INCREMENTAL COMMUNICATION FOR MULTILAYER NEURAL NETWORKS: ERROR ANALYSIS

by

Ali A. Ghorbani
Virendrakumar C. Bhavsar

TR96-104, March 1996

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
E-mail: fcs@unb.ca
www: http://www.cs.unb.ca

Incremental Communication for Multilayer Neural Networks: Error Analysis

Ali A. Ghorbani and Virendrakumar C. Bhavsar

The authors are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada E3B 5A3 Phone: (506)453-4566 Fax: (506)453-3566 E-mail: ghorbani@unb.ca bhavsar@unb.ca

March 4, 1996

Abstract

Artificial neural networks (ANNs) involve large amount of inter-node communications. To reduce the communication cost as well as time of learning process in ANNs, we have earlier proposed incremental inter-node communication method. In the incremental communication method, instead of communicating the full magnitude of the output value of a node, only the increment or decrement to its previous value is sent on a communication link. In this paper, the effects of limited precision incremental communication method on the convergence behavior and performance of multilayer neural networks are investigated. The nonlinear aspects of representing the incremental values with reduced (limited) precision on the commonly used error backpropagation training algorithm are analyzed. It is shown that the nonlinear effect of small perturbation in the input(s)/output of a node does not enforce instability. The analysis is supported by simulation studies of two problems.

I. INTRODUCTION

We have earlier proposed the incremental inter-node communication method for inter-node communication in the Artificial Neural Networks(ANNs)[10]. In the incremental communication method, instead of communicating the whole value of a variable, only the increment or decrement to its previous value is sent on a communication link. The incremental communication, when implemented by limiting the precision used for the incremental values that are communicated between various nodes, reduces the communication complexity of ANNs by limiting the node's input/output bandwidth requirements. Note that in the incremental communication method all the operations inside a node may be carried out using either full precision or other limited precision strategies suggested in literature [2], [12], [13], [14], [24]. It has been shown through simulation that for some problems even 4-bit precision for incremental values in fixed- and/or floating- point representations is sufficient for the network to converge. With 8-12 bit precisions almost the same results are obtained as that with the conventional communication using 32-bit precision.

The use of limited precision incremental values reduces the accuracy and may degrade the performance of an ANN learning algorithm; in other words, it may lead to deviations in the performance of the learning algorithm compared to the full precision implementation of incremental communication. In some circumstances, the limited precision of incremental values may even cause smaller output error than that of the full precision (see the parity problem in [10]). This is due to the fact that the errors that are caused by the limited precision representation are signed magnitude values and can assume both positive and negative values and therefore, they can have both positive and negative impact on the size of the final error (i.e., some of the errors may cancel each other).

The price paid in using reduced precision for the incremental values may be a degradation in the performance of learning algorithms due to the truncation and/or round off errors. These errors contaminate almost all computations, thus, some analysis of their effects is required in order to judge the reliability of the results obtained. The degradation may primarily be the result of two factors. Firstly, when reduced-precision is used, the learning algorithm may yield a larger output error than the case where full precision is used in the representation of incremental values. Secondly, the limited precision errors may accumulate and increase with time, until they destroy the normal operation of the learning algorithm (i.e., the overall error may become so large that the final result obtained may be unacceptable).

The behavior of least squares and stochastic-gradient-based adaptive filtering algorithms in limited precision environment, where the adjustable parameters as well as all internal variables are quantized to within a least significant digit (LSD), has attracted a lot of attention (see for example [3], [4], [11]). The issue of the precision required by a threshold gate in order to implement a linear threshold function is also addressed by few researchers[16]. The finite word length arithmetic effects on the least-mean-square (LMS) adaptive weights is investigated in [1] and other researchers (see for example references in [1]). However, in the case of ANNs and various learning algorithms, the issue of reduced precision effects on the learning behavior has been largely ignored until recently[2], [7], [12], [13], [14], [25].

The main objective of this paper is to analyze the nonlinear effect of the reduced precision input/output incremental values of nodes on the learning behavior of the commonly used multilayer perceptrons (MLPs). In our analysis, we assume that the internal processing within a node is carried out using full precision. We use both mathematical and statistical models to carry out the error analysis and investigate the relationships between the precisions of the incremental values and the convergence of the multilayer neural networks. In other words, the main focus is to analyze effects of small perturbations in the input/output of the internal nodes. The results from mathematical and statistical evaluations are supported by analyzing the effects of the limited precision incremental communication method on two learning problems through computer simulation.

This paper is organized as follows. In the next section the incremental communication method is briefly reviewed. Various sources of errors in ANNs using limited precision are discussed in Section 3. For a chain of nodes with linear and nonlinear activation functions, general equations of limited precision error are derived in Section 4. In Section 5, these equations are used to estimate the error arising in the forward and backward passes of the backpropagation learning algorithm. The simulation results for two learning problems are given in Section 6. Finally, the conclusions of the paper are given.

II. Incremental Communication

Generally, a node behaves in a continuous manner, in other words its output, y, changes by $\pm\Delta y$ which is typically of smaller range than the range of the actual output y. An important decision in communicating a continuous signal from a source node to a destination node is whether to send the full magnitude of the signal, y, over the interconnection or the amount of change, Δy, that has taken place in the unit time interval from step t to step $t+1$. We have shown that the incremental value, Δy, which is typically small in magnitude as well as *range*, would be a better candidate for communication between nodes [10].

In 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. For example, assume that node j has to communicate the signal y_j to node i at different time instants. If $y_j(t)$ is the output of node j at time t and $y_j(t+1)$ is its output at time $t+1$, in conventional communication the communication link will carry the value $y_j(t+1)$. In contrast, in the incremental communication the communication link will carry the value $\Delta y_j(t+1)$, where $\Delta y_j(t+1) = y_j(t+1) - y_j(t)$. At the receiving end, the value $y_j(t+1)$ will be obtained by updating the previous value $y_j(t)$ stored at i with $\Delta y_j(t+1)$.

We have found that for some problems even four-bit precision for Δy in fixed- and/or floating- point representations is sufficient for the network to converge. With the 8-12 bit precisions almost the same results are obtained as that with the conventional communication using 32-bit precision. In these cases, the increase in the number of epochs is found to be small. In fact, in some cases with an appropriate number of bits for incremental values, which is found to be much smaller than for the full-precision values, the number of epochs is very close to and sometimes even better or the same as in conventional communication (see Table II in [10]).

The concept of incremental inter-node communication is applicable to many other classes of ANNs. It can also be used along with the other limited precision strategies for representing variables suggested in the literature. The proposed method of communication can be applied for parallel implementations of the ANNs (see [8] for details).

III. Sources of Limited Precision Errors

In this section we define several basic concepts of limited precision error analysis and show how these concepts can be used in the limited precision error analysis of a node, with single and multiple inputs, containing linear and/or nonlinear activation functions. In the next section we carry out the limited precision error analysis of the incremental communication method using most of the ideas that are developed in this section.

Given the following general case

$$y = f(x), \qquad y^* = f\left(\Phi(x)\right), \tag{1}$$

where x, y, and f are input, output, and activation function of a node, respectively, there are at least two basic sources of errors in the output y. First, there is an error ε_x in the input argument x, which stems from limited precision approximation $\Phi(.)$ of x. Second, there is a propagated error, ε_y, which is generated from applying f to erroneous input. The propagated error grows with the number of operations. For our case we assume $f(.)$ is a nonlinear function that generates propagated error, ε_y, and $\Phi(.)$ is an approximation function, that produces the limited precision error ε_x.

The effect of propagating the limited precision error through functions such as sums, differences, products, and quotient can be calculated very easily (see [23]); however, many detailed calculations are required for more complicated operations, such as nonlinear functions.

Based on the Taylor series, for any differentiable function $f(x)$ and any sufficiently small ε, we have

$$\varepsilon_f = f(x+\varepsilon_x) - f(x) = \frac{df}{dx}\varepsilon_x. \tag{2}$$

Lemma III.1: If logistic function, φ, is used as the activation function of a node and if the input x to a node is represented in *fixed-point 2's-complement* representation with binary point to the right of the most significant bit and x is approximated using *truncation* with *b-bit* precision, then the output error of the node (i.e., the output error, $\varepsilon_{\varphi(x)}$, of the nonlinear logistic function) is bounded by the open-closed interval $\left(-2^{-(b+2)}, 0\right]$, whereas if *rounding* is used the error, $\varepsilon_{\varphi(x)}$, in the output of a node is bounded by the open-closed interval $\left(-2^{-(b+3)}\ ,\ 2^{-(b+3)}\right]$.

Proof. The first derivative of the logistic function φ is found to be

$$\varphi'(x) = \varphi(x)(1-\varphi(x)).$$

For $\varphi(x) \in [0,1]$, then $\varphi'(x) \in [0, 0.25]$. Since

$$\varepsilon_x \in \begin{cases} \left(-2^{-b}, 0\right], & \text{for truncation} \\ \left(-2^{-(b+1)}\ ,\ 2^{-(b+1)}\right], & \text{for rounding} \end{cases}$$

we can conclude that

$$\varepsilon_{\varphi(x)} = \varphi'(x)\varepsilon_x \in \begin{cases} \left(-2^{-(b+2)}\ ,\ 0\right] & \text{for truncation} \\ \left(-2^{-(b+3)}\ ,\ 2^{-(b+3)}\right] & \text{for rounding} \end{cases} \tag{3}$$

□

From the above lemma, it is seen that the range of the propagated error caused by the logistic function is always less than the limited precision error generated by imposing limited precision on the input value of the function.

For the sigmoid functions such as the *hyperbolic tangent* function, the propagated limited precision error can be easily found to be bounded by the open-closed interval $\left(-2^{-b},0\right]$, which is the same as the truncation error of the input of a node. Thus, the propagated error caused by a sigmoid function does not enforce instability. This could be one of the contributing factors to the stability of the learning process when the input(s)/output of the nodes are perturbed due to limited precision representation strategies.

Figure 1 shows the discrepancy between the actual error generated by representing the input value of the logistic function with limited precision and the analytical error computed by using Equation 2. The results of our experiments with other limited precisions show a slight difference between the output of the logistic function sought from full precision and that of the limited precision. The discrepancy becomes smaller as the precision gets larger. With 4-bit and 8-bit precisions, the errors are under ± 0.1 and ± 0.01, respectively. With 12-bit precision, the propagated error is small and may be discarded.

In the ensuing analysis of the limited precision incremental communication error, we consider the limited precision error to be a discrete random variable distributed over a range determined by the number of truncated bits. Further, we assume (as given in [14], [25]) that the limited precision errors have the following properties:

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

A. *Error Generation by Single node*

Let x and $\overline{x}(\overline{x} = \Phi(x))$ represent the full precision and limited precision values of a single input to a node, respectively. Let ε_f denote the error caused by representing the output of a node with reduced precision, and w represent a full precision error free argument(weight).

Using the first order Taylor series, the output of a node whose input is represented with limited precision is given as

$$
\begin{aligned}
y^* = \Phi\left(f(\overline{x})\right) &\equiv f(w(x+\varepsilon_x)) + \varepsilon_f \\
&\approx f(wx + w\varepsilon_x) + \varepsilon_f \\
&\approx f(wx) + (w\varepsilon_x)f'(wx) + \varepsilon_f. \qquad (4)
\end{aligned}
$$

Assuming all arithmetic operations are carried out using full precision representation and furthermore, assuming that the round off errors which may occur as the result of multiplications are ignored, the total error is,

$$
\begin{aligned}
\varepsilon_y = y^* - y &\approx f(wx) + w\varepsilon_x(f'(wx)) + \varepsilon_f - f(wx) \\
&= w\varepsilon_x f'(wx) + \varepsilon_f. \qquad (5)
\end{aligned}
$$

Note that the magnitude of $w\varepsilon_x$ is directly dependent on the magnitude of w (i.e., the error increases as w increases). However, notice that the relative error is independent of the magnitude of w and is equal to

$$\frac{w\varepsilon_x}{w\overline{x}} = \frac{\varepsilon_x}{\overline{x}}.$$

Therefore, in linear systems, when multiplying a reduced precision nonzero value by weight factor w, the relative limited precision error remains unchanged, while the absolute limited precision error is increased $|w|$–fold.

B. Single Node with Multiple Inputs

The output of a function with n input variables, $x_1, x_2, \ldots, x_n$, where each input variable is approximated by limited precision, is approximated as

$$f(x_1 + \varepsilon_{x_1}, x_2 + \varepsilon_{x_2}, \ldots, x_n + \varepsilon_{x_n}) = f(x_1, x_2, \cdots, x_n) + \frac{\partial f}{\partial x_1}\varepsilon_{x_1} + \frac{\partial f}{\partial x_2}\varepsilon_{x_2} + \ldots + \frac{\partial f}{\partial x_n}\varepsilon_{x_n}.$$

Thus for a node with n inputs, the upper bound for the propagated error in the output of the node, ε_y, is given as,

$$\varepsilon_y \leq \left|\frac{\partial f}{\partial x_1}\varepsilon_{x_1}\right| + \ldots + \left|\frac{\partial f}{\partial x_n}\varepsilon_{x_n}\right|. \qquad (6)$$

When the limited precision errors in the input variables are independent and random, then the magnitude of the propagated error can be calculated as

$$\varepsilon_y = \sqrt{(\frac{\partial f}{\partial x_1}\varepsilon_{x_1})^2 + \ldots + (\frac{\partial f}{\partial x_n}\varepsilon_{x_n})^2}. \tag{7}$$

Note that whether or not the errors are independent and random, the Equation 6 always gives an upper bound on ε_y.

Consider a node with a differentiable function

$$y = f(a_1, a_2, \ldots, a_n). \tag{8}$$

Let $a_i = w_i\Phi(x_i)$, $i = 1, 2, \ldots, n$. After the substitutions we have

$$y^* = f\left(w_1\Phi(x_1), w_2\Phi(x_2), \ldots, w_n\Phi(x_n)\right), \tag{9}$$

i.e.,

$$y^* = f\left(w_1(x_1 + \varepsilon_{x_1}), w_2(x_2 + \varepsilon_{x_2}), \ldots, w_n(x_n + \varepsilon_{x_n})\right). \tag{10}$$

In Equation 10, $w_i\varepsilon_{x_i}$, $i = 1, 2, \ldots, n$, represent the weighted limited precision errors of the inputs of the node.

The absolute limited precision error of the node is,

$$|\varepsilon_y| = |y^* - y| = \left|f\left(w_1(x_1 + \varepsilon_{x_1}), \ldots, w_n(x_n + \varepsilon_{x_n})\right) - f\left(w_1x_1, \ldots, w_nx_n\right)\right|. \tag{11}$$

The limited precision errors of input variables, ε_{x_i}, are ordinarily small quantities whose products, squares and higher powers may be neglected. Therefore

$$\begin{aligned} \varepsilon_y &\leq \left|\sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\varepsilon_{x_i}w_i\right| \\ &\leq \sum_{i=1}^{n}\left|\frac{\partial f}{\partial x_i}\right|\left|\varepsilon_{x_i}w_i\right|. \end{aligned} \tag{12}$$

Substituting ∂f by ∂y^*, for small values of ε_{x_i}, we get

$$\varepsilon_y \leq \sum_{i=1}^{n}\left|\frac{\partial y^*}{\partial x_i}\right|\left|w_i\varepsilon_{x_i}\right|. \tag{13}$$

The relative propagated error of y can be calculated as

$$\frac{\varepsilon_y}{y^*} \leq \sum_{i=1}^{n} \left| \frac{\frac{\partial f}{\partial x_i}}{y^*} \right| |w_i \varepsilon_{x_i}| \approx \sum_{i=1}^{n} \left| \frac{\partial}{\partial x_i} \log f(.) \right| |w_i \varepsilon_{x_i}| . \tag{14}$$

Thus,

$$\frac{\varepsilon_y}{y^*} \leq \sum_{i=1}^{n} \left| \frac{\partial}{\partial x_i} \log y \right| |w_i \varepsilon_{x_i}| . \tag{15}$$

IV. Chain of Linear/Nonlinear Nodes

In this section, general equations of limited precision error of a chain of nodes with linear and nonlinear activation functions are derived. In Section 5, these equations are used in the error estimates of the limited precision incremental communication method of the backpropagation learning algorithm.

A. Linear Nodes

Assuming that all the functions applied at the nodes are linear, the bound for the final error for a chain of n linear nodes is the sum of the individual bounds. Thus, the bound of the final error can be expressed as follows:

$$\varepsilon_y \equiv y^* - y = \varepsilon_{f_n} + \sum_{i=2}^{n} f_n(f_{n-1}(\cdots(f_i(\varepsilon_{f_{i-1}})))) \tag{16}$$

where,

$$\varepsilon_{f_1} = f_1(\varepsilon_x).$$

From Equation 16, it is obvious that the magnitude of ε_y depends on the number of operations in the evaluation of y.

B. Nonlinear Nodes

In the case where all or some of the functions f_i used are nonlinear functions, the partial derivative approach gives a reasonable approximation [18, p. 213].

Let y_i^* represent the result of the application of the first i successive operators on an erroneous input x and $y = y_n$ represent the exact result of applying n successive operators.

Then,

$$\begin{aligned}
y_2^* &= f_2(f_1(x+\varepsilon_x)+\varepsilon_{f_1})+\varepsilon_{f_2}. \\
&= f_2(f_1(x))+(\varepsilon_x f_1'(x)+\varepsilon_{f_1})f_2'(f_1(x))+\varepsilon_{f_2}. \\
&= y_2+\varepsilon_x f_1'(x)f_2'(f_1(x))+\varepsilon_{f_1}f_2'(f_1(x))+\varepsilon_{f_2}. \\
&= y_2+\varepsilon_x\frac{\partial y_2}{\partial x}+\varepsilon_{f1}\frac{\partial y_2}{\partial y_1}+\varepsilon_{f_2}\frac{\partial y_2}{\partial y_2}. \\
&= y_2+\varepsilon_x\frac{\partial y_2}{\partial x}+\sum_{i=1}^{2}\varepsilon_{fi}\frac{\partial y_2}{\partial y_i}, \qquad (17)
\end{aligned}$$

where the derivative $\partial y_2/\partial y_i$ is an approximation of the derivative $\partial y_2/\partial y_i^*$. Similarly,

$$\begin{aligned}
y_i^* &= f_i(f_{i-1}(\cdots(f_2(f_1(x+\varepsilon_x)+\varepsilon_{f_1})+\varepsilon_{f_2})\cdots)+\varepsilon_{f_{i-1}})+\varepsilon_{f_i}. \\
&= f_i(y_{i-1}^*)+\varepsilon_{f_i}. \\
&= f_i(f_{i-1}(y_{i-2}^*)+\varepsilon_{f_{i-1}})+\varepsilon_{f_i}. \\
&= f_i(f_{i-1}(y_{i-2}^*)+\varepsilon_{f_{i-1}}f_i'(f_{i-1}(y_{i-2}^*))+\varepsilon_{f_i}. \\
&\;\cdot \\
&\;\cdot \\
&\;\cdot \\
&= y_i+\varepsilon_x\frac{\partial y_i}{\partial x}+\sum_{j=1}^{i}\varepsilon_{f_j}\frac{\partial y_i}{\partial y_j}. \qquad (18)
\end{aligned}$$

The bound for the propagated error after applying i consecutive nonlinear operators on the input x is given as

$$\varepsilon_{y_i}\equiv y_i^*-y_i=\varepsilon_x\frac{\partial y_i}{\partial x}+\sum_{j=1}^{i}\varepsilon_{f_j}\frac{\partial y_i}{\partial y_j}. \qquad (19)$$

Therefore, the final error bound for n successive nonlinear functions can be defined as follows,

$$\varepsilon_y=\varepsilon_{y_n}\equiv y^*-y=\varepsilon_x\frac{\partial y}{\partial x}+\sum_{i=1}^{n-1}\varepsilon_{f_i}\prod_{j=i+1}^{n}\frac{\partial y_j}{\partial y_{j-1}}+\varepsilon_{f_n}. \qquad (20)$$

To incorporate the impact of multiple erroneous inputs, N, on the final incremental communication error of successive nonlinear functions, we can revise Equation 20 as follows

$$\varepsilon_y \simeq \sum_{j=1}^{N_0} \varepsilon_{x_j} \frac{\partial y}{\partial x_j} + \sum_{i=1}^{n} \varepsilon_{f_i} \prod_{j=i+1}^{n} \frac{\partial y_j}{\partial y_{j-1}}. \tag{21}$$

V. Training Multilayer Feedforward Networks

Layered feedforward networks are networks made of layers of nodes with nonlinear activation functions stacked one on top of the other. The outputs of one layer feed the inputs of the following layer through weighted connections. Thus, errors in the input patterns are propagated through the layers. This means that, node's generated errors in layer ℓ become input error to the nodes of the layer $\ell+1$. In this network, the desired input/output mapping is accomplished through a (possibly large) number of iterations where each iteration consists of a forward pass and a backward error propagation. Figure 2 shows the forward pass of the backpropagation learning algorithm using the incremental communication method.

A. *Error Analysis of Forward Pass*

Figure 2 shows how to calculate the activation of a given node in hidden or output layer using the incremental communication method. Let vector $\boldsymbol{y}$ represent the activation values of nodes in a given layer and the vector $\overline{\Delta}\boldsymbol{y}$ represent the corresponding limited precision incremental activation values. Then an element of $\overline{\Delta}\boldsymbol{y}$ is a couple equation $(\Delta\boldsymbol{y}, \varepsilon_{\Delta\boldsymbol{y}})$ where $\Delta\boldsymbol{y}, \varepsilon_{\Delta\boldsymbol{y}} \in \Re$. The value $\varepsilon_{\Delta\boldsymbol{y}}$ is a real number representing an error on $\overline{\Delta}\boldsymbol{y}$. Thus, $\overline{\Delta}\boldsymbol{y}$ is an approximation of the $\Delta\boldsymbol{y}$ with error $\varepsilon_{\Delta\boldsymbol{y}}$. The $\varepsilon_{\Delta\boldsymbol{y}}$ is generally an unknown quantity which can be considered as a random variable, approximated by its mean value and standard deviation.

To illustrate the impact of the incremental communication method and calculate the generated and propagated errors, we consider a two-layer MLP network. Let vectors $\boldsymbol{x}$, $\boldsymbol{h}$, and $\boldsymbol{y}$ represent the input patterns, hidden layer activation values, and network's output values, respectively. The subscripts i, j, and k are used to refer to the nodes in the input, hidden, and output layers, respectively.

Using the incremental communication method, the net input of the hidden node j is computed as

$$net_j^*(t) = \sum_i w_{ji}(t)\overline{x}_i(t) = \sum_i w_{ji}(t)(x_i(t) + \varepsilon_{x_i(t)}), \tag{22}$$

where, $\varepsilon_{x_i(t)}$ is the result of the reduced-precision approximation of $\Delta x_i(t)$ and is calculated as

$$\begin{aligned} \varepsilon_{x_i(t)} = \varepsilon_{\Delta x_i(t)} &= \Phi(x_i(t) - x_i(t-1)) - (x_i(t) - x_i(t-1)) \\ &= \Phi(\Delta x_i(t)) - \Delta x_i(t). \end{aligned} \tag{23}$$

Therefore, the total error received by the hidden node j is,

$$\varepsilon_{net_j(t)} = \sum_i w_{ji}(t)(\varepsilon_{\Delta x_i(t)}). \tag{24}$$

The time step t is dropped from all equations hereafter, unless it is absolutely necessary. The output of the node is computed by applying a nonlinear activation function (usually the sigmoid function) to the net_j.

$$h_j^* = \varphi(net_j^*) = \varphi(\sum_i w_{ji}\overline{x}_j). \tag{25}$$

Note that the threshold can be taken care of by using an extra input unit clamped to -1 and is connected to all units in the network.

The following partial derivative can be used to measure the effects of the changes in the input variables (x_i) on the output of the hidden node j:

$$\begin{aligned} \frac{\partial h_j^*}{\partial \overline{x}_i} &= \frac{\partial h_j^*}{\partial net_j^*}\frac{\partial net_j^*}{\partial \overline{x}_i} \\ &= \frac{\partial \varphi_j^*}{\partial net_j^*}\frac{\partial net_j^*}{\partial \overline{x}_i} \\ &= \left(\varphi_j^*\right)' w_{ji}, \end{aligned} \tag{26}$$

where for the logistic sigmoid function $\left(\varphi_j^*\right)' = \varphi'(net_j^*) = \varphi(net_j^*)\left(1 - \varphi(net_j^*)\right)$.

From the Taylor's theorem, the error generated by function φ is calculated as follows:

$$\varphi(net_j^*) = \varphi(net_j + \varepsilon_{net_j}) = \varphi(net_j) + \varepsilon_{net_j}\varphi'(net_j) + \varepsilon_{net_j}\frac{\varphi''(\theta)}{2}, \tag{27}$$

for $\theta \in \left[net_j, net_j^*\right]$. Assuming that $\varepsilon_{net_j} \frac{\varphi''(\theta)}{2}$ is small enough to be ignored, and furthermore, assuming that the multiply-add operations involved in the inner product of weight and activation vectors can be carried out without generating significant error, the generated limited precision error for the hidden node j is

$$\varepsilon_{\varphi(net_j^*)} = \varepsilon_{net_j} \varphi'(net_j). \tag{28}$$

Since all activation values (x_i) received by the hidden node j are approximated values (i.e., erroneous inputs) and since

$$\varepsilon_{net_j} = \sum_i \varepsilon_{\Delta x_i} w_{ji} \tag{29}$$

using the n-dimensional Taylor series, we can conclude that,

$$\begin{aligned} \varepsilon_{h_j^*} \stackrel{\text{def}}{=} \varepsilon_{\varphi(net_j^*)} &= \sum_i \varepsilon_{\Delta x_i} \frac{\partial \varphi(net_j^*)}{\partial \overline{x}_i} \\ &= \sum_i \varepsilon_{\Delta x_i} \cdot \varphi'(net_j^*) w_{ji}. \end{aligned} \tag{30}$$

If we adopt the standard deviation $\sigma(\varepsilon_{h_j^*})$ as our measure of the uncertainty in ε, then Equation 30 is really the upper limit on the error. Regardless of whether the errors in x are independent or not, and regardless of whether they are normally distributed or not, the error will not exceed the right-hand side of Equation 30.

Equation 30 yields the maximum possible error if and only if the error of all the terms of net_j are the largest possible terms and they have the same signs. The chance of both conditions to happen at the same time is negligible. Moreover, given a large number of terms, the errors in separate terms could be of opposite sign and consequently, partially neutralize one another. This is the main reason why the performance of the reduced precision incremental communication method is problem dependent.

The practical limited precision error, $\varepsilon^{(p)}$, of the weighted input of node j at time n can be calculated using probability theory as follows[6],

$$\varepsilon_{net_j^*}^{(p)} \leq \xi \sqrt{N_0}, \tag{31}$$

where N_0 is the number of nodes sending their outputs to node j and the absolute errors in the terms of net_j^* do not exceed the value ξ with a probability, P, exceeding the value ϖ; that is,

$$P\left(|\varepsilon_{\Delta x_i}.w_{ji}| \leq \xi\right) > \varpi. \tag{32}$$

Therefore, if b bits are used to represent incremental values of full precision values of B bits, the largest possible absolute truncation error is 2^{-b}, and the practical limited precision error, $\varepsilon^{(p)}$, of net_j^* is

$$\begin{aligned} \varepsilon^{(p)}_{net_j^*} &\leq 2^{-b} \cdot 2^{B}\sqrt{N_0} = 2^{(B-b)}\sqrt{N_0} \\ \varepsilon^{(p)}_{net_j^*} &\leq 2^{\hat{b}}\sqrt{N_0} \end{aligned} \tag{33}$$

where $\hat{b}$ is the number of bits that is being truncated from the incremental values (i.e., $\hat{b} = B - b$). The practical error of the hidden node j at step n is given as

$$\varepsilon^{(p)}_{\varphi_j^*} \leq 2^{\hat{b}}\sqrt{N_0}\varphi'\left(net_j\right). \tag{34}$$

Thus, when the logistic sigmoid function is used to compute the output of the hidden node j, we have,

$$\varepsilon^{(p)}_{\varphi_j^*} \in \left[0\ ,\ \sqrt{N_0}2^{\hat{b}-2}\right]. \tag{35}$$

From this we can conclude that the generated error with the output of any nonlinear logistic function is bounded by the closed interval $\left[0\ ,\ \sqrt{N_0}2^{\hat{b}-a}\right]$, where a depends on the properties of the function and its first derivative. For example, a is 0 for $tanh$ and 2 for logistic sigmoid functions (see Lemma III.1).

It is known that all the values of the truncation error are equally likely, up to the maximum value $max(e)$ (i.e., it is a random variable with uniform distribution) [21]. The assumption of randomness is justifiable only when x_i and w_{ji} are both random. Assuming that the variables x_i and w_{ji} are independent of each other and uniformly distributed in $\left[-2^{B-1}, 2^{B-1}\right]$, we have

$$\mu_{e_{\Delta x_i}} = E\left[\varepsilon_{\Delta x_i}\right] = 0$$

$$\begin{aligned}
\mu_{\varepsilon^{(p)}_{\varphi^*_j}} = E\left[\varepsilon^{(p)}_{\varphi^*_j}\right] &= \sqrt{N_0}2^{\hat{b}-3} \\
E\left[(x_i)^2\right] = E\left[(w_{ji})^2\right] &= \frac{2^{2B}}{12} \\
E\left[\left(\varepsilon^{(p)}_{\varphi^*_j}\right)^2\right] &= \frac{N_0 2^{2\hat{b}-4}}{3} \\
\sigma^2_{\varepsilon^{(p)}_{\varphi^*_j}} &= \frac{N_0 2^{2\hat{b}-6}}{3}
\end{aligned} \tag{36}$$

where μ and σ^2 represent the mean and variance, respectively. The distribution of the practical error of the hidden node j with *high* confidence is uniform in the range, $\mathcal{H}$, may be given as [15],

$$\begin{aligned}
\mathcal{H}_{\varepsilon^{(p)}_{\varphi^*_j}} &\overset{\text{def}}{=} \left[\mu_{\varepsilon^{(p)}_{\varphi^*_j}} \pm \sqrt{3\sigma^2_{\varepsilon^{(p)}_{\varphi^*_j}}}\right] \\
&= \left[\sqrt{N_0}2^{\hat{b}-3} \pm \sqrt{\frac{3N_0 2^{2\hat{b}-6}}{3}}\right] \\
&= \left[0\ ,\ \frac{\sqrt{N_0}}{4}2^{\hat{b}}\right].
\end{aligned} \tag{37}$$

In the incremental communication method the net^*_j is computed without imposing any precision restriction on the multiply-add operations. Moreover, we assume that the accumulator used to store net^*_j is large enough to hold a full precision value. Therefore, the coefficient ϱ_j, which is an indication of the nonlinear effect of the propagated error of the node j can be determined by the following ratio,

$$\begin{aligned}
\varrho_j \overset{\text{def}}{=} \frac{E\left[\left(\varepsilon^{(p)}_{\varphi^*_j}\right)^2\right]}{E\left[\left(net^*_j\right)^2\right]} &= \frac{\frac{N_0 2^{2\hat{b}-4}}{3}}{\frac{2^{2B}}{12}} \\
&= \frac{N_0}{4}2^{-2b}.
\end{aligned} \tag{38}$$

Since the denominator of Equation 38 increases exponentially, the error ratio drops sharply towards zero. Figure 3 depicts the behavior of ϱ_j as a function of N_0 and b. The partial derivatives of ϱ_j with respect to b and N_0 are given as

$$\frac{\partial \varrho_j}{\partial b} = \frac{-(\ln 2)N_0}{2}2^{-2b} = 0.14\varrho_j\ ,$$

$$\frac{\partial \varrho_j}{\partial N_0} = \frac{2^{-2b}}{4}.$$

Therefore, with precision b, the slope of ϱ_j is negative and proportional to itself. The partial derivative with respect to b shows that ϱ_j decreases very fast as b increases.

From Equation 38, the required precision b for fixed N_0 can be approximated as follows,

$$b = \frac{1}{2}\log_2 N_0 - \left(\frac{1}{2}\log_2 \varrho_j + 1\right). \tag{39}$$

Since the factor $\log_2 N_0$ grows rather slowly as we increase N_0, the precision b also grows slowly with an increase in the size of the network. Holt and Hwang report a similar finding in their paper [14].

The net input of node k in the output layer that receives incremental activation values from all hidden nodes is calculated as

$$net_k^* = \sum_j w_{kj}\left(h_j^*\right), \tag{40}$$

where, $h_j^* = (h_j + \varepsilon_{\varphi(net_j^*)}) + \varepsilon_{\Delta(h_j+\varepsilon_{\varphi(net_j^*)})}$. For simplicity let us denote

$$\varepsilon_{\Delta\overline{h}_j} \equiv \varepsilon_{\Delta(h_j+\varepsilon_{\varphi(net_j^*)})}. \tag{41}$$

Therefore, from Equations 40 and 41, the total error received by the node k and the output of this node are computed as follows

$$\varepsilon_{net_k} = \sum_j w_{kj}\left(\varepsilon_{\Delta\overline{h}_j} + \varepsilon_{\varphi(net_j^*)}\right) \tag{42}$$

$$y_k^* = \varphi\left(net_k^*\right) = f\left(\sum_j w_{kj}\overline{h}_j^*\right) \tag{43}$$

The effect of changes in the hidden layer activation vector on the output of the node k is determined by using the partial derivatives,

$$\begin{aligned}\varepsilon_{y_k} &= \sum_j\left(\varepsilon_{\varphi_j} + \varepsilon_{\Delta h_j}\right)\frac{\partial y_k}{\partial h_j}\\ &= \sum_j\left(\varepsilon_{\varphi_j}\right)\frac{\partial y_k}{\partial h_j} + \sum_j\left(\varepsilon_{\Delta h_j}\right)\frac{\partial y_k}{\partial h_j}\end{aligned}$$

$$\begin{aligned} &= \sum_j \left(\sum_i (\varepsilon_{\Delta x_i}) \frac{\partial h_j}{\partial x_i} \right) \frac{\partial y_k}{\partial h_j} + \sum_j \left(\varepsilon_{\Delta h_j} \right) \frac{\partial y_k}{\partial h_j} \\ &= \sum_i (\varepsilon_{\Delta x_i}) \frac{\partial y_k}{\partial x_i} + \sum_j \left(\varepsilon_{\Delta h_j} \right) \frac{\partial y_k}{\partial h_j} \qquad (44) \end{aligned}$$

To extend the error calculation of the forward pass to the networks with more than one hidden layer, let y_k^L represent the activation value of the node k in the output layer. substituting the results obtained above into Equation 21, we get

$$\varepsilon_{y_k^L} \cong \sum_{i=1}^{N_0} \varepsilon_{x_i} \frac{\partial y_k^L}{\partial x_i} + \sum_{\ell=2}^{L} \sum_{j=1}^{N_\ell} \varepsilon_{y_j} \prod_{r=\ell+1}^{L} \frac{\partial y_r}{\partial y_{r-1}}, \qquad (45)$$

where N_ℓ is the number of nodes in layer ℓ. Again, we assume that all arithmetic operations (addition, subtraction, and inner product operations) shown in the Figure 2 can be carried out without generating any error.

From Equations 37 and refchp6:e525, The practical error with the net input of node k at step n is given as,

$$\begin{aligned} \varepsilon_{net_k^*}^{(p)} &\le \sum_j 2^{B-1} \left(2^{-b} + \frac{\sqrt{N_0}}{4} 2^{\hat{b}} \right) \\ &\le \frac{\sqrt{N_1}}{2} 2^{\hat{b}} \left(1 + \frac{\sqrt{N_0}}{4} 2^{B} \right). \qquad (46) \end{aligned}$$

When b bits are used to send the output of hidden nodes to the next (in this case output) layer, it is statistically safe to assume that the limited precision error with the output of each hidden node is not exceeding 2^{-b}. This assumption is particularly valid in the case of the incremental communication method where all operations inside a node are carried out in full precision. Therefore, assuming the logistic sigmoid activation function and using the Equation 34 we find that, the practical limited precision error of the output node k at step n is uniformly distributed in the range $\left[0\ ,\ \frac{\sqrt{N_1}}{4} 2^{\hat{b}}\right]$, where N_1 is the number of hidden nodes sending their output to the output node k. The expected practical error is given as

$$E\left[\left(\varepsilon_{\varphi_k^*}^{(p)} \right)^2 \right] = \frac{(\frac{\sqrt{N_1}}{4} 2^{\hat{b}})^2}{3} = \frac{N_1}{48} 2^{2\hat{b}} = \frac{N_1}{48} \frac{2^{2B}}{2^{2b}} \qquad (47)$$

Since the denominator, 2^{2b}, in Equation 47 increases very fast, it is seen that the expected limited precision error decreases sharply as b increases.

The expected output value of the node k with the conventional communication method, assuming uniform distribution, is given as

$$E\left[(y_k)^2\right] = \frac{2^{2B}}{12}.$$

Thus, the relative practical limited precision error is given as

$$\begin{aligned}\tilde{\varepsilon}_k \overset{def}{=} \hat{\varepsilon}^{(p)}_{\varphi^*_k} &\cong \frac{E\left[\left(\varepsilon^{(p)}_{\varphi^*_k}\right)^2\right]}{E\left[(y_k)^2\right]} \\ &\cong \frac{N_1}{2^{2(b+1)}} \qquad \text{for } 0 < b \leq B. \end{aligned} \tag{48}$$

Figure 4 shows the relative practical limited precision error of an output node as function of both N_1 and b. This relative error may be interpreted from Equation 48 as

$$\tilde{\varepsilon}_k \longrightarrow \begin{cases} 0 & \text{when } b \to B \\ \frac{N_1}{4} & \text{when } b \to 0 \end{cases}$$

A network of 100 hidden nodes ($N_1 = 100$), with 8-bit precision has less than one percent error with its output nodes. This is generally in agreement with our simulation results (see Figures 9 and 15).

B. Error Analysis of Backward Pass

The backward pass starts at the output layer by passing the error signals backward, layer by layer, and computing the local gradient (learning signal, $\delta = \frac{\partial E}{\partial net}$) for a node. The error signal of the node k using the sigmoid activation function is calculated as,

$$\begin{aligned}\delta_k &= [d_k - y_k]\, \varphi'(net_k) \\ &= y_k(1-y_k)\,[d_k - y_k], \end{aligned} \tag{49}$$

where d_k represents the desired output of the output node k. The effect of the changes in y_k on the magnitude of δ_k is determined by the following partial derivative,

$$\begin{aligned}\frac{\partial \delta_k}{\partial y_k} &= -\varphi'_k + (d_k - y_k)\frac{\partial \varphi'_k}{\partial y_k} \\ &= -y_k(1-y_k) + (d_k - y_k)(1 - 2y_k). \end{aligned} \tag{50}$$

Therefore,

$$\begin{aligned}\varepsilon_{\delta_k^*} &= \frac{\partial \delta_k}{\partial y_k}\varepsilon_{y_k} \\ &= \left[-y_k\left(1-y_k\right)+\left(d_k-y_k\right)\left(1-2y_k\right)\right]\varepsilon_{y_k}. \end{aligned} \tag{51}$$

Since the desired outputs for the hidden nodes are not available, the learning signal of the hidden node j is calculated as,

$$\delta_j^* = \varphi_j' \sum_{k=1}^{N_2} \delta_k^* w_{jk}, \tag{52}$$

where N_2 is the number of nodes in the output layer.

The parameters that are involved in the calculation of δ_j are all contaminated by limited precision error during the forward pass, therefore, the limited precision error of δ_j can be expressed as,

$$\varepsilon_{\delta_j} = \varphi_j' \sum_{k=1}^{N_2} \varepsilon_{\delta_k^*} w_{jk} + \varphi_j' \sum_{k=1}^{N_2} \delta_k^* \varepsilon_{w_{jk}}^* + \varphi_j'' \sum_{k=1}^{N_2} \delta_k^* w_{jk}, \tag{53}$$

where φ_j'' represents the second derivative of φ_j.

The limited precision error in δ_k consists of two parts: (a) the genuine limited precision error which is the result of representing the δ_k with reduced precision and (b) the propagated error caused by various operations of the forward pass. The propagated error associated with δ_k is given as

$$\delta_k^* = \left(d_k - \left(y_k + \varepsilon_{y_k}\right)\right)\left(\varphi_k^*\right)' \tag{54}$$

$$\varepsilon_{\delta_k^*} = \varepsilon_{y_k^*}\varphi_k' \cong \varepsilon_k^{(p)}\varphi_k' \tag{55}$$

where $\varphi_k' = (\varphi_k^*)'$ is considered to be error free. From 55, we get

$$\varepsilon_{\delta_k^*}^{(p)} \in \left[0\ ,\ \frac{\sqrt{N_1}}{16}\frac{2^B}{2^b}\right] \tag{56}$$

and

$$E\left[\left(\varepsilon_{\delta_k^*}\right)^2\right] = \frac{N_1}{256}\frac{2^{2B}}{2^{2b}} \tag{57}$$

The weights of the network are adapted using

$$\begin{aligned} w_{kj}(t) &= w_{kj}(t-1) + \eta\delta_k h_j, && \text{for the output layer} && (58)\\ w_{ji}(t) &= w_{ji}(t-1) + \eta\delta_j x_i, && \text{for the hidden layer} && (59)\end{aligned}$$

where η is the learning rate.

The effect of the limited precision incremental communication method on the weights of the network can be expressed as,

$$\begin{aligned} \varepsilon_{\Delta w_{kj}} &= \eta\delta_k^* \varepsilon_{h_j^*} + \eta h_j^* \varepsilon_{\delta_k^*}, && \text{for the output layer} && (60)\\ \varepsilon_{\Delta w_{ji}} &= \eta\delta_j^* \varepsilon_{x_i} + \eta x_i \varepsilon_{\delta_j^*}, && \text{for the hidden layer} && (61)\end{aligned}$$

Thus, the practical relative error with the incoming delta weights of the output nodes is defined as

$$\tilde{\varepsilon}^{(p)}_{\Delta w_{kj}} = \frac{\varepsilon^{(p)}_{h_j^*}}{h_j} + \frac{\varepsilon^{(p)}_{\delta_k^*}}{\delta_k} \qquad (62)$$

and similarly,

$$\tilde{\varepsilon}^{(p)}_{\Delta w_{ji}} = \frac{\varepsilon^{(p)}_{x_i}}{x_i} + \frac{\varepsilon^{(p)}_{\delta_j^*}}{\delta_j} \qquad (63)$$

The expected values of $\tilde{\varepsilon}^{(p)}_{\Delta w_{kj}}$ can be approximated as follows

$$\begin{aligned} E\left[\left(\tilde{\varepsilon}^{(p)}_{\Delta w_{kj}}\right)^2\right] &= \frac{E\left[\left(\varepsilon^{(p)}_{h_j^*}\right)^2\right]}{E\left[(h_j)^2\right]} + \frac{E\left[\left(\varepsilon^{(p)}_{\delta_k^*}\right)^2\right]}{E\left[(\delta_k)^2\right]}\\ &= \frac{(N_1/48)\,2^{2\hat{b}}}{2^{2B}/12} + \frac{(N_1/256)\,2^{2\hat{b}}}{2^{2B}/12}\\ &= 0.3N_1 2^{-2b} \qquad (64)\end{aligned}$$

Figure 5 shows the behavior of $\tilde{\varepsilon}^{(p)}_{\Delta w_{kj}}$ as a function of N_1, and b. In general, the results of theoretical findings are in agreement with the simulation results.

VI. Simulation Results

The usefulness of the incremental communication method has been tested on various learning problems [7], [9], [10], which involve binary as well as real-valued inputs and outputs. In this section, we have chosen different benchmark problems, to serve as additional examples, with only real-valued inputs and outputs to illustrate the foregoing analysis. We investigate the performance of the limited precision incremental communication method and show the effect of error propagation on the training behavior.

To support the results derived in the previous sections, simulations are performed for two learning problems: (a) Fuzzy XOR problem and (b) Sine function. In both problems, the primary goal is to study the effect of learning problem complexity and network structure on the error propagation and learning abilities of the limited precision incremental communication method. The experiments consist of training the networks while varying the precision of the incremental values of activation, learning signal and delta weight in the fixed- or floating-point incremental communication. The results obtained from imposing various limited precisions are compared with those of the conventional communication method. The evolution of the network's parameters (weights) during learning and the closeness of weight vectors with those of the conventional communication method are examined.

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

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

The second measure of similarity is to measure how closely one vector points in the direction of another vector by calculating the angle (θ) between two weight vectors $\boldsymbol{w}$ and $\boldsymbol{v}$.

$$\cos(\theta) = \frac{\boldsymbol{w} \cdot \boldsymbol{v}}{|\boldsymbol{w}||\boldsymbol{v}|}.$$

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

A. Fuzzy Exclusive-Or (XOR)

This problem is an example of nonlinear separable problem with real-valued inputs and output. The Fuzzy exclusive-or problem is presented to a network consisting of a layer of two input nodes, a layer of one hidden node, and one output node. There are two cross-connections from input nodes to output node. We choose the training examples used in [19] and these are shown in Table I.

The network is trained to generate a small output value when x_1 and x_2 are either both small or large. The network is also trained to generate a large output value when one of the inputs is small and the other one is large. The standard backpropagation algorithm with online-update strategy is used to train this network. The network parameters are set for $\eta = 0.8$, $\lambda = 0.7$, and $\tau = \pm 1$. The nonlinear sigmoid function is used with hidden node as well as output node. The training is considered complete when all training patterns are learned to within 0.1 error.

The convergence behavior of the Fuzzy XOR problem using incremental as well as conventional communication methods for a sample run is depicted in Figure 6. In other words, Figure 6 represents the error for varying precision of fixed- and floating-point representations as a function of the number of epochs. We have chosen a sample run with slightly longer training epochs than the average training epochs in order to examine the effect of large number of iterations on the propagated error. The average number of training epochs for this problem using the above learning parameters is around 90 epochs. It is seen that as the precision of incremental values increases, the number of epochs required for convergence gets closer to the number of epochs required for convergence with conventional communication. Table II summarizes the results of training processes for fixed-and floating-point representations with varying precisions.

The effects of the limited precision manifest themselves in the various stages of the processing, or even before the actual processing begins as is the case of representing the incremental values of the input patterns with limited precision. Training results for conventional and the incremental communication methods are compared over the first 840 presentations of the input patterns. Each trial of the limited precision incremental communication method uses the same initial weights and learning parameters that are used with the conventional communication method. The hidden node has 3 incoming weights (2 weights coming from 2 input nodes and one weight coming from bias node).

Figure 7 gives the plots of the angles and Euclidean distances of the weight vectors of fixed- and floating-point representations for various precisions with the weight vector of the conventional communication. It is seen that after 600 presentations the weight vectors with 6- and 8-bit fixed-point representations get closer to the conventional weight vector. Thus, from this result we can see that the dynamic range of the floating-point representation does not have significant effect in reducing the limited precision errors for some problems. The other reason behind the behavior is the use of a greater number of bits to the right of the binary point (accuracy) in the fixed-point representation. The 6-bit fixed-point representation is implemented using 5 bits fraction whereas 4-bit float has only 4 bits for the fractional part. The 8-bit fixed precision value is represented as 1 bit to the left and and 7 bit to the right of binary point. On the other hand, the 6-bit float only uses 6 bits for the mantissa.

There are 4 incoming weights for output node. Figure 8 shows the angles and Euclidean distances of the incoming weight vector of the output node in the incremental communication method with that of the conventional communication method. The results shown in this figure also confirm the above conclusion that the exponent of floating-point representation does not play a major rule in reducing the limited precision error of this problem. Therefore, 6-bit fixed-point realization of this problem may not suffer from the relatively small precision. This problem can even be trained with as low as 4-bit fixed-point representation without any instability during the training period. However, the network trained with 4-bit fixed-point fails to generate correct answers for all the test patterns. Our experiments shows that the generalization capability of a network is affected below a certain precision level. For this problem it is 4-bit.

There are 7 desired outputs for 7 input examples. The output node generates an output for each presentation and hence a vector of 7 outputs in each epoch. The norm (length) of the output vector of each epoch is used to compare the output vector generated by the incremental communication method with the output vector generated by the conventional communication method. Figure 9 depicts the difference between the norm of the output vector generated by the incremental communication method and the norm of the output vector generated by the conventional communication method. The bars in this figure represent the absolute deviations. This figure also confirms that the 6-bit fixed-point precision with 5 bits of accuracy has closer performance to that of the full precision conventional communication than the 4-bit floating-point representation with 4 bits of accuracy. Therefore, once again we come to the conclusion that the realization of this network with 6-bit fixed-point representation is preferred over the 4-bit floating-point representations. Moreover, the communication cost with 4-bit floating-point representation is nearly twice the communication cost with 6-bit fixed-point representation.

B. The Sine Function

In this problem a network is trained to learn the nonlinear mapping between the input and desired output. A four-layered feedforward network, consisting of 1 input node, 8 nodes in the first hidden layer, 3 nodes in the second hidden layer, and 1 output node, is used. A network of one hidden layer is capable of learning this problem. The main reason that we have chosen a larger network is to examine the effect of error propagation through 4 layers and larger number of nodes.

The input of this problem consists of 41 points of equal parts in the range ($\pm\pi$) and the output is the value of the sine function. The quickprop update strategy is used to train the network. In the experiment the weights were initialized with a uniform random number in the range (-1,1). The nonlinear *tanh* function is used to compute the hidden nodes activity levels, whereas a linear function is used with the output layer to calculate the network's output. The termination criterion used was that either the sum of the squared error is below 0.01 or that the number of epochs exceeds 490 (20090 presentations), which ever occurs first. For conventional communication as well as the incremental communication, we have set other network parameters as $\eta = 0.25$, $\mu = 1.75$. The weight decay is set to 0.00001.

Figure 10 represents the error for varying precision of fixed- and floating-point representation as a function of the number of epochs. It is seen that for fixed number of epochs (i.e., 490 epochs) the 12-bit fixed-point exhibits better performance than 4-bit floating-point. This is expected because with 12-bit fixed-point, 3 bits are used to the left of binary point and 9 bits to the right of binary point, whereas with 4-bit floating point only 4 bits are used with the mantissa. Because of the range of the values that are used with this problem, the 8-bit exponent of the floating-point representation does not play a significant rule in preserving the magnitude of the values involved. Table III summarizes the training results.

There are 8 outgoing weights from the input unit. Figure 11 shows the evolutions of weights $w1$, $w3$, $w6$, and $w7$ for different precision levels during the training phase. It is seen that the weights with 10- and 12-bit fixed-point precisions are evolving in close vicinity. The floating-point precisions are exhibiting almost the same behavior as that of the fixed-point. The weight $w7$ is not evolving in close vicinity with 4- and 8-bit floating-point precisions. Thus, no definite conclusion can be drawn from the weights evolutions. However, it is possible to infer that, as the precision of the incremental values increases the corresponding weight evolutions getting closer to that of the full precision conventional method. In Figure 11 we can see this from the weight evolutions with 8-bit floating-point precision.

Figure 12 shows the angles and Euclidean distances between the outgoing weight vector of the input node with varying precision incremental communication and the corresponding weight vector of the input node with conventional communication. This is a more appropriate way of representing the closeness properties of the weights. It is seen that the weight adjustments are smaller with incremental communication method, however, the weights with incremental and conventional communication methods are generally in the same directions. It may be noted that the weight adjustment rule corrects the weights in the same direction as that of the conventional method. The size, or simply length, of the weight correction vector is the main difference between the conventional and incremental methods.

In Figure 12 we can see that the weight vector of 10-bit fixed-point representation is in closer vicinity (angle and Euclidean distance) to that of conventional weight vector. It is known that with the gradient descent optimization method [23], all nearby negative gradient paths lead to the same local minimum. Thus, it is not necessary to be very accurate and follow the negative gradient exactly; we can rather use smaller precision for the parameters and still expect to get to the local minima.

There are four nodes in layer 3 of the network. Figure 13 shows the evolution of weights of these nodes. The output node has a four elements incoming weight vector. Figure 14 depicts the angles (closenesses) of weight vectors of various precision levels with that of the conventional method. It is seen that the 10- and 12-bit fixed-point precisions weight vectors are in 2nd quad where as the conventional and floating-point representation weight vectors are in the 1st quad. This shows again another property of gradient descent method with which one can approach the optimum point from various angles.

Figures 15 depicts the influence of the incremental communication with 4-bit floating-point and 12-bit fixed-point precisions for incremental values on the output of the network. The bars in this figure show the deviations of the outputs of the network using the incremental communication and the outputs of the conventional communication. This figure also shows the relative deviations of the incremental outputs from the conventional outputs for 41 input patterns. The plot of 4-bit float shows that still 2 patterns remain to be learned, whereas the plot of 12-bit fixed indicates that only one pattern is to learned. Thus, noticing that the 4-bit float and 12-bit fixed have the same communication costs (i.e., both use 12 bits) the appropriate choice would be 12-bit fixed-point representation.

VII. Conclusions

Artificial neural networks can be implemented by incorporating the incremental communication method. The incremental communication method is aimed at reducing the communication complexity of artificial neural networks by limiting the node's input/output bandwidth requirements. In this paper, we have used mathematical models to investigate the effects of small perturbations in the input(s)/output of the internal nodes. These small perturbations arise due to the reduced precision representation of incremental values that are communicated between nodes in the incremental communication method. Since the limited precision arithmetic is computationally error prone, we have derived the required formulas for the calculation of the propagated errors in the chain of linear and nonlinear nodes. It has been shown that the rate of growth of the propagation error is directly proportional to the number of operations involved.

The nonlinear effect of representing the input of the sigmoid function with reduced precision was analyzed. It has been shown that the propagation error caused by the sigmoid function is always less than the limited precision error generated by imposing limited precision on the input value of the function.

The effect of representing the incremental input/output values with reduced precision on the commonly used error backpropagation training algorithm of the multilayer neural networks was analyzed. The mathematical derivations of the errors caused by the limited precision incremental communication have been supported by analyzing the results of our simulation on two learning problems. The results obtained from the simulations, in general, have been found to be in agreement with the results of theoretical analysis.

In conclusion, the incorporation of the incremental communication method in the multilayer perceptrons leads to convergence without enforcing extra instability and excessive increase in learning time. The results of the simulation studies given in this paper and our earlier work[8], [9], [10] clearly indicate substantial savings in communication costs for implementation of multilayer perceptrons on parallel computers; the incremental communication method is certainly attractive for VLSI realizations.

REFERENCES

[1] M. Andrews and R. Fitch, "Finite Word Length Arithmetic Computational Error Effects On The LMS Adaptive Weights," *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing,* pages 628-631, 1977.

[2] T. Baker. "Implementation Limits for Artificial Neural Networks," *Master Thesis,* Oregon Graduate Institute, Dept. of Computer Science and Eng., Feb. 1990.

[3] C. Caraiscos and B. Liu, "A Roundoff Error Analysis of the LMS Adaptive Algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing* **ASSP-32**(1): 34-41, Feb. 1984.

[4] J. M. Cioffi, "Limited-Precision Effects in Adaptive Filtering," *IEEE Transactions on Circuits and Systems* **CAS-34**(7): 821-833, 1987.

[5] D. D. Cavilia, M. Valle, and G. M. Bisio, "Effects of Weight Discretization on the Back Propagation Learning Method: Algorithm Design and Hardware Realization," Proc. of the International Joint Conference on Neural Networks, San Diego, CA, Vol. 2, pp. 631-637, 1990.

[6] B. P. Demidovich and I. A. Maron *Computational Mathematics,* Translated from the Russian by G. Yankovsky, Mir publishers, Moscow, Russia, 1987.

[7] A. A. Ghorbani and V. C. Bhavsar, "Incremental Inter-node Communication in Artificial Neural Networks," In *Proc. of the Second International Conference on Automation, Robotics and Computer Vision (ICARCV '92)* 1: INV-7.5.1-7.5.5, Photoplates Pte Ltd., Singapore, Sept. 1992.

[8] A. A. Ghorbani and V. C. Bhavsar, "Artificial Neural Networks With Incremental Communication on Parallel Computers," Proc. of the 5TH UNB Artificial Intelligence Symposium, Fredericton, Canada, pp. 15-28, The University of New Brunswick Press, Fredericton, NB, Aug., 1993.

[9] A. A. Ghorbani and V. C. Bhavsar, "Training Artificial Neural Networks Using Variable Precision Incremental Communication," Proc. of the IEEE World Congress On Computational Intelligence (ICNN'94), Vol. 3, pp. 1409-1414, Orlando, Florida, June 1994.

[10] A. A. Ghorbani and V. C. Bhavsar, "Incremental Communication for Multilayer Neural Networks," IEEE Transactions on Neural Networks, Vol. 6, No. 6, pp. 1375-1385, 1995.

[11] R. D. Gitlin, J. E. Mazo, and M. G. Taylor, "On the Design of Gradient Algorithms for Digitally Implemented Adaptive Filters," *IEEE Transactions on Circuit Theory* **CT-20**(2): 125-136, 1973.

[12] M. Hoehfeld and S. E. Fahlman, "Learning with limited numerical precision using the cascade-correlation algorithm," IEEE Transactions on Neural Networks, Vol. 3, No. 4, pp. 602-611, July 1992.

[13] P. W. Hollis, J. S. Harper, and J. J. Paulos, "The effects of precision constraints in a backpropagation learning network," Neural Computation, Vol. 2, pp. 363-373, 1990.

[14] J. L. Holt and J. N. Hwang, "Finite Precision Error Analysis of Neural Network Hardware Implementations," *IEEE Transactions on Neural Networks* **42**(3): 281-290, 1993.

[15] R. A. Johnson and G. Bhattacharyya, *Statistics: Principles and Methods,* John Wiley & Sons, New York, NY, USA, 1987.

[16] R. Meir and J. F. Fontanari, "On Learning Noisy Threshold Functions with Finite Precision Weights," *Proc. of Sixth Workshop on Computational Learning Theory* COLT'92, pages 280-286, PA, USA, July 1992.

[17] K. Nakayama, S. Inomata, and Y. Takeuchi, "A digital multilayer neural network with limited binary expressions," Proc. of the International Joint Conference on Neural Networks, San Diego, CA., Vol. 2, pp. 578-592, 1990.

[18] S. M. Pizer and V. L. Wallace, *To Compute Numerically, Concepts and Strategies*, Little, Brown and Co., Boston, MA, USA, 1983.

[19] S. G. Romaniok and L. O. Hall, Divide and Conquer Neural Networks, *Neural Networks* 6(8): 1105-1116, 1993.

[20] G. Strang, *Calculus*, Wellesley-Cambridge Press, Wellesley, MA, USA, 1992.

[21] J. R. Taylor, *An Introduction to Error Analysis*, University Science Books, Mill Valley, CA, USA, 1982.

[22] D. J. Wilde and C. S. Beightler, *Foundations of optimization*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1967.

[23] J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, Prentice-Hall Inc., Englewood Cliffs, NJ, USA, 1963.

[24] Y. Xie and M. Jabri, "Training Algorithms for Limited Precision Feedforward Neural Networks," *Technical Rep. No. 1991-8-3*, Dept. of Electerical Engineering, The University of Sydney, Autralia, 1991.

[25] Y. Xie and M. Jabri. "Analysis of the effects of quantization in multilayer neural networks using a statistical model," IEEE Transactions on Neural Networks, Vol. 3, No. 2, pp. 334-338, 1992.

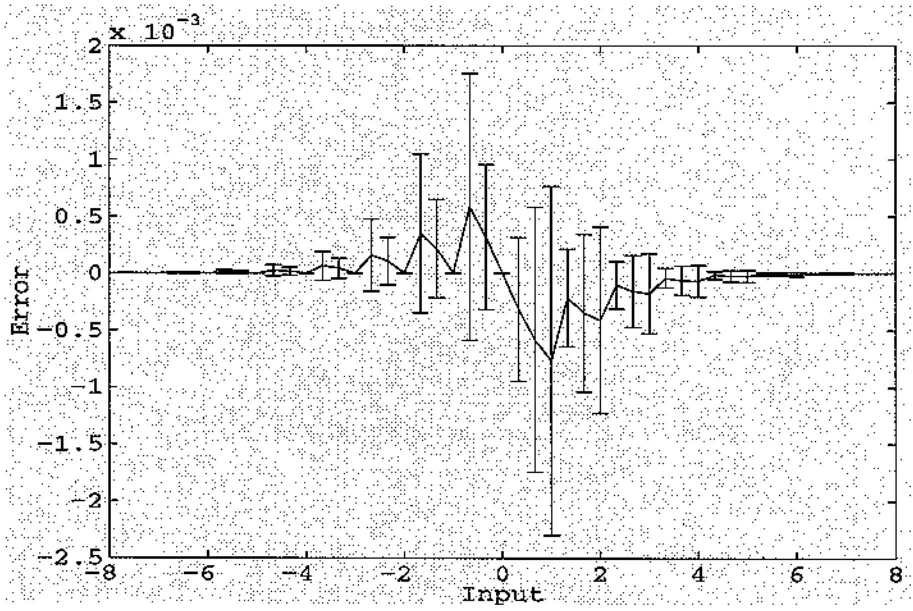


Fig. 1. The actual and analytical errorbar with 8-bit precision fixed-point input for the logistic function.

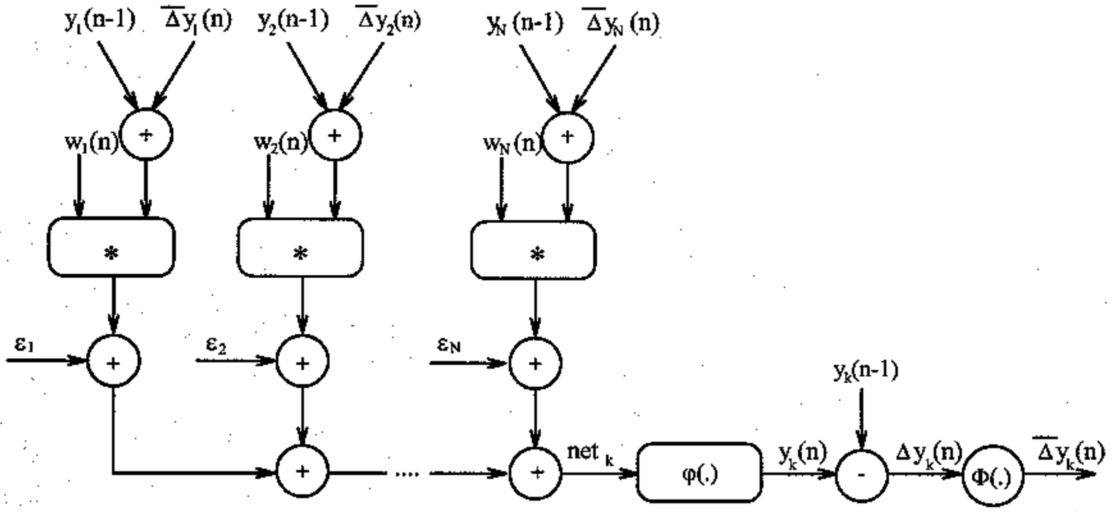


Fig. 2. Forward pass operations using incremental communication method.

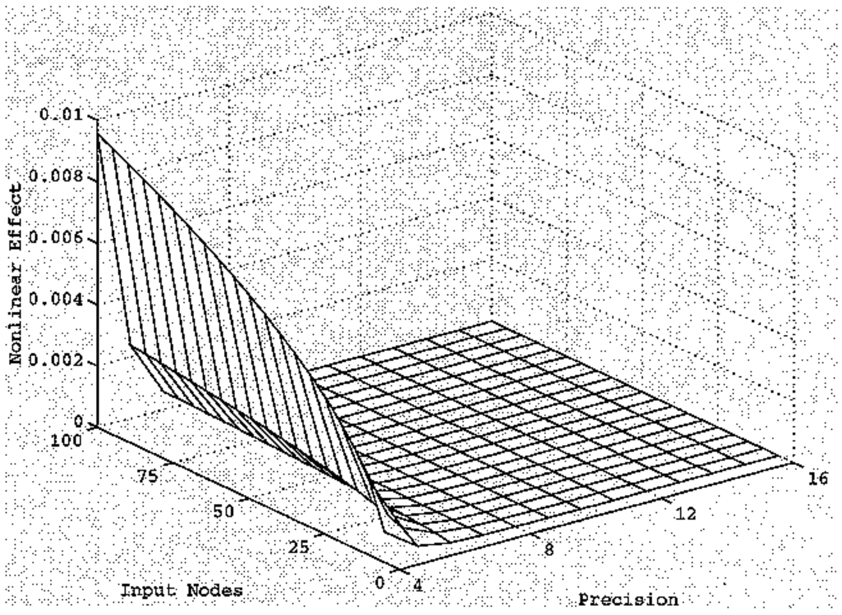


Fig. 3. The nonlinear effect of the propagated limited precision error on the output of a hidden node.

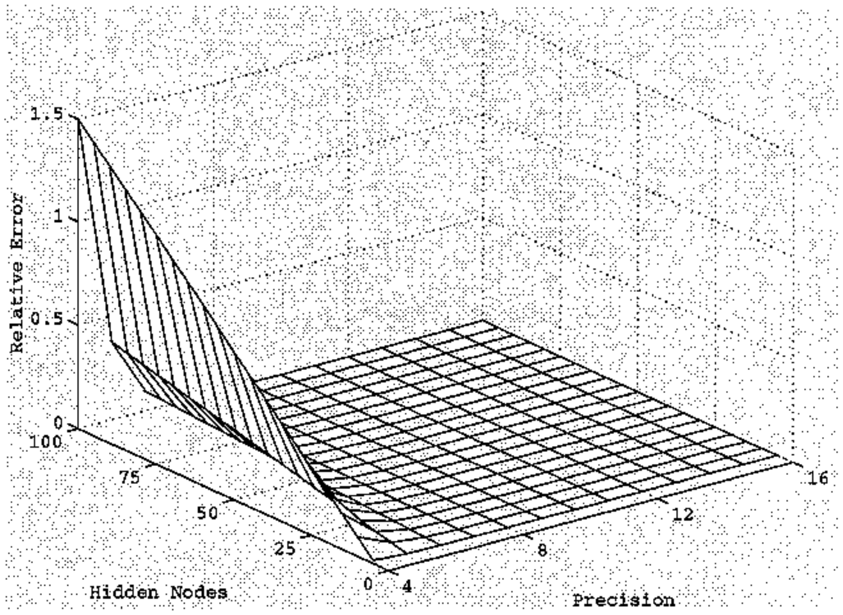


Fig. 4. The relative limited precision error of an output node.

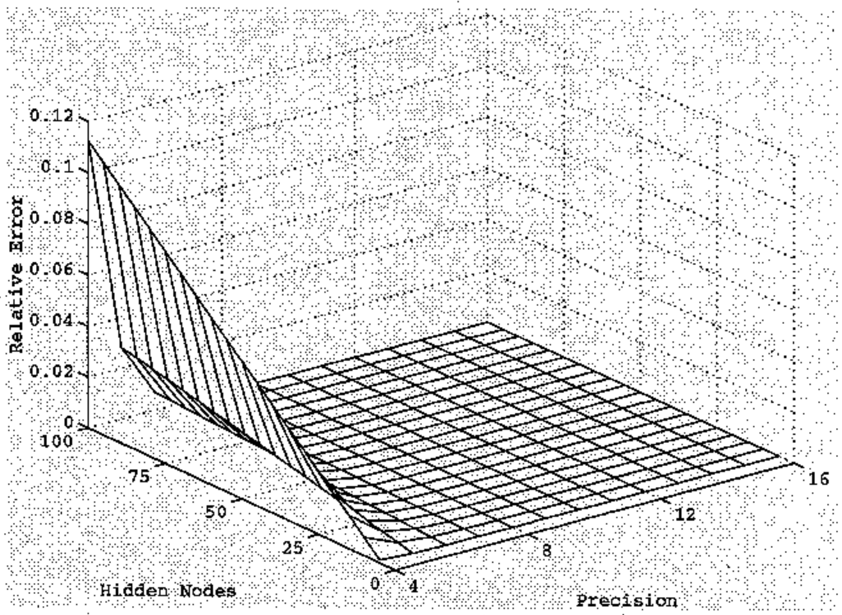


Fig. 5. The expected relative error with the incoming weights of the output nodes.

TABLE I

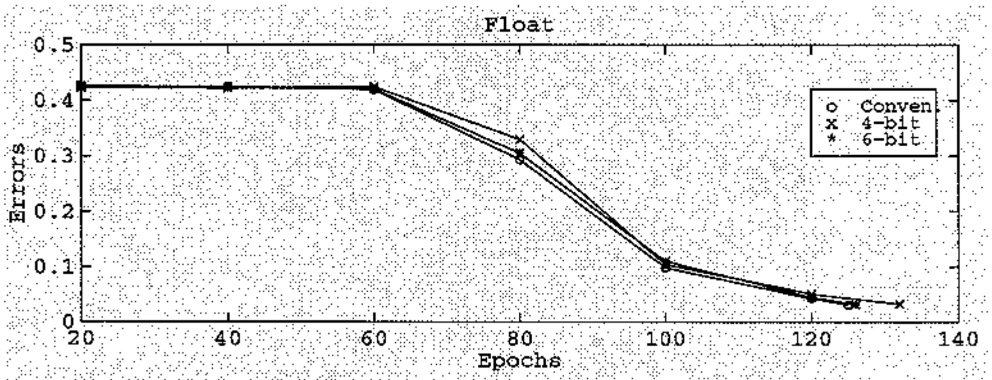
FUZZY EXCLUSIVE-OR TRAINING EXAMPLES.

	x_1	x_2	y
1	0.81	0.81	0.10
2	0.81	0.20	0.91
3	0.20	0.81	0.91
4	0.20	0.20	0.10
5	0.05	0.05	0.05
6	0.00	0.95	0.96
7	0.95	0.05	0.95

TABLE II

Training results for the Fuzzy XOR problem using conventional and the incremental communications.

	After 100 Epochs		Convergence	
Representation	Success	Error	Epochs	Error
Conventional	57.14%	0.09816	125	0.03077
6-bit float	57.14%	0.10974	126	0.03130
4-bit float	42.86%	0.10409	132	0.03202
8-bit fixed	57.14%	0.09971	125	0.03106
6-bit fixed	57.14%	0.10716	126	0.03145



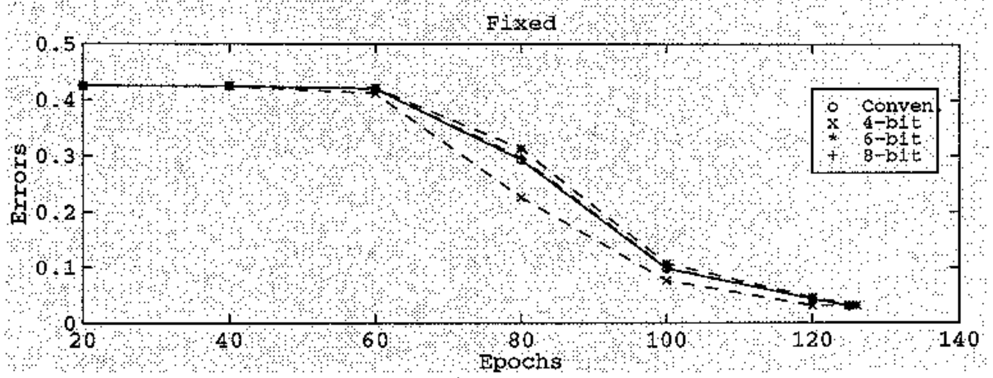


Fig. 6. Errors versus number of epochs with the fixed- and floating- point incremental communications for Fuzzy XOR problem.

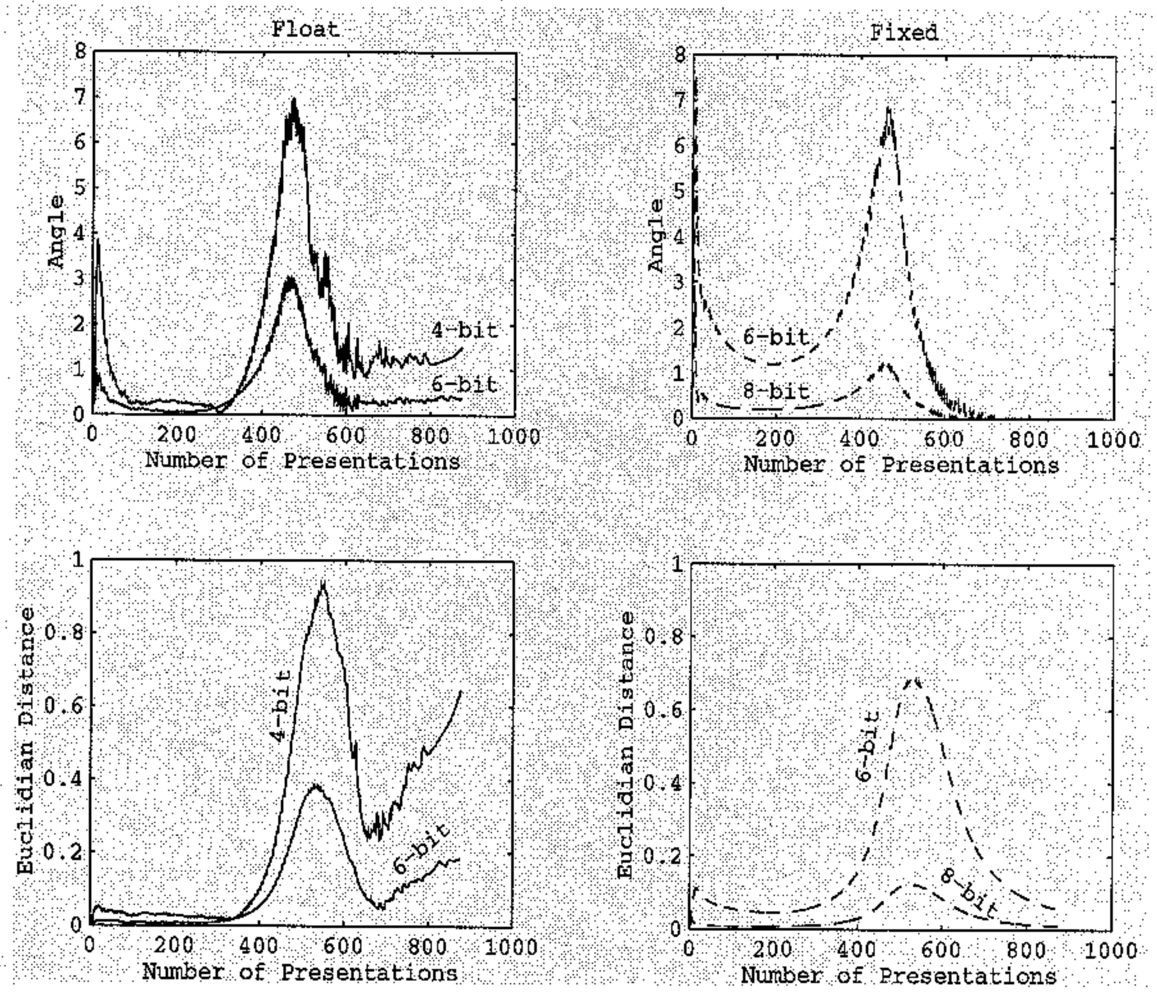


Fig. 7. Angle and Euclidean distance of the hidden node's incoming weight vector with fixed- and floating-point incremental communications for Fuzzy Exclusive-Or problem.

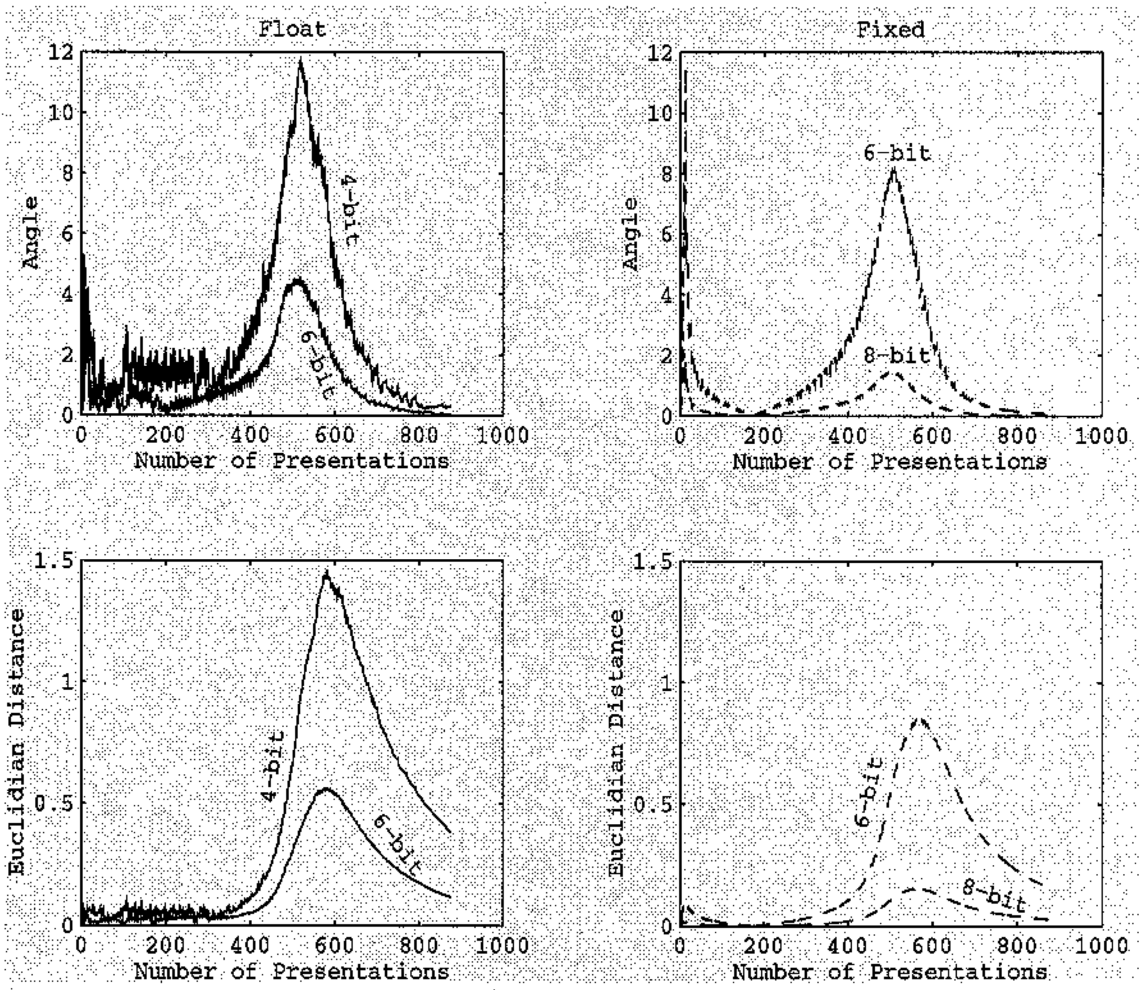


Fig. 8. Angle and Euclidean distance of output node incoming weight vector with fixed- and floating-point incremental communications for Fuzzy Exclusive-Or problem.

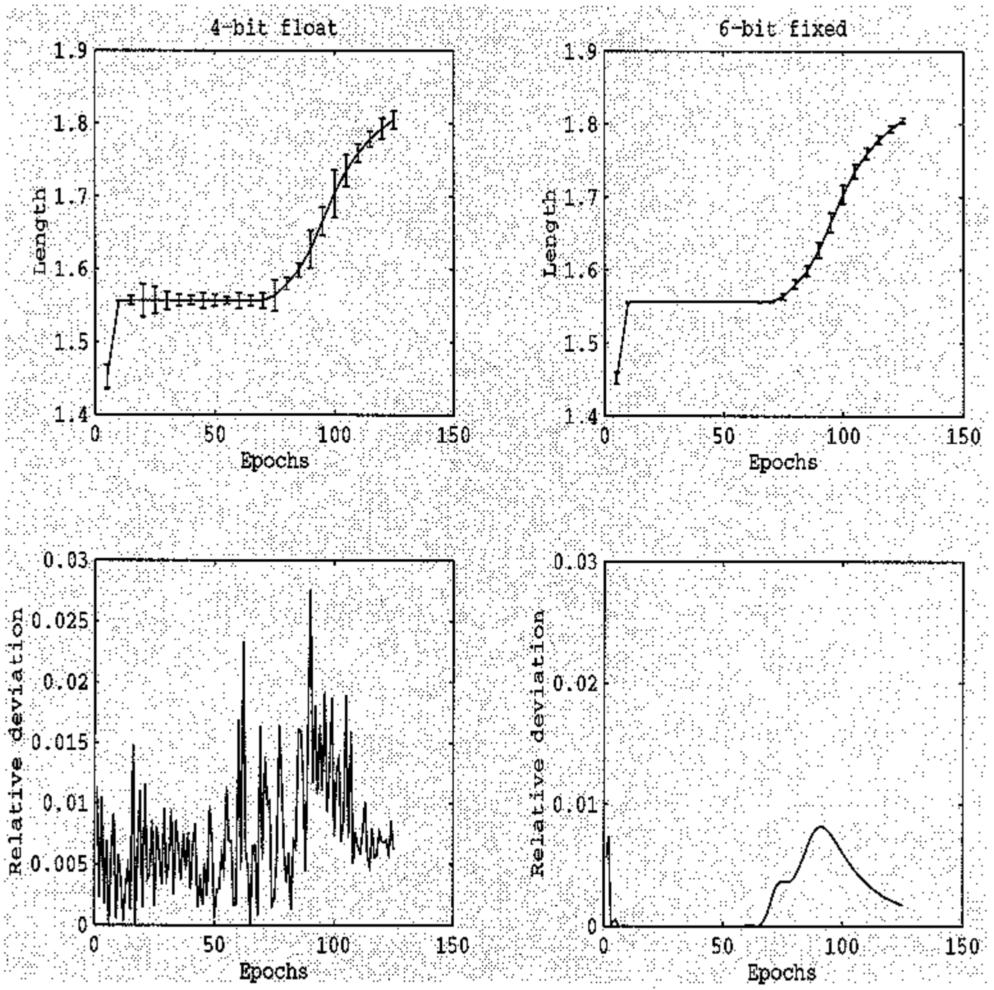


Fig. 9. Absolute and relative deviations of output vector norms with fixed- and floating- point incremental communications for Fuzzy Exclusive-Or problem.

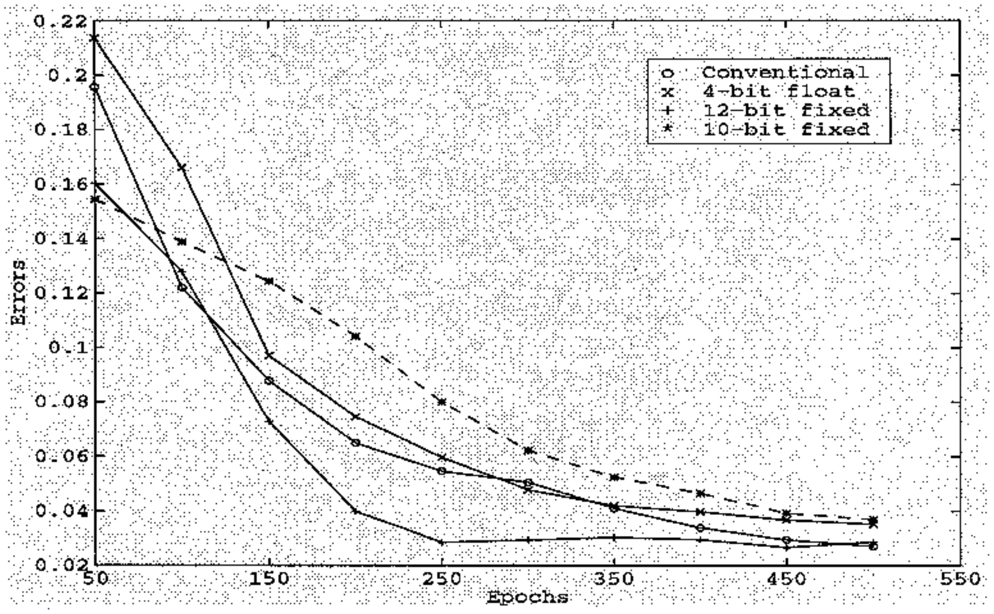


Fig. 10. Errors with the fixed- and floating- point communication for Sine function

TABLE III

TRAINING RESULTS FOR THE SINE FUNCTION USING CONVENTIONAL AND INCREMENTAL COMMUNICATIONS.

	After 400 Epochs		After 490 Epochs	
Representation	Success	Error	Success	Error
Conventional	97.56%	0.03386	100.00%	0.02698
8-bit float	90.24%	0.04386	92.68%	0.03800
4-bit float	90.24%	0.03984	95.12%	0.03514
12-bit fixed	95.12%	0.02958	97.56%	0.02865
10-bit fixed	92.68%	0.04662	92.68%	0.03690
8-bit fixed	24.39%	0.23799	24.39%	0.23799

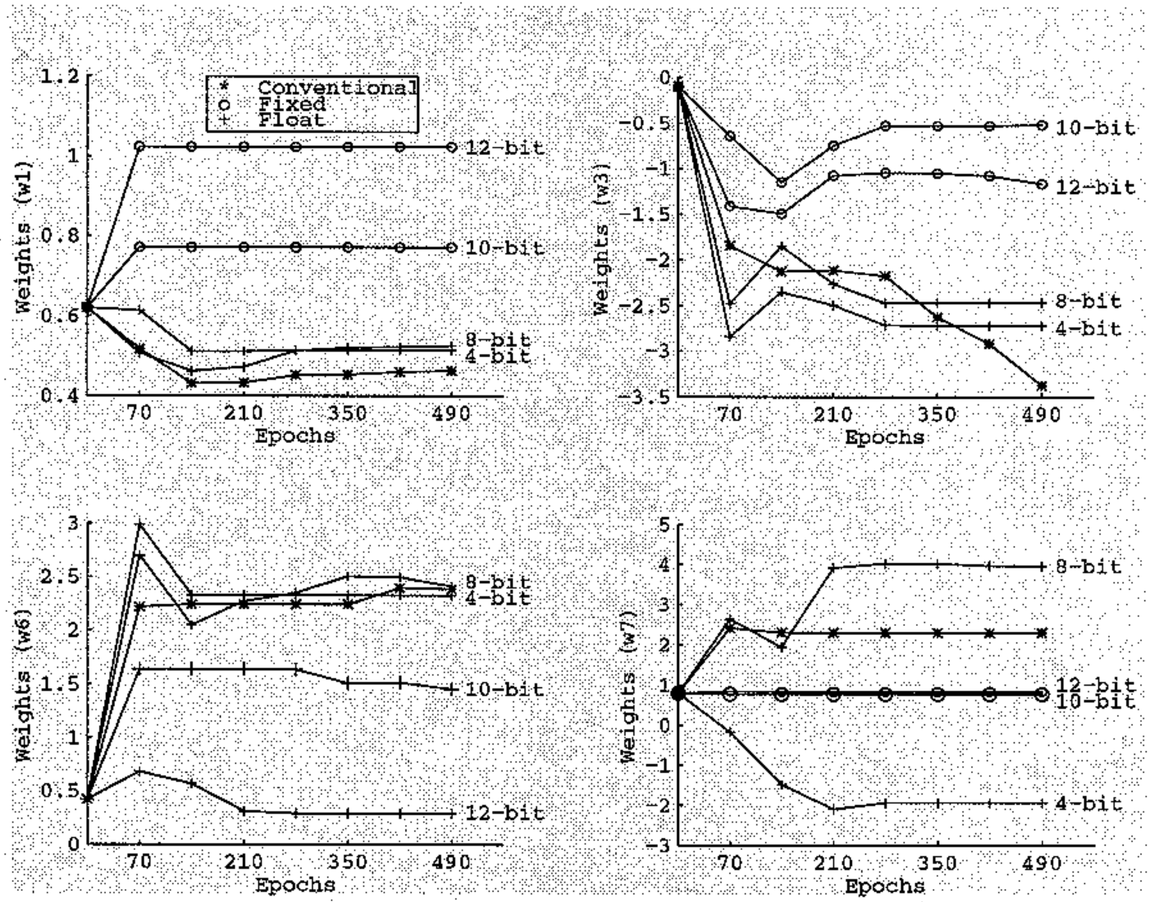


Fig. 11. Input node outgoing weights evolutions with the fixed- and floating- point communications for the Sine function

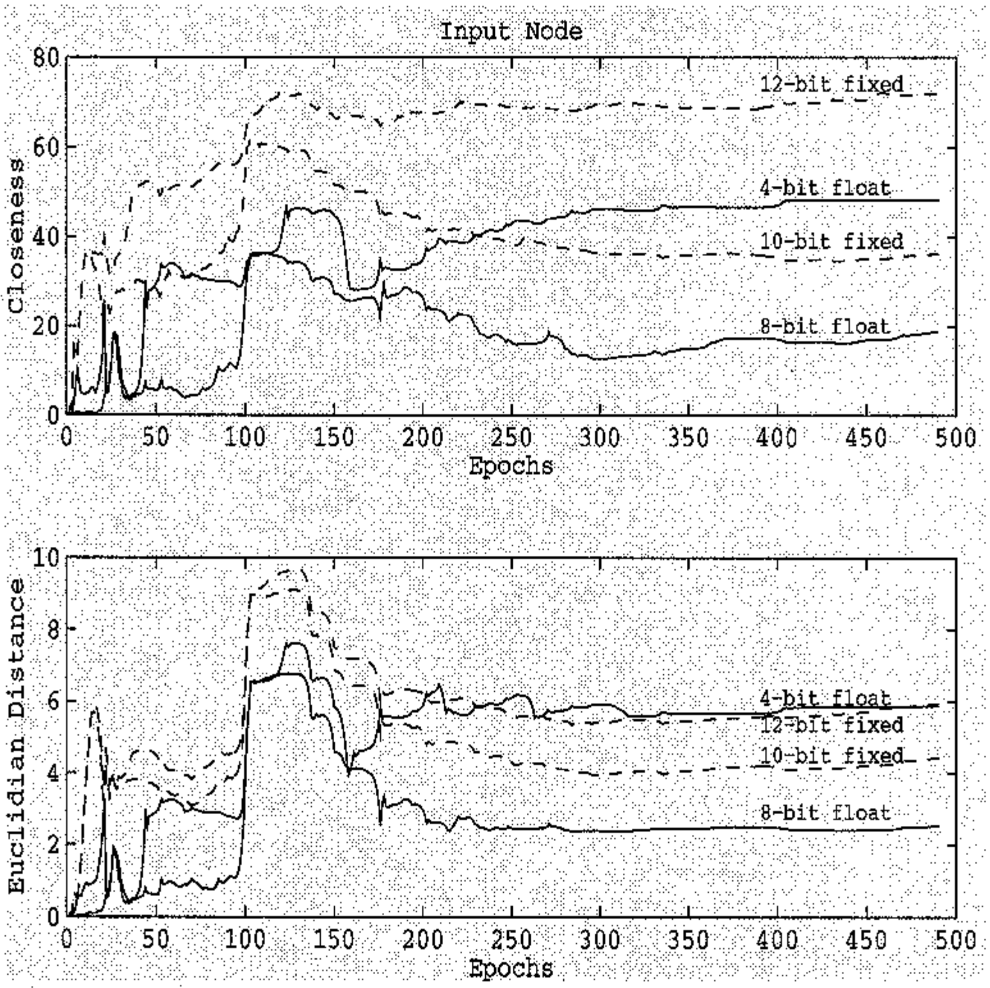


Fig. 12. Angles and Euclidean distances of input node outgoing weight vector with the fixed- and floating-point communications for the Sine function

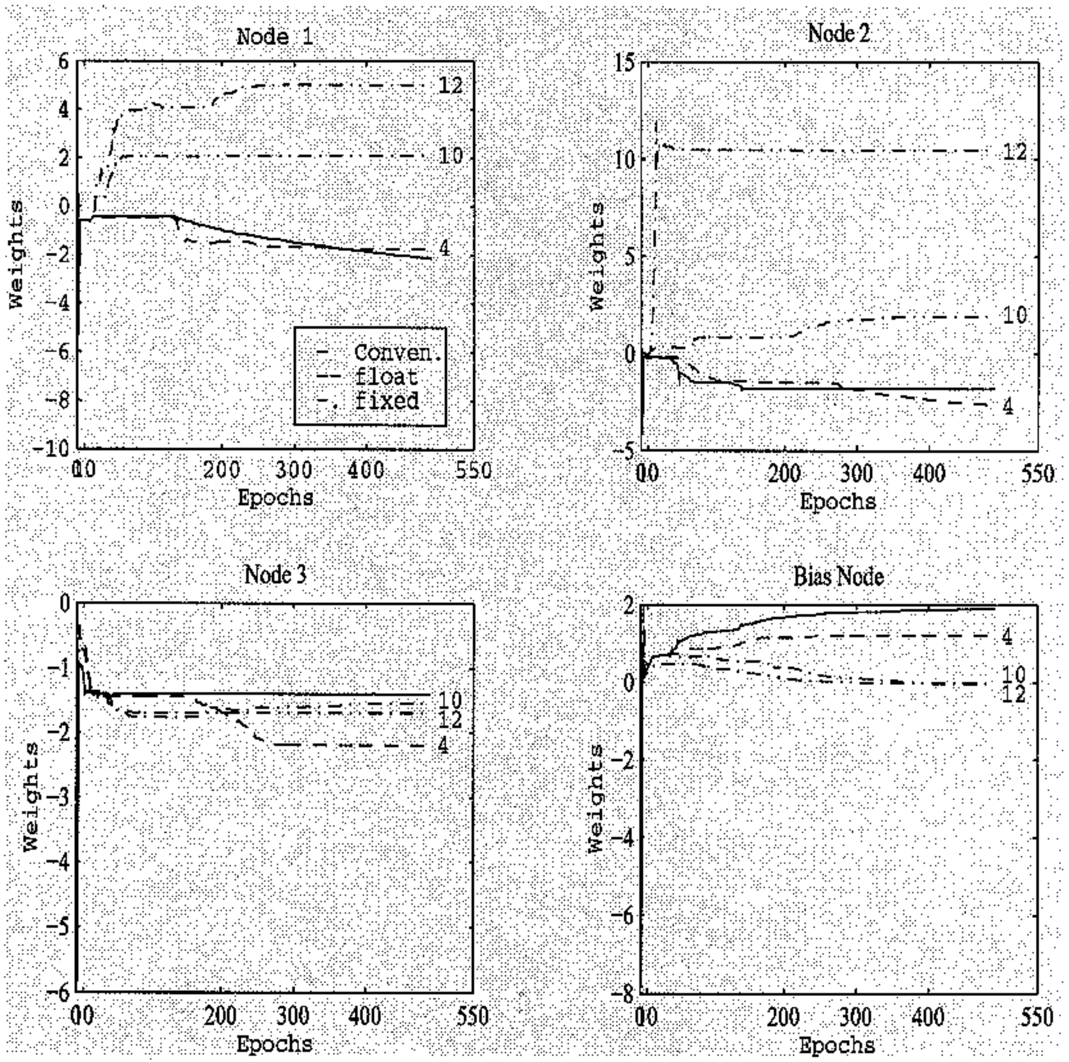


Fig. 13. Output node incoming weights evolutions with the fixed- and floating- point communications for the Sine function

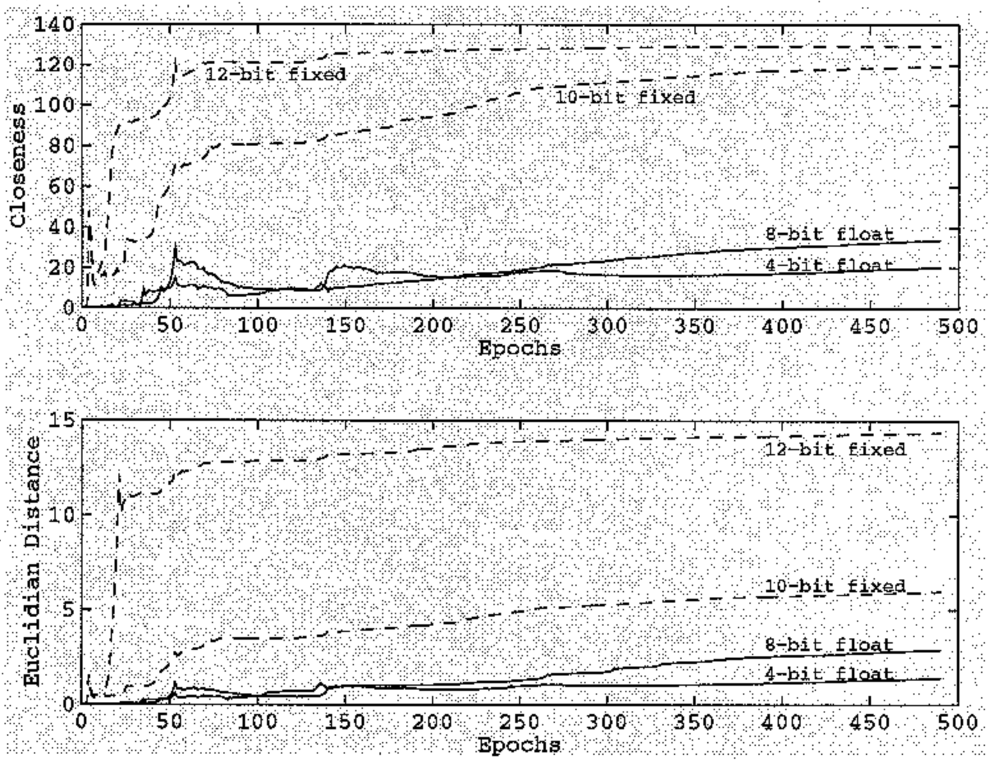


Fig. 14. Angles and Euclidean distances of the output node incoming weight vector with the fixed- and floating- point communications for the Sine function

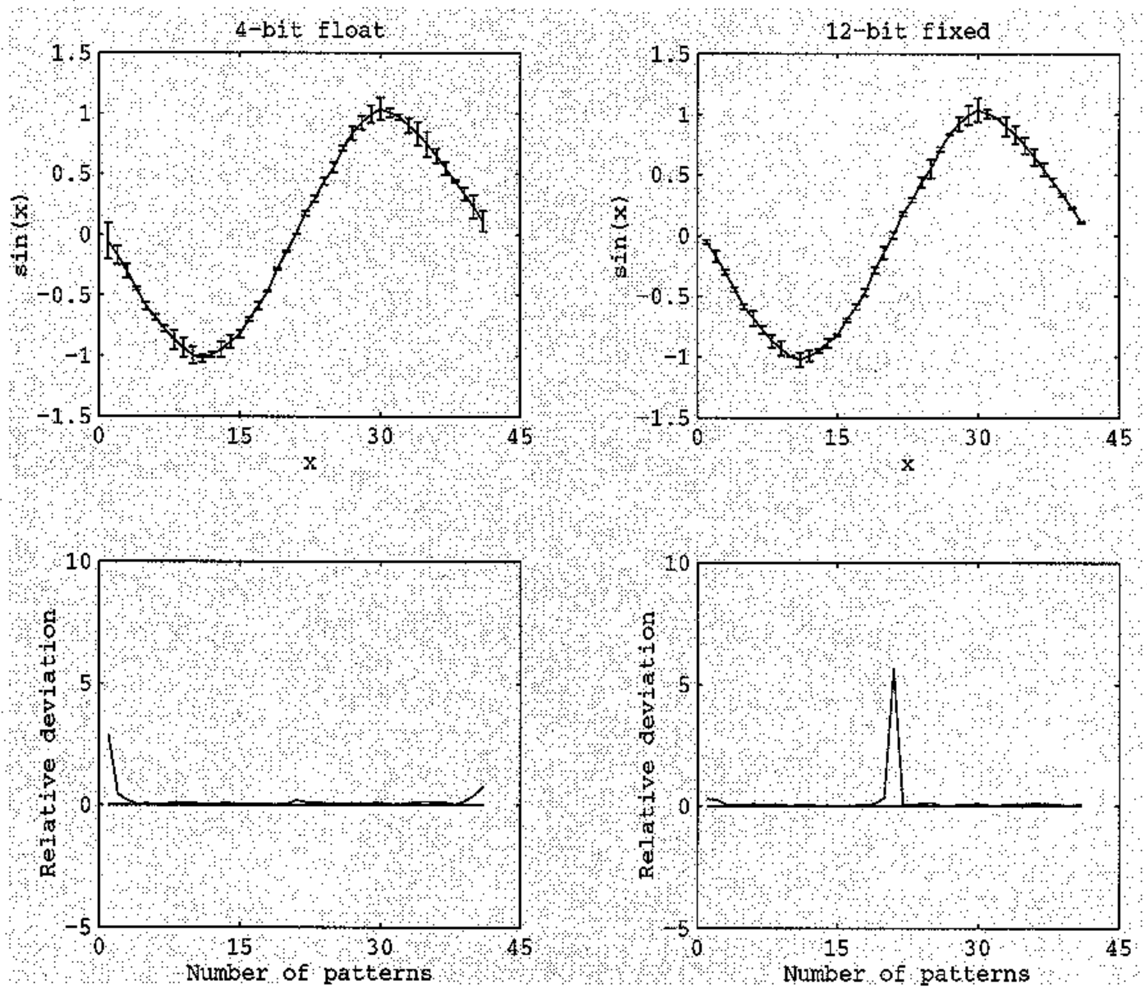


Fig. 15. Absolute and relative errors in the output values of the network with the 12-bit fixed- and 4-bit floating- point communications for the Sine function