value of $y(t+1)$.

The proposed incremental communication method would result in reducing the interconnection and intercommunication costs by a factor $\omega = b/\overline{b}$, where b represents the number of bits used to represent the full precision value of a variable x and $\overline{b}$ represents number of bits used to represent the limited precision incremental value, $\overline{\Delta} x$. For hardware implementations of ANNs, only $\overline{b}$ lines would be required for interneuron communications as compared to b lines in conventional implementations. The incremental communication decreases the cost of interconnection (i.e., number of lines in the communication link) by an amount directly proportional to the magnitude of ω and the number of interconnections. If bit-serial communication is used with a single line between a pair of neurons, fewer steps would be required for the incremental communication. The bit-serial communication has been used in parallel computers based on transputers [20] and the connection machine [18]. Alteratively, if multiple neurons are mapped to each of the processors in a parallel computer, the message lengths for interprocessor communications could be reduced. Note that our discussion above assumes that all operations inside a node are carried out with full precision. However, as discussed in [10], the incremental communication can be incorporated along with limited precision and range strategies for interneuron connection weights suggested in the literature [7], [14], [16]. The reduction in the communication cost of the incremental communication is at the expense of additional memory and additional operations within neurons.

The effectiveness of the proposed communication scheme for multilayer feedforward network architectures has been examined in previous works [10], [11]. It has been shown, that for some problems even 4-b precision in fixed- and floating-point representations is sufficient for the network to converge. With 8–12 b-precisions, almost the same results are obtained as those with the conventional communication using 32-b precision [10]. The proposed method can lead to significant savings in the intercommunication cost for implementations of ANNs on parallel computers as well as the interconnection cost of very large scale integration (VLSI) realizations.

Although the incremental communication method has been found to be suitable for a class of supervised ANNs, namely, the multilayer perceptrons, it is not known whether the method

Manuscript received June 13, 2002; revised May 12, 2003. This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC) under Grants RGPIN0089 and RGPIN2277441.

The authors are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, E3B 5A3, Canada (e-mail: ghorbani@unb.ca; bhavsar@unb.ca).

Digital Object Identifier 10.1109/TNN.2004.839357

can be incorporated effectively into other supervised as well as unsupervised ANN paradigms.

Adaptive resonance theory (ART) networks [1], [2] are unsupervised ANNs that dynamically determine the number of clusters based upon a "vigilance" parameter. There have been some attempts to implement ART architectures on parallel machines and in VLSI [17]. Similar to multilayer perceptron, the number of interconnections and the number of intercommunications are directly proportional to the number of nodes in the comparison and recognition layers. Therefore, the reduction in the communication and interconnection costs is desirable. The limited precision incremental values cannot represent respective variables accurately and therefore may affect the performance (e.g., convergence quality and rate) of a learning algorithm, either positively or negatively. In some cases, the limited precision incremental communication may result in smaller output error since the representational errors may have positive as well as negative values therefore some of the representational errors may cancel each other. The main objective of this paper is to incorporate the limited precision incremental communication method in an ART network and examine its convergence behavior.

The paper is organized as follows. In the following section, the ART2 architecture is first briefly reviewed. Subsequently, modified node architectures using incremental communication are presented. The incremental communication method is incorporated into the ART2 learning algorithm and simulation results for a test problem are presented in Section III. A theoretical error analysis of the effects of the limited-precision incremental communication method on the behavior of the ART2 network is carried out in Section IV. Finally, conclusions are given in the last section.

II. ART2 Networks

The basic principles of ART are given by Grossberg [12], [13]. ART2 networks self-organize recognition categories in response to analog as well as binary input patterns [1], [2]. In the following subsections, we briefly review the basic concepts and notations of ART2 networks. Subsequently, the architectures of ART2 nodes incorporating incremental communication are outlined.

A. Basic Concepts

ART2 networks consists of three fields: a preprocessing field F_0, a layer of processing units called feature representation field F_1, and a layer of output units called category representation field F_2. F_1 and F_2 are fully connected in both directions via weighted connections called pathways. The set of pathways with corresponding weights is called an adaptive filter. The weights in the adaptive filter encode the long-term memory (LTM) traces. Patterns of activation of F_1 and F_2 nodes are called short-term memory (STM) traces. The connections leading from F_1 to F_2, and from F_2 to F_1 are called bottom-up and top-down adaptive filters, respectively. There are also corresponding bottom-up weights and top-down weights.

Fig. 1 illustrates the ART2 architecture used in this paper. F_0 field has the same structure as that of F_1 field and both of them have M nodes. The ith node in F_0 field only connects to the corresponding ith F_1 node in one direction, namely, from F_0 to F_1. The F_0 field preprocesses input patterns. Each input pattern can be represented as a vector I which consists of analog or binary components I_i ($i = 1, \ldots, M$). Since there is no significant advantage of using incremental communication for binary input patterns, in this paper we consider only analog input patterns for ART2 networks. The nodes of F_2 field are numbered as $J = M+1, \ldots, N$, that is, the F_2 field consists of $N-M$ nodes which are fully connected among themselves in both directions [1].

The fields F_1 and F_2, as well as the bottom-up and top-down adaptive filters constitute the attentional subsystem of ART2 networks. An auxiliary orienting subsystem becomes active when a bottom-up input to F_1 fails to match the learned top-down expectation (vector z_J is composed of $N - M$ components, z_{Ji} ($i = 1, \ldots, N - M$) readout by the active category representation at F_2). In this case, the orienting subsystem is activated and causes rapid reset of the active category representation in F_2. This reset event automatically induces the attentional subsystem to proceed with a parallel search. Alternative categories are tested until either an adequate match is found or a new category is established.

B. ART2 System Dynamics

We present here only those aspects of ART2 networks that will be referred throughout this paper. For an in-depth, discussion see [1]–[3].

1) Preprocessing Field F_1: Each sublayer of the F_0 and F_1 STM fields carries out two computations: the summation of intrafield and interfield inputs to the layer and the normalization of the resulting activity vector. Equations (1)–(6) characterize the STM activities, p_i, q_i, u_i, v_i, w_i, and x_i, computed at the F_1 field

$$p_i = \begin{cases} u_i, & \text{if } F_2 \text{ is inactive} \\ u_i + g(y_J)z_{Ji}, & \text{if } F_2 \text{ is active} \end{cases} \tag{1}$$

$$q_i = \frac{p_i}{e + \|\mathbf{p}\|} \tag{2}$$

$$v_i = f(x_i) + bf(q_i) \tag{3}$$

$$u_i = \frac{v_i}{e + \|\mathbf{v}\|} \tag{4}$$

$$w_i = Q_i + au_i \tag{5}$$

$$x_i = \frac{w_i}{e + \|\mathbf{w}\|} \tag{6}$$

where $\|\mathbf{p}\|$ denotes the L_2 norm of vector $\mathbf{p}$, the parameter e is set to 0 for simplicity and y_J is the STM activity of the Jth F_2 node. The nonlinear signal function f in (3) is typically of the form

$$f(x) = \begin{cases} 0, & \text{if } 0 \le x < \theta \\ x, & \text{if } x \ge \theta. \end{cases} \tag{7}$$

2) Input Representation Field F_0: At the F_0 field, STM activities Q_i, U_i, V_i, and X_i, are the same as the corresponding STM activities at the F_1 field. The other two STM activities, P_i and W_i, are defined as

$$P_i = U_i \tag{8}$$

$$W_i = I_i + au_i. \tag{9}$$

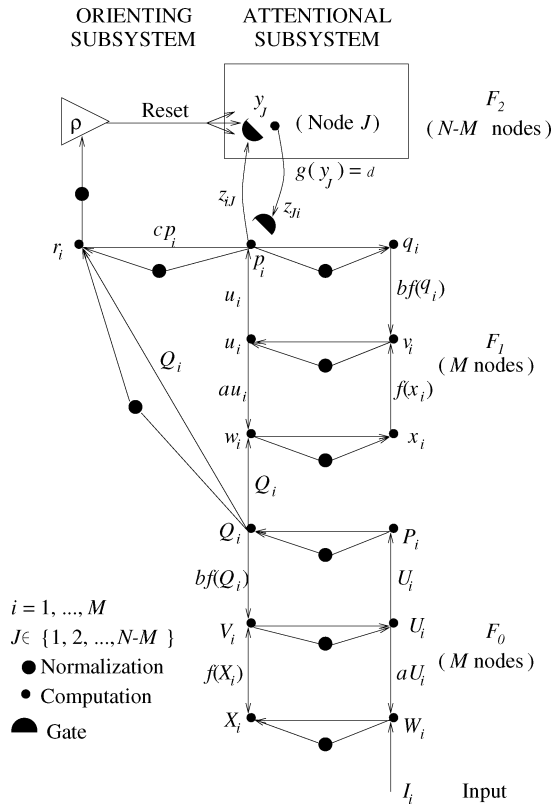


Fig. 1. ART2 architecture used in the simulator (adapted from [5]).

3) Category Representation Field F_2: One of the key properties of F_2 is the reset of active F_2 nodes that is carried out by using a gated dipole field. The main elements of the dipole field dynamics may be characterized as (10), shown at the bottom of the page. If F_2 is inactive, all $g(y_j) = 0$. An active F_2 competitive field is said to be designed to make a choice if only one node ($j = J$) receives the largest total input from F_1. In this case, $g(y_J)$ equals a constant d.

4) ART2 LTM Equations: When F_2 makes a choice, if the Jth F_2 node is active and is the winner node then the top-down and bottom-up LTM trace equations for ART2 are given by

$$\frac{d}{dt}z_{Ji} = d(p_i - z_{Ji}) = d(1-d)\left(\frac{u_i}{1-d} - z_{Ji}\right) \tag{11}$$

$$\frac{d}{dt}z_{iJ} = d(p_i - z_{iJ}) = d(1-d)\left(\frac{u_i}{1-d} - z_{iJ}\right) \tag{12}$$

where $0 < d < 1$. Note that d on the right side of (11) and (12) is a network parameter, and d on the left side represents the differential sign. These notations are the same as those given in [1] and [2]. For all $j \neq J$ we get $dz_{ji}/dt = 0$ and $dz_{ij}/dt = 0$.

We let the top-down initial LTM values satisfy

$$z_{ji}(0) = 0 \tag{13}$$

for $i = 1, \ldots, M$ and $j = M+1, \ldots, N$. This condition ensures that no reset occurs when an uncommitted F_2 node first becomes active and therefore learning can begin.

5) Match and Reset: The vector $\mathbf{r}$ monitors the degree of match between the F_1 bottom-up input $\mathbf{Q}$ and the top-down input dz_J. System reset occurs iff

$$\|\mathbf{r}\| < \rho \tag{14}$$

where ρ is a dimensionless vigilance parameter between 0 and 1. Vector $\mathbf{r}$ obeys

$$r_i = \frac{Q_i + cp_i}{e + \|\mathbf{Q}\| + \|c\mathbf{p}\|} \tag{15}$$

where $c > 0$ and $e = 0$. The norm of the vector r is given by

$$\|\mathbf{r}\| = \frac{[1 + 2c\|\mathbf{p}\|\cos(\mathbf{Q},\mathbf{p}) + c^2\|\mathbf{p}\|^2]^{1/2}}{1 + c\|\mathbf{p}\|} \tag{16}$$

where $\cos(\mathbf{Q},\mathbf{p})$ denotes the cosine of the angle between the vector $\mathbf{Q}$ and the vector $\mathbf{p}$.

C. Node Architectures Using Incremental Communication

Since fields F_1 and F_2 are fully connected in both directions, the ART2 network involves large amount of internode communication. Therefore, we propose the incremental communication method for the internode communication between the F_1 and F_2 fields. In the incremental communication method, all operations inside a node are carried out using full precision. We have incorporated the incremental communication method into the ART2 learning algorithm and modified the architecture of the F_0, F_1, and F_2 nodes using incremental communication. Figs. 2 and 3 illustrate the F_1 node and the F_2 node architectures, respectively, using the incremental communication method. In Fig. 2, the incremental value $\overline{\Delta}z_{ji}(t+1)$ is received from an F_2 node given in Fig. 3 and the incremental value $\overline{\Delta}u_i(t+1)$ is sent to other F_1 nodes. The function $\Phi(.)$ is used to truncate the values of $\Delta u_i(t+1)$ and $\Delta z_{Ji}(t+1)$ to limited precision values $\overline{\Delta}u_i(t+1)$ and $\overline{\Delta}z_{Ji}(t+1)$, respectively. The α and β icons represent the points of connections to operators within the F_1 and F_2 nodes.

III. Simulation Studies

We have incorporated the incremental communication method into the learning algorithm in order to investigate its effects on the convergence behavior of ART2 networks. We have implemented an ART2 simulator to investigate the effects of the limited precision fixed- and floating-point incremental values on the representational and convergence abilities of the network. Note that incremental communication requires more

$$g(y_J) = \begin{cases} d, & \text{if } T_J = \max\{T_j\text{: the } j\text{th } F_2 \text{ node has not been reset on the current trial}\} \\ 0, & \text{otherwise} \end{cases}. \tag{10}$$

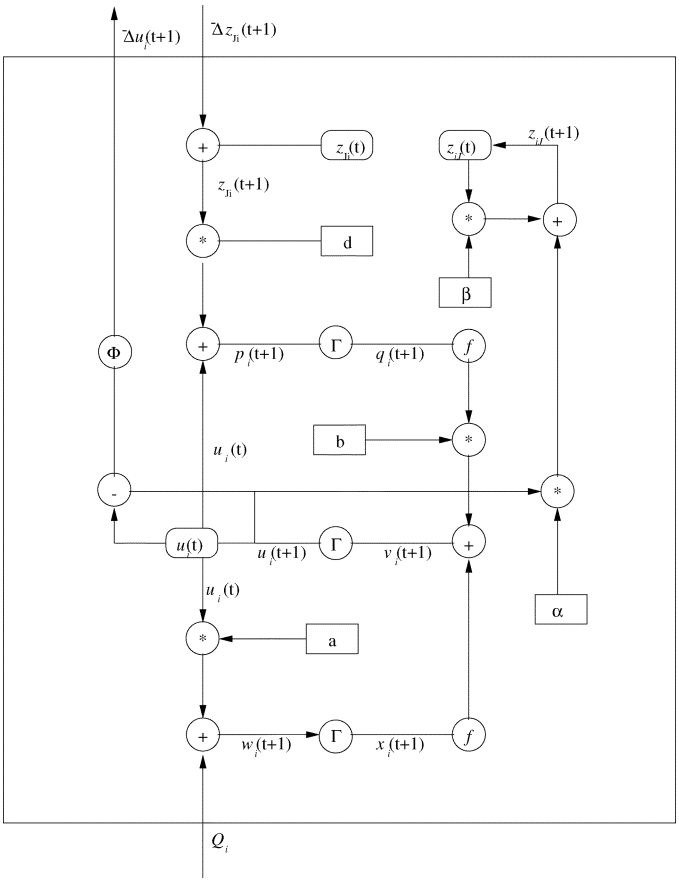


Fig. 2. F_1 node architecture after the learning begins.

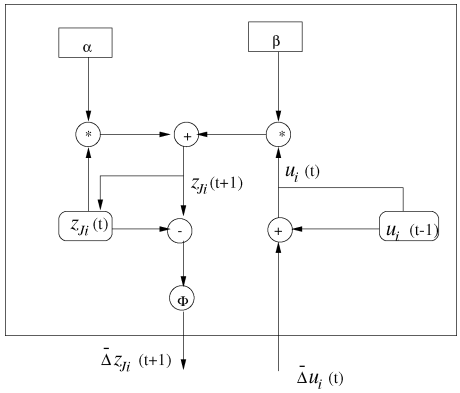


Fig. 3. F_2 node architecture after the learning begins.

operations in a node to generate incremental values to be sent to other nodes and at the receiving end these incremental values are used to update the values of the corresponding variables. This implies that the execution time per iteration will increase. However, in parallel implementation the communication cost will be reduced by using incremental communication method. In our previous work, we have found that the reduction in the communication cost results into overall speedup (for details see [10]).

In our simulation studies, we compare the performance of the conventional (standard) and incremental communication methods using the number of iterations for convergence as a metric (as commonly used by many researchers) rather than the execution time on a particular computer. It is well known that measuring the actual simulation time may not be a good indication of the performance due to many factors (for example, CPU architecture, cache, and main memory characteristics).

This simulation is based on the simulation program given in [9]. The ensuing subsections describe the simulator characteristics, the test problem chosen for the simulation studies, and various simulation parameters. Subsequently, we present simulation results for two sets of experiments.

A. ART2 Simulator

The learning rule used in the simulator is the slow learning rule, meaning that the weights are updated by integrating the differential (11) and (12). The conventional (standard) as well as the incremental communication methods are implemented. In the conventional communication, all the parameters are represented in full precision (32 b), whereas in the incremental communication method all incremental values are represented with reduced precision. The network parameters as well as the convergence criterion value can be set by the user. The convergence criterion value is chosen so that all the input patterns are learned.

When a limited precision is used to represent the argument x in (7), it is possible that the function f will lose its smoothness and become piecewise linear. It is anticipated that this will affect the network convergence.

The incremental values can be represented using fixed- and floating-point representations. With the fixed-point representation, the position of the binary point is determined by the user based on the problem chosen. In the floating point representation, few significant bits of mantissa and full 8-b value of the exponent is used.

The simulation is primarily based on the standard ART2 algorithm. The input from the preprocessing field F_0 to the field F_1 is fed through three sublayers and is modified at each step. These steps perform normalization, contrast enhancement and noise suppression of the input before it is fed to the output field F_2 for competition. When an input from F_0 is filtered through the feedforward connections, competitive learning takes place in the F_2 nodes. A winner is decided by comparing the activations produced among all the F_2 nodes. The best matching vector is now passed back to F_1. The reset rule defined in (14) is applied. If no reset occurs, the top-down and bottom-up weights are modified according to (11) and (12). Otherwise, the best matching node is disabled, the system is reset, and the whole process is repeated. If all nodes in F_2 become disabled a new node is created.

The simulation experiments consist of training an ART2 network using the conventional and the incremental communication schemes while varying the precision of the incremental values of selected F_1 STM variables (namely, p_i and u_i, $i = 1, \ldots, M$) and top-down as well as bottom-up LTM variables.

B. Test Problem and Simulation Setup

The test problem chosen for simulation is composed of 50 analog input patterns which are designed to be similar to the 50 patterns given in [2, Fig. 1]. For details of these patterns

please refer to [8, App. 1] and http://www.cs.unb.ca/profs/ghorbani/art-data.html. Each input pattern is considered as a vector $\mathbf{I} = (I_1, I_2, \ldots, I_M)$, where $M = 25$.

The learning goal is to cluster the given input patterns into recognition categories. The patterns that are grouped in the same category usually share some common features. The number of categories obtained may change with different network parameters. We use two sets of network parameters for simulation experiments. Two sets of experiments (Experiments 1 and 2) are carried out. The values of all system parameters for the two sets of experiments are chosen to be equal, except for the threshold (θ) and vigilance (ρ) parameters. For each set of experiments, we use the same network parameters and vary the type (fixed or floating-point) as well as the precision of the incremental values; however, these two sets of experiments use different network parameters. The absolute average discrepancy between the top-down expectations (i.e., the final top-down vectors for the established category after the network converges) of the conventional and the incremental communication is referred to as the *error* in the following subsections.

For the ART2 network using the conventional communication method, we have selected the architectures and parameters that give good performance. To have comparable results, the same parameters and architectures are also used with the incremental communication method. For the fixed-point representation we use 2 b to the left and k b to the right of binary point (i.e., a total of $k + 2$ b are used). For the floating-point representation we use 8 b for the exponent and k b for the mantissa (i.e., a total of $k + 8$ b are used). In our experiments, the value of k is varied from 2 to 20 in the step of one bit at a time.

For our experiments, we chose some of the parameters as follows: $a = b = 5$, $d = 0.8$, and $c = 0.22$. For Experiment 1, the vigilance parameter ρ is selected to be the same as given in [2], $\rho = 0.95$. Note that in all graphs presented in this paper the precision represents the value k.

C. Experiment 1

For this experiment we set the threshold θ to $= 0.23$. The training is considered complete when the category structure established on one complete presentation of the 50 inputs remains stable thereafter. When the network converges, the 50 input patterns are classified into 14 recognition categories for both conventional and incremental communication methods. It would not be surprising if in some cases one obtains different number of recognition categories.

Fig. 4 represents the error as a function of the precision of fixed- and floating-point representations. Initially, as the precision of incremental values increases, the error decreases quickly. The error with the floating-point representation is consistently lower than that with the fixed-point representation. This is expected since the floating-point representation allows a large dynamic range of values.

Fig. 5 depicts the total number of iterations required by the network to converge using conventional and incremental communications with various precisions. It is seen that the number of iterations required for the floating-point representation is always close to that for the conventional communication. When more than 8 b are used for the fixed-point representation, the number of iterations is always close to, and sometimes even smaller than, that for the conventional (standard) communication.

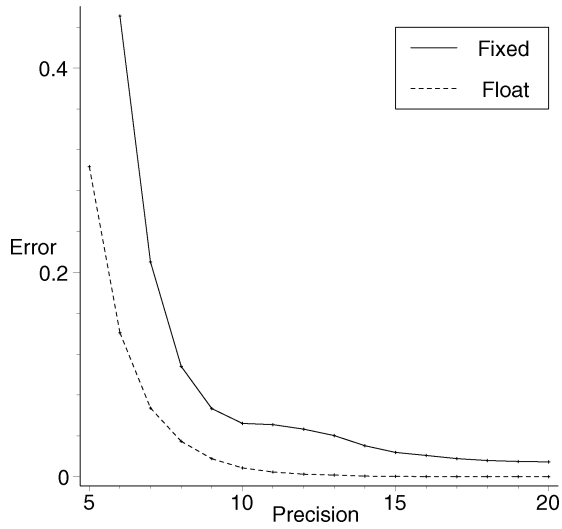


Fig. 4. Error in top-down expectations.

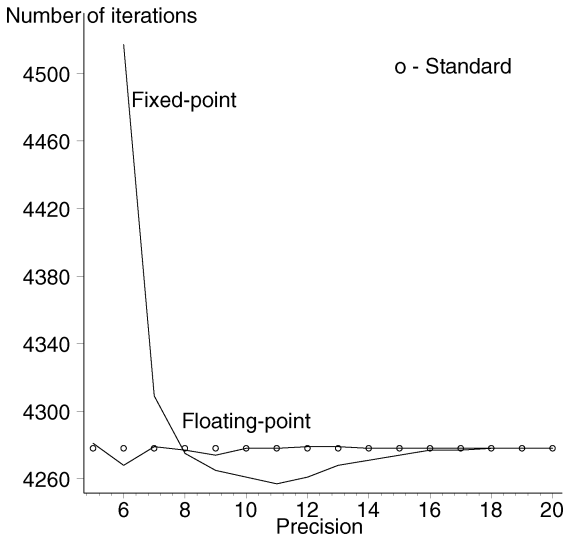


Fig. 5. Total number of iterations.

The increase/decrease in the computation and communication costs for selected precisions are given in Table I. In the table, the variable $\mathrm{t_{comm}}$ represents the increase/decrease in the communication time. The variables $\mathrm{t_{comp1}}$, and $\mathrm{t_{comp2}}$ represent the increase/decrease in the computation time for Experiments 1 and 2, respectively. Note that a negative value indicates the decrease, and a positive value denotes the increase in time. It is seen that the use of incremental communication results in substantial savings in the communication time with very small, if at all, increase in the computation time. For the floating-point representation, 5-b mantissa is found to be enough to obtain the same recognition categories as those with the conventional communication method. For the fixed-point representation the precision of $k = 6$ b is found to be adequate to obtain the same recognition categories as those with the conventional communication.

TABLE I
PERCENTAGE INCREASE/DECREASE IN THE COMPUTATION AND COMMUNICATION TIMES FOR EXPERIMENTS 1 AND 2

Precision	Fixed-point			Floating point		
k	t_{comm}	t_{comp1}	t_{comp2}	t_{comm}	t_{comp1}	t_{comp2}
10	-62.5	-0.39	-1.2	-43.7	—	-0.03
8	-68.7	-0.07	-1.2	-50.0	-0.02	—
6	-75.0	+5.59	+0.8	-56.2	-0.23	-0.29
5	-78.1	+3.81	+4.4	-59.3	+0.07	-0.09

t_{comp1} and t_{comp2} represent the computation time for Experiments 1 and 2, respectively.

t_{comp1} and t_{comp2} represent the computation time for Experiments 1 and 2, respectively.

D. Experiment 2

For this experiment, we chose $\theta = 0.21$ which is slightly below that for Experiment 1 and $\rho = 0.98$ which is a little higher than that for Experiment 1. In this case, the input patterns get classified into 21 categories compared to 14 categories for Experiment 1. This is mainly due to the higher value of vigilance (ρ). For both representations $k = 5$ gives the same recognition categories as those with the conventional communication.

Figs. 6 and 7 show the error and the total number of iterations required by the network to converge using the conventional and the incremental communications with various precisions. Table I gives increase/decrease in the computation and communication times for the two number representations.

IV. ERROR ANALYSIS

The incremental communication is implemented by limiting the precision used for the values that are communicated between various nodes. Limited precision reduces the accuracy and may affect the performance of an ANN learning algorithm. Note that in the incremental communication method all operations inside a node are carried out using full precision. In this section, we investigate the effects of limited precision incremental communication method on the convergence behavior and performance of ART2 networks.

First, various possible sources for error are briefly discussed. Subsequently, we consider the ART2 dynamics and analyze error generation and propagation during the training of ART2 networks. Two similarity measures are introduced and the changes in weights during the learning phase are compared between the conventional and the incremental communication schemes.

A. Sources of Errors

The limited precision error in computations ε falls into two main categories: the generated error and the propagated error [11], [19]. The generated error depends on the methods (e.g., truncation and roundoff) used to obtain limited precision incremental values. The propagated error is the error in the value of a function due to the error in its arguments. The propagated error increases with the number of operations. If the value of a variable x is truncated to obtain a reduced precision variable $\overline{x}$, the generated error is $\varepsilon = \overline{x} - x$. If the precision used for the incremental values is k b, the magnitude

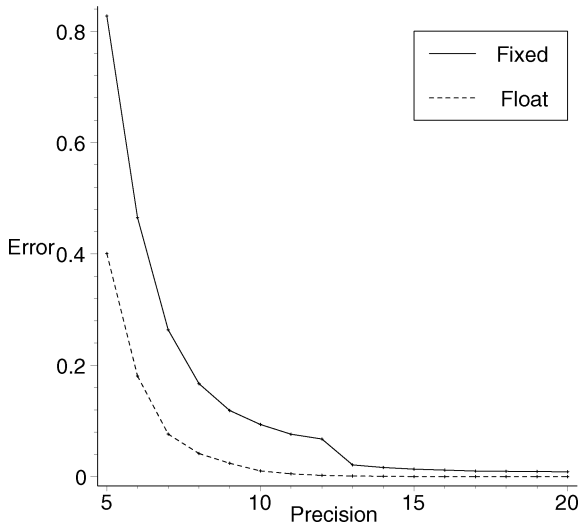


Fig. 6. Error in top-down expectations.

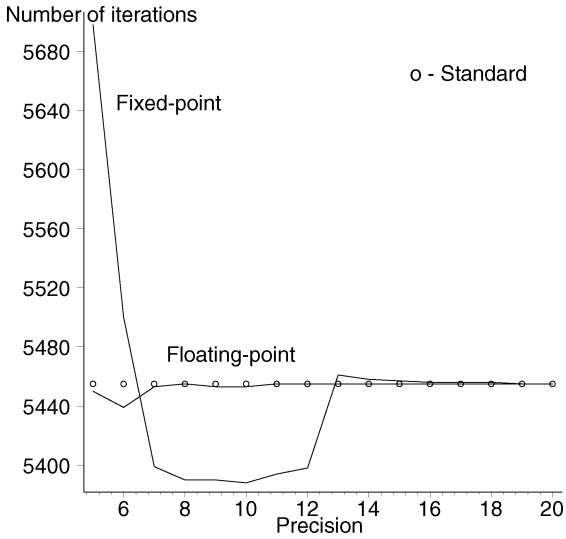


Fig. 7. Total number of iterations.

of the maximum generated error is equal to 2^{-k}. In the ensuing analysis, we consider the limited precision error to be a discrete random variable distributed over a range determined by the number of truncated bits (digits).

We assume that the limited precision errors have the following properties:

1) $\Phi(.)$ is a stationary random limited precision process;
2) limited precision errors are independent of each other;
3) limited precision errors are uncorrelated with the inputs/outputs;
4) limited precision errors are uniformly distributed.

Suppose that each component w_i ($i = 1, \ldots, M$) of a vector $\mathbf{w}$ has an error ε_{w_i} and these components are used to compute the function $x_i = w_i/\|\mathbf{w}\|$, where $\|\mathbf{w}\|$ is the L_2 norm of $\mathbf{w}$. Assuming that the errors are independent and random, then the error in x_i is given as [19, pp. 13]

$$\varepsilon_{x_i} = \sqrt{\left(\frac{\partial x_i}{\partial w_1}\varepsilon_{w_1}\right)^2 + \cdots \left(\frac{\partial x_i}{\partial w_M}\varepsilon_{w_M}\right)^2}. \qquad (17)$$

We denote

$$\varepsilon_w = \max\{\varepsilon_{w_1}, \varepsilon_{w_2}, \ldots, \varepsilon_{w_M}\}.$$

Lemma 1: The error in x_i is

$$\varepsilon_{x_i} \leq \frac{\varepsilon_w}{||\mathbf{w}||}. \tag{18}$$

Proof: The partial derivatives in (17) are given as

$$\frac{\partial x_i}{\partial w_i} = \frac{||\mathbf{w}||^2 - {w_i}^2}{||\mathbf{w}||^3} \text{ and}$$
$$\frac{\partial x_i}{\partial w_j} = \frac{-w_i w_j}{||\mathbf{w}||^3}.$$

Therefore

$$\varepsilon_{x_i} \leq \varepsilon_w \sqrt{\left(\frac{||\mathbf{w}||^2 - {w_i}^2}{||\mathbf{w}||^3}\right)^2 + \sum_{j=1, j\neq i}^{M} \left(\frac{|-w_j w_i|}{||\mathbf{w}||^3}\right)^2}.$$

Simplifying we get

$$\varepsilon_{x_i} \leq \frac{\varepsilon_w}{||\mathbf{w}||^2} \sqrt{||\mathbf{w}||^2 - {w_i}^2}.$$

If $||\mathbf{w}||^2 \gg {w_i}^2$ then ${w_i}^2$ can be omitted. In this case, we obtain the upper bound of ϵ_{x_i} as

$$\varepsilon_{x_i} \leq \frac{\varepsilon_w}{||\mathbf{w}||}.$$

□

From the above lemma, it is seen that the error propagated by the normalization function $x_i = w_i/||\mathbf{w}||$ is always less than or equal to the normalized maximum of its input errors.

B. Analysis of ART2 System Dynamics

The ART2 module considered here includes the principal components of all ART modules, namely, an attentional subsystem, which contains an input representation field F_1 and a category representation field F_2, and an orienting subsystem which interacts with the attentional subsystem to carry out an internally controlled search process. The two fields are linked by both a bottom-up $F_1 \rightarrow F_2$ adaptive filter and top-down $F_2 \rightarrow F_1$ adaptive filter. A path from the ith F_1 node to the jth F_2 node contains an LTM trace, or an adaptive bottom-up weight, z_{ij}, a path from the jth F_2 node to the ith F_1 node that contains a top-down weight z_{ji}. These weights multiply path signals between fields.

Table II summarizes F_0 STM activities for the first three iterations based on (1)–(9). It is seen that after the second iteration all STM activity values do not change, which implies that all F_0 nodes are converged. This allows normalized $\mathbf{I}$ (i.e., vector $\mathbf{Q}$) to be used as an input to the orienting subsystem. As a result, $\mathbf{Q}$ does not change when F_2 becomes active and F_0 provides stable inputs to the orienting subsystem throughout the trial.

All F_1 STM nodes remain proportional to $\mathbf{Q}$ so long as F_2 remains inactive. Vectors $\mathbf{w}$ and $\mathbf{v}$ are amplified by intrafield feedback, and $\mathbf{u} = \mathbf{q} = \mathbf{p} = \mathbf{x} = \mathbf{Q}$. Thus, F_1 has invariance property when F_2 is inactive. The STM activity values in the first two iterations are summarized in Table III. Subsequently, an uncommitted F_2 node becomes active (i.e., there are nonzero top-down weights from F_2 to F_1). The LTM trace z_{Ji} can be obtained by solving the (11) as

$$z_{Ji} = \frac{1}{1-d} u_i (1 - e^{-d(1-d)t}) + A e^{-d(1-d)t} \tag{19}$$

where A is the initial value of z_{Ji} and t represents the discrete time step. In the simulator, A is set to zero when a new category is established. All STM activity values of the F_1 field, except p_i, do not change during the successive iterations. Based on (1) and (19) p_i is given as

$$p_i = u_i + d z_{Ji} = \frac{1}{1-d} Q_i (1 - d e^{-d(1-d)t}). \tag{20}$$

In the simulator slow learning rule is used and the weights are updated by integrating the differential (11) and (12). The differential equations for the weights are solved using a fourth-order Runge–Kutta solver with a fixed step size ($h = 0.1$) and network parameter $d = 0.8$ as is used for computational experiments. This solver is described as the following:

$$\Delta 1 = hd(1-d)\left(\frac{u_i^{(n-1)}}{1-d} - z_{Ji}^{(n-1)}\right)$$
$$\Delta 2 = hd(1-d)\left(\frac{u_i^{(n-1)}}{1-d} - \frac{\Delta 1}{2} - z_{Ji}^{(n-1)}\right)$$
$$\Delta 3 = hd(1-d)\left(\frac{u_i^{(n-1)}}{1-d} - \frac{\Delta 2}{2} - z_{Ji}^{(n-1)}\right)$$
$$\Delta 4 = hd(1-d)\left(\frac{u_i^{(n-1)}}{1-d} - \Delta 3 - z_{Ji}^{(n-1)}\right)$$
$$z_{Ji}^{(n)} = z_{Ji}^{(n-1)} + \frac{\Delta 1 + 2\Delta 2 + 2\Delta 3 + \Delta 4}{6}. \tag{21}$$

In this case, z_{Ji} at time n is given as

$$z_{Ji}{}^{(n)} = \alpha z_{Ji}{}^{(n-1)} + \beta u_i{}^{(n-1)} \tag{22}$$

where $\alpha = 0.98412732$ and $\beta = 0.07936339971$.

C. Analysis of ART2 Training

The incremental communication method is used with the F_1 and F_2 fields. To compute the top-down weights STM activity values of F_1 nodes are needed for computations in F_2 nodes and

TABLE II
F_0 STM ACTIVITIES IN ITERATIONS 1, 2, AND 3

Iterations	W_i	X_i	P_i	Q_i	V_i	U_i
Iteration 1	I_i	$I_i/\|\|\mathbf{I}\|\|$	0	0	$I_i/\|\|\mathbf{I}\|\|$	$I_i/\|\|\mathbf{I}\|\|$
Iteration 2	$I_i(1+a/\|\|\mathbf{I}\|\|)$	$I_i/\|\|\mathbf{I}\|\|$	$I_i/\|\|\mathbf{I}\|\|$	$I_i/\|\|\mathbf{I}\|\|$	$(1+b)I_i/\|\|\mathbf{I}\|\|$	$I_i/\|\|\mathbf{I}\|\|$
Iteration 3	$I_i(1+a/\|\|\mathbf{I}\|\|)$	$I_i/\|\|\mathbf{I}\|\|$	$I_i/\|\|\mathbf{I}\|\|$	$I_i/\|\|\mathbf{I}\|\|$	$(1+b)I_i/\|\|\mathbf{I}\|\|$	$I_i/\|\|\mathbf{I}\|\|$

TABLE III
F_1 STM ACTIVITIES IN ITERATIONS 1 AND 2

Iterations	w_i	x_i	p_i	q_i	v_i	u_i
Iteration 1	Q_i	Q_i	0	0	Q_i	Q_i
Iteration 2	$(1+a)Q_i$	Q_i	Q_i	Q_i	$(1+b)Q_i$	Q_i

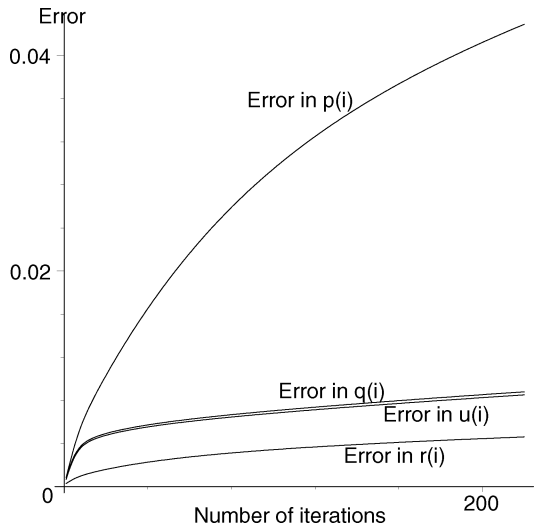


Fig. 8. Analytical errors in p_i, q_i, u_i, and r_i versus the number of iterations for 10-b precision.

these top-down weights are in turn required to compute STM activity values of F_1 nodes. In the training phase, errors due to the incremental communication get propagated through iterations. In the following subsection, error analysis of the first two steps is given. We assume that limited precision fixed-point incremental values are used. Subsequently, we analyze errors for a general case.

1) Error Analysis of the First Two Iterations: We consider the case of establishing a new category. To update z_{Ji} value, $\overline{\Delta}u_i$ will be sent from F_1 to F_2 [see (22)]. According to the Runge–Kutta solver when it is the first time to update z_{Ji}, the total error propagated to the Jth node of F_2 is

$$\varepsilon_{\Delta u_i} = \Phi(\Delta u_i) - \Delta u_i = \overline{\Delta} u_i - \Delta u_i$$

where $\Phi(.)$ represents the function that converts Δu_i to the limited precision fixed-point value $\overline{\Delta}u_i$. For k-bit fixed-point representation we obtain

$$\varepsilon_{\Delta u_i} \in (-2^{-k}, 0].$$

Let ${z^*_{Ji}}^{(1)}$ represent the approximated value of $z^{(1)}_{Ji}$ at the first iteration. Based on (22), the error in ${z^*_{Ji}}^{(1)}$ is given as

$$\varepsilon^{(1)}_{z^*_{Ji}} \le \beta \varepsilon_{\Delta u_i}.$$

Note that $z^*_{Ji} = z_{Ji} + \varepsilon^{(1)}_{z^*_{Ji}}$.

To compute the F_1 STM activity value represented by p_i ($i = 1, \ldots, M$), z^*_{Ji} needs to be transmitted from F_2 to F_1. The reduced precision value of z^*_{Ji}, denoted by $\overline{z^*_{Ji}}$, will be sent to an F_1 node. Therefore

$$\overline{z^*_{Ji}} = \Phi(z^*_{Ji}) = z_{Ji} + \varepsilon^{(1)}_{z^*_{Ji}} + \varepsilon_{\Delta z^*_{Ji}}$$

where the generated error

$$\varepsilon_{\Delta z^*_{Ji}} \in (-2^{-k}, 0].$$

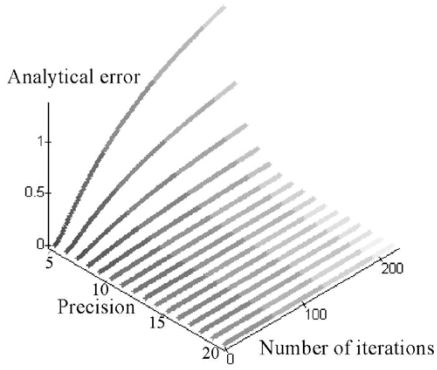


Fig. 9. Analytical errors in top-down weights as a function of the function of the number of iterations and precision.

The total error in a top-down weight is $\varepsilon^{(1)}_{z^*_{Ji}} + \varepsilon_{\Delta z^*_{Ji}}$. Since there is no error in u_i for the first iteration, based on (1) and (10), the error in p_i is

$$\varepsilon^{(1)}_{p^*_i} \le d(\varepsilon^{(1)}_{z^*_{Ji}} + \varepsilon_{\Delta z^*_{Ji}}).$$

According to (2) and Lemma 1, the propagated error in q_i is

$$\varepsilon^{(1)}_{q^*_i} \le \frac{\varepsilon^{(1)}_{p^*_i}}{\|\mathbf{p}^{(1)}\|}.$$

Since for the first iteration there is no error in x_i, by (3) the error in v_i is given as

$$\varepsilon^{(1)}_{v^*_i} \le b\varepsilon^{(1)}_{q^*_i}.$$

Similar to (4) and (15), the propagated errors in u_i and r_i are

$$\varepsilon^{(1)}_{u^*_i} \le \frac{\varepsilon^{(1)}_{v^*_i}}{\|\mathbf{v}\|} \text{ and}$$
$$\varepsilon^{(1)}_{r^*_i} \le \frac{2c}{1 + c\|\mathbf{p}^{(1)}\|}[\varepsilon^{(1)}_{u^*_i} + d(\varepsilon^{(1)}_{z^*_{Ji}} + 2^{-k})].$$

In the second iteration, at the time of updating z_{Ji}, note that both $u^{(1)}_i$ and $z^{(1)}_{Ji}$ have been contaminated by errors. Therefore, based on (22) we have

$$\varepsilon^{(2)}_{z^*_{Ji}} \le \alpha\varepsilon^{(1)}_{z^*_{Ji}} + \beta\varepsilon^{(1)}_{u^*_i}.$$

According to the analysis of F_1 STM activities given in the previous subsection, we get $\|\mathbf{v}\| = 1 + b$ and $\|\mathbf{w}\| = 1 + a$. The propagated errors in $p^{(2)}_i$, $w^{(2)}_i$, $x^{(2)}_i$, and $v^{(2)}_i$ are also affected by $u^{(1)}_i$ in addition to other errors in the first iteration. Thus

$$\varepsilon^{(2)}_{p^*_i} \le \varepsilon^{(1)}_{u^*_i} + d(\varepsilon^{(2)}_{z^*_{Ji}} + 2^{-k})$$
$$\varepsilon^{(2)}_{w^*_i} \le a\varepsilon^{(1)}_{u^*_i}$$
$$\varepsilon^{(2)}_{x^*_i} \le \frac{\varepsilon^{(2)}_{w^*_i}}{\|\mathbf{w}\|} = \frac{1}{1+a}\varepsilon^{(2)}_{w^*_i}$$
$$\varepsilon^{(2)}_{v^*_i} \le \varepsilon^{(2)}_{x^*_i} + b\varepsilon^{(2)}_{q^*_i}.$$

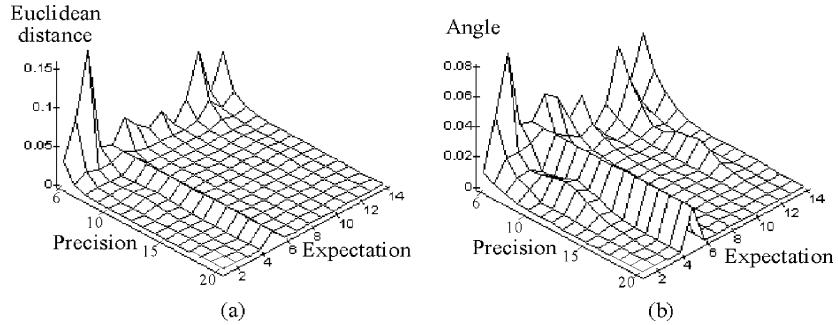


Fig. 10. Euclidean distance and angle between top-down expectation vectors with *fixed-point* incremental communication and those with conventional communication in Experiment 1.

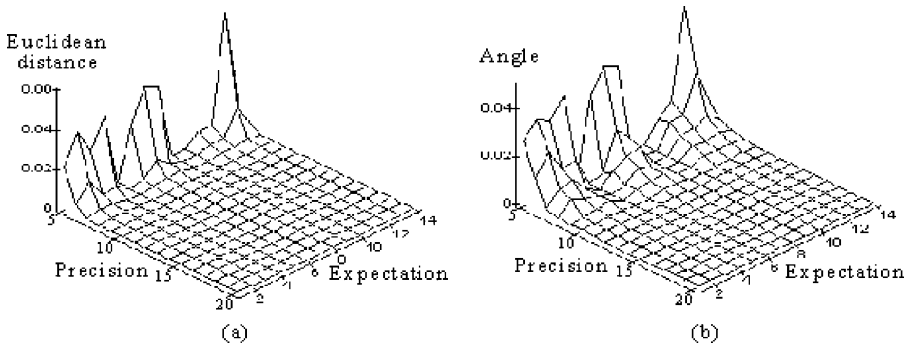


Fig. 11. Euclidean distance and angle between top-down expectation vectors with *floating-point* incremental communication and those with conventional communication in Experiment 1.

2) Error Analysis for the nth Iteration: In successive iterations, the errors in the STM activity values of F_1 nodes for the successive iterations can be derived similar to those for the second iteration

$$\begin{aligned}
\varepsilon_{z_{Ji}^*}^{(n)} &\le \alpha\varepsilon_{z_{Ji}^*}^{(n-1)} + \beta\varepsilon_{u_i^*}^{(n-1)} \\
\varepsilon_{w_i^*}^{(n)} &\le a\varepsilon_{u_i^*}^{(n-1)} \\
\varepsilon_{x_i^*}^{(n)} &\le \frac{1}{1+a}\varepsilon_{w_i^*}^{(n)} = E\varepsilon_{w_i^*}^{(n)} \\
\varepsilon_{p_i^*}^{(n)} &\le \varepsilon_{u_i^*}^{(n-1)} + d(\varepsilon_{z_{Ji}^*}^{(n)} + 2^{-k}) \\
\varepsilon_{v_i^*}^{(n)} &\le \varepsilon_{x_i^*}^{(n)} + b\varepsilon_{q_i^*}^{(n)} \\
\varepsilon_{q_i^*}^{(n)} &\le \frac{\varepsilon_{p_i^*}^{(n)}}{\|\mathbf{p}^{(n)}\|} = G^{(n)}\varepsilon_{p_i^*}^{(n)} \\
\varepsilon_{u_i^*}^{(n)} &\le \frac{1}{1+b}\varepsilon_{v_i^*}^{(n)} = H\varepsilon_{v_i^*}^{(n)} \\
\varepsilon_{r_i^*}^{(n)} &\le \frac{2c}{1+c\|\mathbf{p}^{(n)}\|}[\varepsilon_{u_i^*}^{(n)} + d(\varepsilon_{z_{Ji}^*}^{(n)} + 2^{-k})]
\end{aligned} \tag{23}$$

where for F_1 STM activities we assume that $E = 1/(1+a)$, $H = 1/(1+b)$, $G(n) = 1/\|\mathbf{p}^{(n)}\|$, $\varepsilon_{z_{Ji}^*}^{(0)} = 0$, and $\varepsilon_{u_i^*}^{(0)} = 2^{-k}$.

Since the fourth-order Runge–Kutta solver is used in the simulator and $z_{Ji}^{(1)} = \beta Q_i$, by (22) the top-down weight values in the nth iteration can be obtained as

$$\begin{aligned}
z_{Ji}^{(n)} &= \alpha z_{Ji}^{(n-1)} + \beta u_i^{(n-1)} \\
&= \alpha^{(n-1)}\beta Q_i + \alpha^{(n-2)}\beta Q_i + \cdots + \alpha\beta Q_i + \beta Q_i \\
&= \beta\frac{1-\alpha^n}{1-\alpha}Q_i.
\end{aligned} \tag{24}$$

Therefore the norm of top-down weights in the nth iteration is given as

$$\|z_J^{(n)}\| = \beta\frac{1-\alpha^n}{1-\alpha}. \tag{25}$$

By (1), (10), and (24) the norm of vector $\mathbf{p}$ for the nth iteration is given as

$$\|\mathbf{p}^{(n)}\| = 1 + d\beta\frac{1-\alpha^n}{1-\alpha}.$$

After applying n successive operators, the final errors in u_i and z_{Ji} can be expressed as

$$\begin{aligned}
\varepsilon_{u_i^*}^{(n)} \le\ & H(Ea + bG^{(n)})\varepsilon_{u_i^*}^{(n-1)} + bdG^{(n)} \\
& \times \left(\beta\frac{1-\alpha^n}{1-\alpha}2^{-k} + \beta\sum_{l=1}^{n-1}\alpha^{n-1-l}\varepsilon_{u_i^*}^{(l)} + 2^{-k}\right)
\end{aligned} \tag{26}$$

$$\varepsilon_{z_{Ji}^*}^{(n)} \le \beta\frac{1-\alpha^n}{1-\alpha}2^{-k} + \beta\sum_{l=1}^{n-1}\alpha^{n-1-l}\varepsilon_{u_i^*}^{(l)} \tag{27}$$

where $\varepsilon_{z_{Ji}^*}^{(0)} = 0$ and $\varepsilon_{u_i^*}^{(0)} = 2^{-k}$.

From (23) and (27), it is seen that the magnitude of errors in z_{Ji}, and therefore in p_i, are increasing functions of the number of iterations. The growth can be serious if the number of iterations is very large or the precision used is too low. Equations (26) and (27) give the maximum possible errors if and only if all the errors on the right sides of the equations are the largest possible errors and they have the same signs. It is very unlikely that both conditions are satisfied at the same time. Moreover, in

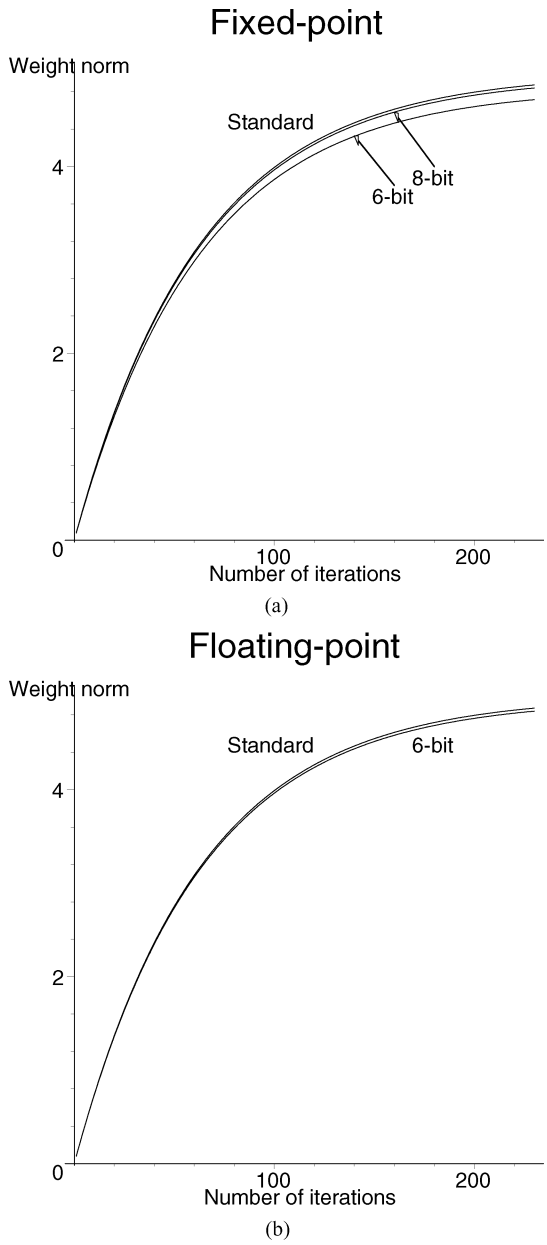


Fig. 12. Evolution of top-down weight norm with incremental and conventional communications for the pattern which accesses an uncommitted F_2 node in Experiment 1. (a) Fixed-point communication. (b) Floating-point communication.

practice the errors could be of opposite signs. All these factors might contribute toward good convergence behavior of incremental communication.

Through the analysis given above we can see that the propagated error caused by normalization does not enforce instability. The errors computed based on the equations derived in this section are referred to as *analytical errors*. Fig. 8 depicts the relation among the errors in p_i, q_i, u_i, and r_i for 10-b precision. According to (2) and (15) q_i is obtained by normalizing vector $\mathbf{p}$, and r_i is computed by normalizing vectors $\mathbf{Q}$ and $\mathbf{p}$. The errors in r_i and q_i are much less than that in p_i. The error in u_i is almost the same as that in q_i. This is reasonable since for conventional communication u_i and q_i should have the same value during the learning process as shown in Table III.

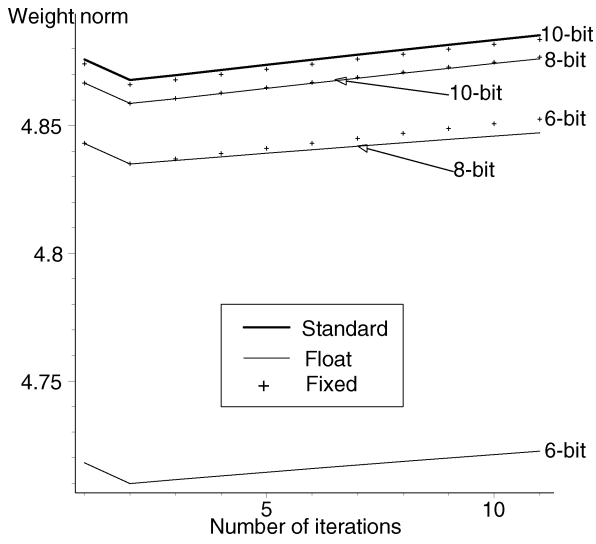


Fig. 13. Evolution of top-down weight norm with incremental and conventional communications for the pattern which accesses a committed F_2 node in Experiment 1.

As the precision increases, the errors in F_1 STM activities and weights should decrease as discussed previously in this section. This is observed for the analytical error of the top-down weights given in Fig. 9.

The above error analysis assumes a learning process during which a new category is established. When an established category is chosen, the value of A in (19) is not zero and $||z_J||$ is close to $1/(1-d)$ at the beginning as well as at the end of its learning trial. In this case, although vectors $\mathbf{u}$, $\mathbf{q}$, $\mathbf{p}$, and $\mathbf{x}$ are not exactly equal to vector $\mathbf{Q}$, their values are close to the value of $\mathbf{Q}$. Moreover, the number of iterations required by the network to converge is less than that for establishing a new category. Therefore, when an established category is chosen, the learning process can be seen as the adjustment of the weights for a few more iterations and following the learning process of the input pattern which accesses the same category most recently. In the following subsection, this is tested by analyzing the learning process of a selected pattern.

D. Simulation Results

We examine the incremental communication method with respect to the changes in the weights during learning and closeness of weight vectors with those of the conventional communication. Simulations are performed for two sets of network parameters. The results obtained with various limited precisions incremental communication are compared with those for the conventional communication.

Two similarity measures are used. The first similarity measure is the Euclidean distance between weight vectors. The Euclidean distance between two weight vectors $\mathbf{w}$ and $\mathbf{v}$ with cartesian coordinates $(w_1, v_1), (w_2, v_2), \ldots, (w_n, v_n)$ is obtained by

$$d(\mathbf{w}, \mathbf{v}) = \sqrt{(w_1 - v_1)^2 + (w_2 - v_2)^2 + \cdots + (w_n - v_n)^2}.$$

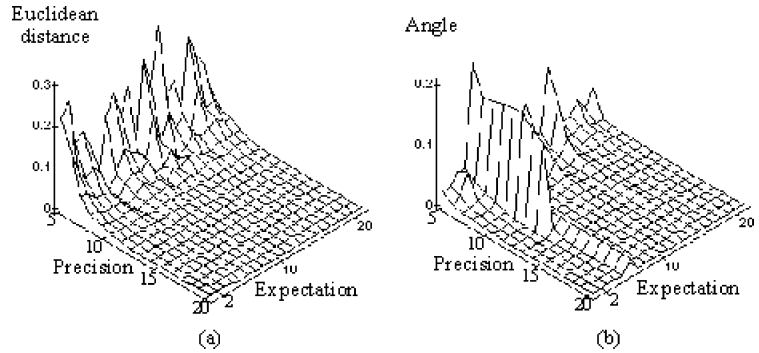


Fig. 14. Euclidean distance and angle between top-down expectation vectors with *fixed-point* incremental communication and those with conventional communication in Experiment 2.

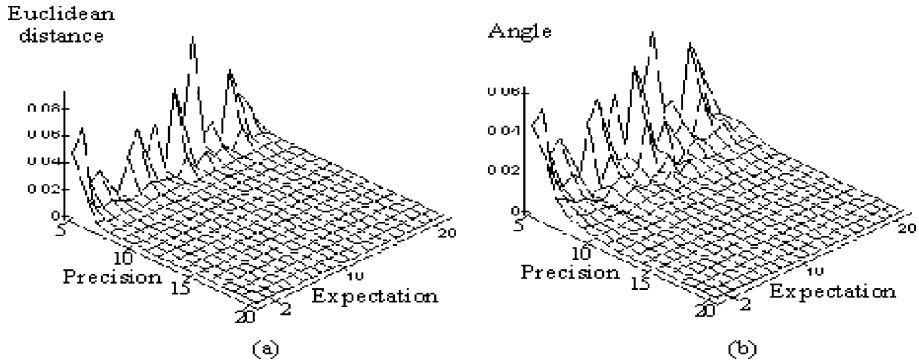


Fig. 15. Euclidean distance and angle between top-down expectation vectors with *floating-point* incremental communication and those with conventional communication in Experiment 2.

The second similarity measure calculates the angle (θ) between two weight vectors $\mathbf{w}$ and $\mathbf{v}$ as

$$\theta = \arccos\left(\frac{\mathbf{w}\cdot\mathbf{v}}{||\mathbf{w}||||\mathbf{v}||}\right).$$

Since two weight vectors may not have the same length, the dot product fails as a measure of similarity. However, when the angle θ between two weight vectors is accompanied with their Euclidean distance, one can clearly asses their closeness or similarity.

Experiment 1: The convergence behavior for the test problem given in Section III is examined using the incremental as well as conventional communication methods for two selected patterns. To examine the effect of large number of iterations on the propagated error we have chosen a pattern that requires large number of iterations. For ART2 networks the number of learning iterations required by an uncommitted F_2 node is much larger than that required for a committed F_2 node.

Fig. 10 gives plots of the Euclidean distance and the angle between 14 top-down expectations for the incremental fixed-point communication versus the conventional communication. Fig. 11 shows similar results for the floating-point representation. The floating-point representation is found to have better performance than the fixed-point representation.

Fig. 12 shows the evolutions of the top-down weight norm for the selected pattern accessing an uncommitted F_2 node during the learning phase. With 8-b fixed-point and 6-b floating-point precisions we obtain almost the same behavior as that with the conventional communication.

Fig. 13 gives the evolutions of the top-down weight norm for the selected pattern accessing a committed F_2 node. It is seen that when we use 8-b floating-point and 10-b fixed-point representations for the weights we obtain almost the same behavior.

Experiment 2: The plots of the Euclidean distance and the angle between top-down expectation vectors for the fixed-point and floating-point representations of incremental communication and the conventional communication are given in Figs. 14 and 15, respectively. The trends in the weights for the incremental and the conventional communication methods are generally similar. This implies that in the incremental communication the weight adjustment rule modifies the weights in the same direction as that of the conventional communication. A comparison of the two figures shows that the top-down expectation vectors for the floating-point representation are always closer to those obtained with the conventional communication.

The evolutions of the top-down weight norm for the selected pattern accessing an uncommitted F_2 node during the learning phase is shown in Fig. 16. It is seen that with 8-b fixed-point and 5-b floating-point precisions we obtain almost the same behavior as that with the conventional communication.

Fig. 17 gives the evolutions of the top-down weight norm for the selected pattern accessing a committed F_2 node. It is seen that we obtain almost the same behavior when we use 8-b floating-point representation and 10-b fixed-point representation for the weights. We can see that the length of the weight

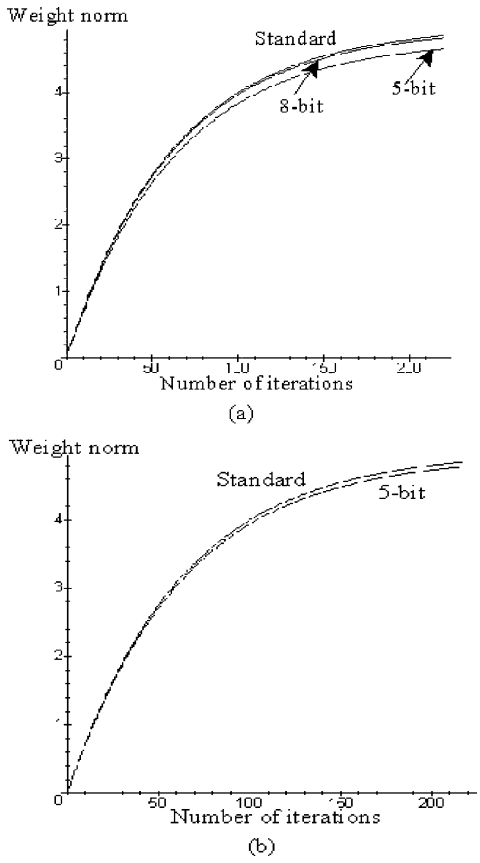


Fig. 16. Evolution of top-down weight norm with incremental and conventional communications for the pattern which accesses an uncommitted F_2 node in Experiment 2.

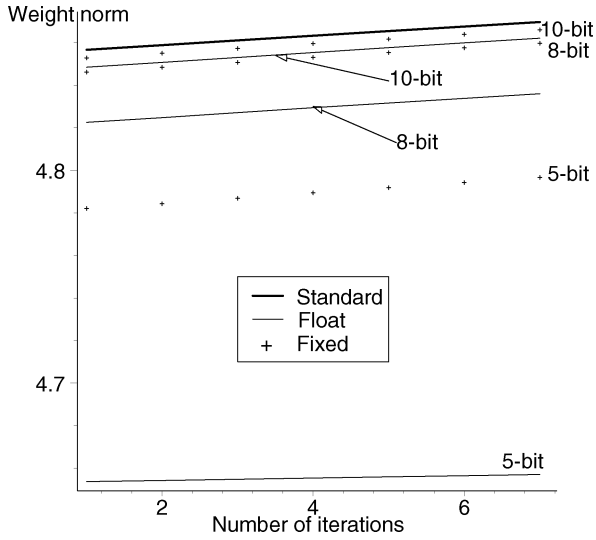


Fig. 17. Evolution of top-down weight norm with incremental and conventional communications for the pattern which accesses a committed F_2 node in Experiment 2.

vector is the main difference between the conventional and incremental methods. Note that the weight vector norm of the 10-b fixed-point representation is closer to the norm of the conventional weight vector as compared to the 8-b floating-point representation.

Simulation and analytical results have shown that the errors due to the limited precision incremental communication do not cause serious network performance degradation. In other words, the limited precision incremental communication method allows network convergence without inducing instabilities and excessive increase in computation time.

The simulation results given in this paper are based on slow learning method. The incremental communication method can also be used with the fast learning method. In fast learning, the speed of learning is high enough so that weights can reach their asymptotic values while the current input is presented. However, it is known that fast learning is not able to smooth out noise and may result in over-fitting. Since incremental communication introduces errors which could be considered as noise, we suspect that the use of incremental communication with ART2 using fast learning may not produce similar results as conventional communication.

V. Conclusion

In this paper, the incremental communication method is used to reduce the communication and computation costs of ART2 networks. In the incremental intercommunication, instead of communicating the full magnitude of a variable only the increment or decrement of its previous value is sent on a communication link. A simulator is developed to investigate the behavior and convergence ability of ART2 networks using limited precision incremental communication.

Simulation results with slow learning show that the increase in the number of iterations required by the network is small when the incremental communication is used. In some cases with an adequate number of bits, the required number of iterations is very close to, or sometimes even smaller than, that for the conventional communication. It is demonstrated that the communication costs are reduced by the limited precision incremental communication method. Simulation studies should be carried out using fast learning. Furthermore, incremental communication can be applied to other types of ART networks that use analog (real-valued) input patterns (e.g., ARTMAP [4], Fuzzy-ART [5], Fuzzy-ARTMAP [6], and fuzzy lattice neurocomputing (FLN) [15]).

A theoretical error analysis is carried out to analyze the effects of limited precision error. The error propagated by the normalization function is found to be always less than or equal to the normalized maximum of its input error. The effects of limited precision error on the orienting subsystem is found to be very minor. This allows us to argue that the normalization and nonlinear feedback processes are unlikely to cause instability in the network. The analytical results are in general agreement with simulation results.

The incremental communication method reduces the communication cost significantly. Therefore, the method is attractive for parallel and special-purpose VLSI implementations of ART2 networks. The trends of the simulation and analytical results in this paper are almost the same as those in our previous work on feedforward neural networks. Thus, the incremental communication method is suitable for feedforward as well as recurrent neural networks.

REFERENCES

[1] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vis. Graph. Image Process.*, vol. 37, pp. 54–115, 1987.

[2] ——, "ART2: Self-organization of stable category recognition codes for analog input patterns," *Appl. Opt.*, vol. 26, pp. 4919–4930, 1987.

[3] ——, "ART2-A: An adaptive resonance algorithm for rapid category learning and recognition," *Neural Netw.*, vol. 4, pp. 493–504, 1991.

[4] G. A. Carpenter, S. Grossberg, and J. H. Reynorlds, "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Netw.*, vol. 6, pp. 565–588, 1991.

[5] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Netw.*, vol. 4, no. 6, pp. 759–771, 1991.

[6] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynorlds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, Sep. 1992.

[7] D. D. Cavilia, M. Valle, and G. M. Bissio, "Effects of weight discretization on the backpropagation learning method: Algorithm design and hardware realization," in *Proc. Int. Joint Conf. Neural Networks*, vol. 2, San Diego, CA, 1990, pp. 631–637.

[8] M. Chen, "Incremental communication for adaptive resonance theory networks," MCS thesis, Faculty Comput. Sci., Univ. New Brunswick, Fredericton, NB, Canada, 1998.

[9] P. Gaudiano. (1997) ART2 – A Simple ART2 Simulation Program. [Online]. Available: ftp://cns-ftp.bu.edu/pub/ art2.shar.gz

[10] A. A. Ghorbani and V. C. Bhavsar, "Incremental communication for artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1375–1385, Nov. 1995.

[11] ——, "Incremental communication for multilayer neural networks: Error analysis," *IEEE Trans. Neural Netw.*, vol. 9, no. 1, pp. 68–82, Jan. 1998.

[12] S. Grossberg, "Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors," *Biol. Cybern.*, vol. 23, pp. 121–134, 1976.

[13] ——, "Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, and illusions," *Biol. Cybern.*, vol. 23, pp. 187–202, 1976.

[14] P. W. Hollis, J. S. Harper, and J. J. Paulos, "The effects of precision constraints in a backpropagation learning network," *Neural Computat.*, vol. 2, pp. 363–373, 1990.

[15] V. G. Kaburlasos and V. Petridis, "Fuzzy lattice neurocomputing (FLN) models," *Neural Netw.*, vol. 13, no. 10, pp. 1145–1170, 2000.

[16] K. Nakayama, S. Inomata, and Y. Takeuchi, "A digital multilayer neural network with limited binary expressions," in *Proc. Int. Joint Conf. Neural Networks*, vol. 2, San Diego, CA, 1990, pp. 578–592.

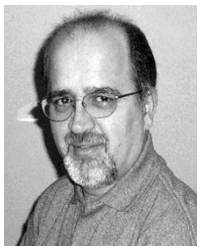
[17] T. Serrano-Gotarredona, B. Linares-Barranco, and A. G. Andreou, *Adaptive Resonance Theory Microchips: Circuit Design Techniques*. Norwell, MA: Kluwer, 1998.

[18] A. Singer, "Implementations of artificial neiral networks on connection machine," *Parallel Comput.*, vol. 14, pp. 305–315, 1990.

[19] J. R. Taylor, *An Introduction to Error Analysis*. Mill Valley, CA: Univ. Science Books, 1982.

[20] "INMOS limited," in *The Transputer Databook*, 2nd ed. Berkeley, CA: Consolidated Printers, 1989.

Ming Chen Photograph and biography not available at time of publication.



Ali A. Ghorbani (M'95) received the M.Sc. degree from George Washington University, Washington, DC, in 1979 and the Ph.D. degree from the University of New Brunswick (UNB), Fredericton, NB Canada, in 1995.

He was on the faculty of the Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran, from 1987 to 1998. Since 1999, he has been with the faculty of Computer Science, UNB, where he is currently a Professor of Computer Science. He is also a Member of the Privacy, Security, and Trust (PST) Team, National Research Council (NRC) of Canada. He has held a variety of positions in academia for the past 24 years. His research originated in software development, where he designed and developed a number of large-scale systems. His current research focus is neural networks, web intelligence, agent systems, complex adaptive systems, and trust and security. He established the Intelligent and Adaptive Systems (IAS) research group in 2002 at the faculty of Computer Science, UNB. The IAS group pursues research on machine and statistical learning, data mining, intelligent agents, and multiagent systems. The group is also home to R&D in Web intelligence, network security and application of multiagent systems to e-Health. He authored more than 90 research papers in journals and conference proceedings and has edited four volumes. He is on the editorial board of *Computational Intelligence* (CI), an international journal. He is also an Associate Editor for the *International Journal of Information Technology and Web Engineering*.

Dr. Ghorbani a Member of ACM, the IEEE Computer Society, and ANNS.

Virendrakumar C. Bhavsar (M'84) received the B.Eng. degree in electronic and telecommunications from the University of Poona, India, in 1971 and the M.Tech. and Ph.D. degrees in electrical engineering from the Indian Institute of Technology, Bombay, in 1973 and 1982, respectively.

He was on the faculty of the Department of Computer Science and Engineering, Indian Institute of Technology, from 1974 to 1983. Since 1983, he has been at the University of New Brunswick (UNB), Fredericton, NB Canada, where he is currently a Professor and Dean of the faculty of Computer Science. He is also the Director of Advanced Computing Research Lab (ACRL) at UNB. He was a Visiting Professor at the Center for Development of Advanced Computing, Pune, India, in 1990. He has authored more than 130 research papers in journals and conference proceedings and has edited two volumes. His current research interests include parallel and distributed processing, learning machines, and multiagent systems.

Dr. Bhavsar is a member of ACM, SIAM, and CIPS, and holds Information Systems Professional (ISP).