The Effects of Quantization on Multilayer Neural Networks

Günhan Dündar, *Member, IEEE,* and Kenneth Rose, *Senior Member, IEEE*

Abstract— **The effect of weight quantization in multilayer neural networks is discussed. A method is derived by which one can predict the performance degradation at the output given the properties of the network and number of bits of quantization. Predictions from this method are evaluated against simulation results. An algorithm to decrease the noise at the output is presented and the results are compared with those above.**

I. Introduction

Due to the speed/space requirements, many different implementations of neural networks have appeared in the literature [1]–[4]. These range from completely analog to completely digital circuitry with mixed-mode designs lying somewhere in between. All of these approaches, however, have been rather *ad hoc* in that very few, if any, have explored the constraints that circuit nonidealities—like nonlinear synapses, neurons that deviate from ideal functions, or errors and limitations in storing weights—bring about. One of the problems encountered in the design of neural-network circuitry both in the digital and in the analog domains is the storage of weights.

Many different techniques have been used for weight storage. Some have stored weights in digital memories [2], either on- or off-chip, some have stored them on capacitors and have incorporated refreshing circuitry [5], some have used several transistors which could be switched according to the weight [6], and some have used EAROM (electrically alterable read only memory) cells. The common question here is the amount of accuracy required to store the weights. In digital memories, this question manifests itself in a choice between wider word lengths in memory which will result in trade-offs in size and, possibly, speed. In dynamic storage, it will determine the size and even the usability of the technique. Intel's 80170NX analog neural-network chip claims greater than six-bit typical precision [7], while the network proposed in [6] has only four bits. Digital implementations [2] may have 16–32 bits of resolution. The question that has to be addressed is the minimum number of bits necessary for successful operation.

In this work, we will try to predict the error in the network given the properties of the network and the number of bits of quantization. We will also introduce a small variation in the backpropagation algorithm, whereby one can get away with a smaller number of bits without sacrificing performance. We will use two examples to demonstrate our results. These two examples were chosen so that they would be representative of the two typical applications of multilayer networks. One of them is a function approximator, namely, a sine generator. The other one represents a pattern classification application and is a four-bit A/D (analog-to-digital) converter.

Manuscript received June 22, 1993; revised December 28, 1993, July 20, 1994, and April 12, 1995.

G. Dündar is with the Department of Electrical and Electronic Engineering, Bogazici University, Istanbul, Turkey.

K. Rose is with the Center for Integrated Electronics and Electronics Manufacturing, Rensselaer Polytechnic Institute, Troy, NY 12180 USA.

IEEE Log Number 9415042.

II. Mathematical Analysis of Quantization Using a Statistical Model

The derivation below is based on an earlier derivation by Xie and Jabri [8], [9]. Some of our approaches, however, are different. For example, we are defining a minimum and maximum for the weight values (as is done in practice) and defining quantization width in terms of these. Furthermore, we are using the true neuron functions rather than the linearized version presented by Xie. When they quantize, Xie and Jabri use a neuron which is linear and whose output limits are determined by the quantization itself rather than the more conventional nonlinear neuron whose output varies between zero and one (or -1 and one). As a result, the maximum and minimum outputs for Xie and Jabri's neuron will change according to the resolution. Since the weights are stored in the synapses, we have defined a maximum and minimum value for quantization intervals depending on the number of bits and have kept the neuron function intact. Therefore, our outputs do not change with resolution, but vary between zero and one. Using the neuron of Xie and Jabri results in a simplification in the calculations, but makes the results less reliable as a guide to the behavior of more general neural networks. In our derivation, we remove this restriction and use a sigmoidal neuron function instead of the hard-limited linear neuron function of Xie and Jabri. Here x^0 are the inputs, y^0 are the signals to the inputs of the first layer of neurons, x^1 are the outputs of the first layer of neurons, y^1 are the inputs to the second layer of neurons, w are the weights, and N is the number of bits of quantization used.

Assume that all the inputs are continuously valued and uniformly distributed between zero and one

$$y_i^0 = \sum_{k=0}^{K_1-1} w_{ik}^0 x_k^0.$$

This equation is a linear transformation where the w_{ik}^0 are the elements of a matrix transforming the vector x_k^0 into the vector y_i^0. Here $x_i^1 = f(y_i^0)$, with $f(u) = (1+e^{-u})^{-1}$, represent the weights joining the inputs to the first layer of neurons, K_1 is the number of inputs, and f is the sigmoidal function used very commonly in neurons. The above equations hold for an ideal network. When there is quantization, however, the following modifications have to be made

$$y_i^0 = \sum_{k=0}^{K_1-1} (w_{ik}^0 + \Delta w_{ik}^0) \cdot x_k^0 = \sum_{k=0}^{K_1-1} w_{ik}^0 x_k^0 + \Delta y_i^0$$

where Δw_{ik}^0 is the quantization error (noise) of the weights and

$$\Delta y_i^0 = \sum_{k=0}^{K_1-1} \Delta w_{ik}^0 \cdot x_k^0.$$

If the weights are quantized with a reasonably large number of bits (≥ 8) and Δ_0 is the quantization level of the least significant bit, then it is reasonable to make the standard approximation of a rectangular probability distribution function for the quantization error. Thus, the quantization error has a mean of zero and a variance

$$\sigma_{\Delta_0}^2 = \frac{\Delta_0^2}{12}.$$

The standard deviation of the input can easily be found by

$$\sigma_{x_k^0}^2 = \int_{-\infty}^{\infty} \left(x - \frac{1}{2}\right)^2 \cdot p(x)dx$$
$$= \int_0^1 \left(x^2 - x + \frac{1}{4}\right)dx = \frac{1}{12}.$$

We further assume that the weights are uniformly distributed between $W_{\max}$ and $-W_{\max}$. Hence, their average is zero and their variance is

$$\sigma_{w_{ik}^0}^2 = \frac{W_{\max}^2}{3} = \frac{(\Delta_0 \cdot 2^N)^2}{12}$$

since $W_{\max} = \Delta_0 2^{(N-1)}$. Substituting these values in the expressions for y and Δy

$$\sigma_{y_i^0}^2 = \frac{(\Delta_0 \cdot 2^N)^2}{144} \cdot K_1$$
$$\sigma_{\Delta y_i^0}^2 = \frac{K_1 \Delta_0^2}{144}$$

where K_1 is the number of inputs. Using the standard deviation for y, one can define an interval where y will most probably exist. If we let

$$A = \sigma_{y_i^0}$$

for simplicity of notation, then y will be in the interval $(-A, A)$ for over two-thirds of the values. Now that we know the standard deviation of the signal and the noise and their means just before the neuron, we can estimate the same parameters after the neuron. We know that

$$p_{x_i^1} = p_{y_i^0} \cdot \frac{dy}{dx}$$

where p_{xi} and p_{yi} are the probability distributions of the variables x_i and y_i. But

$$x = \frac{1}{1+e^{-y}} \rightarrow y = \ln\left(\frac{1}{1-x}\right) \rightarrow \frac{dy}{dx} = \frac{1}{x(1-x)}.$$

Hence, approximating that y_i^0 is uniformly distributed over the interval $(-A, A)$

$$p_{x_i^1} = \frac{1}{2A} \cdot \frac{1}{x(1-x)}.$$

Now let us find the mean for x

$$E[x_i^1] = \int_{-\infty}^{\infty} (x \cdot p_x)dx = \int_{f(-A)}^{f(A)} \left(\frac{x}{x(1-x)} \cdot \frac{1}{2A}\right)dx = \frac{1}{2}.$$

Using the same approach gives

$$\sigma_{x_i^1}^2 = \frac{1}{2A} \cdot \int_{f(-A)}^{f(A)} \left(\left(x - \frac{1}{2}\right)^2 \cdot \frac{1}{x(1-x)}\right)dx$$
$$= \frac{1}{2A}\left(\frac{A}{2} - \tanh\left(\frac{A}{2}\right)\right).$$

One can find the signal at the input of the second layer of neurons using the procedure outlined above

$$y_i^1 = \sum_{l=0}^{K_2-1} (w_{il}^1 + \Delta w_{il}^1) \cdot (x_l^1 + \Delta x_l^1) \approx \sum_{l=0}^{K_2-1} (w_{il} \cdot x_l^1) + \Delta y_i^1$$

where K_2 is the number of neurons in the first layer and Δy_i^1 is defined as

$$\Delta y_i^1 = \sum_{l=0}^{K_2-1} w_{il}^1 \cdot \Delta x_l^1 + \sum_{l=0}^{K_2-1} x_l^1 \cdot \Delta w_{il}^1.$$

We also know the distribution of the weights joining the first and second layers. The following equations can be written immediately

$$\sigma_{w_{ik}^1} = \frac{(\Delta_1 \cdot 2^N)^2}{12}, \quad \sigma_{\Delta_1^2} = \frac{\Delta_1^2}{12}.$$

Now, these values can again be plugged into the equations for y, giving

$$\sigma_{y_i^1}^2 = \frac{(\Delta_1 \cdot 2^N)^2}{12} \cdot \frac{K_2}{2A}\left[\frac{A}{2} - \tanh\left(\frac{A}{2}\right)\right].$$

Please note that, contrary to our assumption, the x_i's used in calculating this value are not, in fact, independent. They are obtained, however, by applying a nonlinearity to a linear combination of the inputs. This operation decreases correlation between two specific x_i's. Thus, while the independence assumption is not exactly true, it is a good approximation. Xie and Jabri have also made this assumption implicitly in their calculations.

Calculating the standard deviation in the error in x is very difficult due to the nonlinear nature of the problem. Here, we use a "small-signal" approximation. Please note that we are using this approximation only to propagate the standard deviation of the error to the next level. This is not the same as Xie and Jabri's neuron because they use their linear neuron for propagating the signals. The "small-signal" approximation can be used for nonlinear neuron functions since the error should be small compared to the signal and the deviation in the error is again small as compared to the error itself. Hence, the neuron function is approximately linear for the deviation of the error.

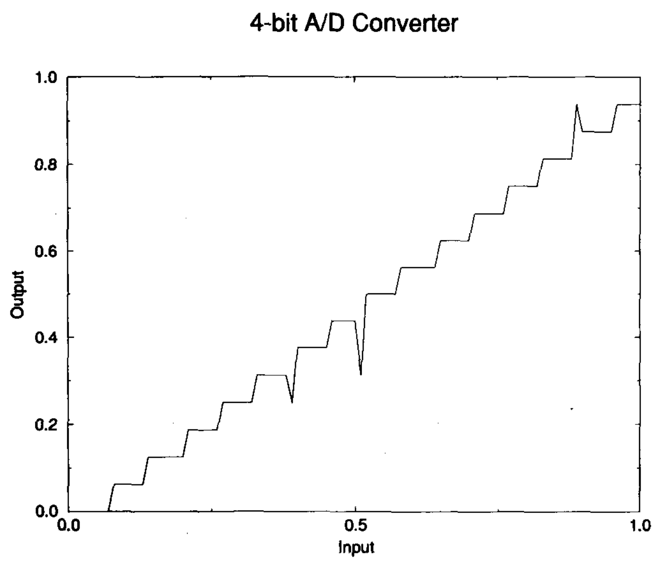

Fig. 1. Transfer characteristics of a four-bit A/D converter realized by neural networks. Please note that there are three misclassified patterns and some artifacts due to the insufficient number of points used to generate the graph.

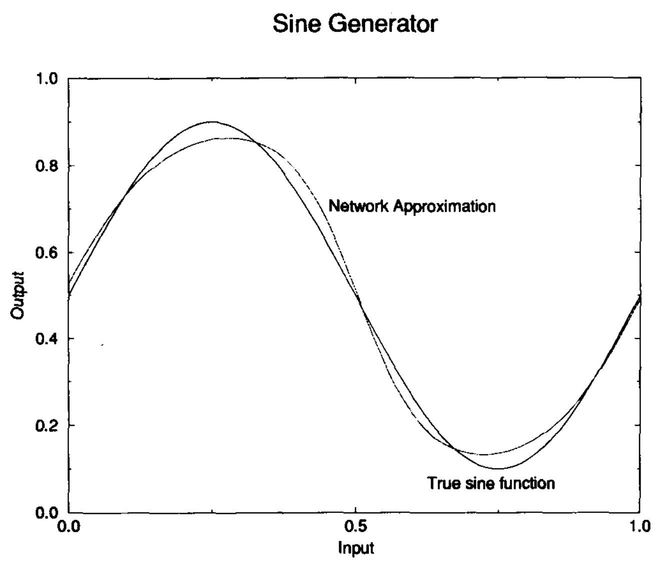

Fig. 2. A neural network sine generator.

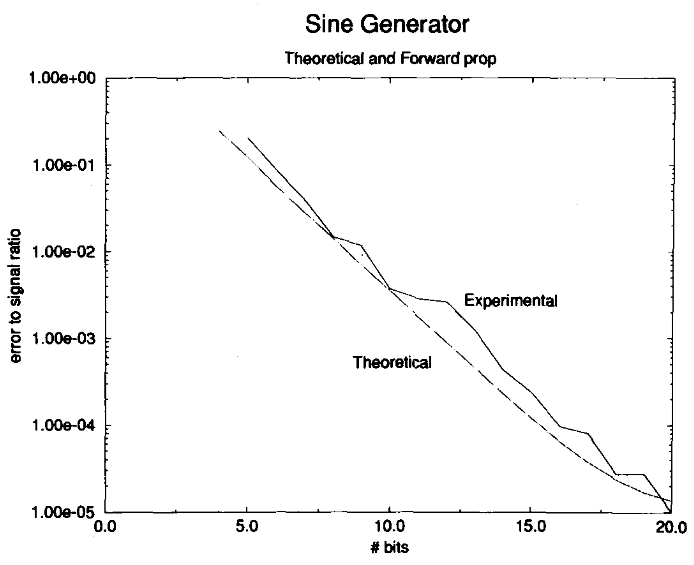

Fig. 3. Comparison of simulation results with theoretical results for the sine generator.

We define "an effective operating point" by determining how much of the signal will pass through the nonlinearity and by how much it will be compressed. We choose an arbitrary r so that the sigmoid has a slope $1/2r$ from $-r$ to r and zero otherwise. It follows directly that only $2r/2A$ of the noise passes through the neuron and is compressed by $1/2r$. This results in the following expression

$$\sigma^2_{\Delta x_l^2} = \left(\frac{1}{2A}\right)^2 \cdot \sigma^2_{\Delta y_l^1}.$$

Then

$$\sigma^2_{\Delta y_l^2} = K_2 \cdot \left[\frac{(\Delta_1 2^N)^2}{12} \cdot \sigma^2_{\Delta x_l^1} + \sigma^2_{x_l^1}\sigma^2_{\Delta_1}\right].$$

III. Simulation Results for the Forward Mode

The nature of each network is different. The resolution required by neural networks differs depending on the application. In the derivation above, the error depends on the maximum values set for the weights and the size of the network. We have taken two networks of medium size as examples here. One of them is a four-bit A/D converter and the second one is a sine wave generator. The A/D converter has one input, 15 hidden nodes, and four outputs. It gives a digital output for inputs ranging from zero to one. The sine generator has one input, 10 hidden nodes, and one output which realizes the function $y = 0.5 + 0.4\sin(x/2\pi)$. The transfer characteristics of the trained networks are given in Figs. 1 and 2, respectively.

For the A/D converter, we measure the error in terms of the percentage of inputs that are misclassified. A pattern is said to be misclassified if it is different, even by one bit, from the ideal pattern. Our performance measure, percent misclassification, is the percentile ratio of the misclassified patterns to the total number of patterns. For the sine generator, the error measure is in terms of SNR. The rms value for the sine wave is calculated. Then, the rms error between the theoretical and the simulation results was calculated as the rms noise. These were divided to get the SNR (signal-to-noise ratio) for the sine wave. It was observed that the error decreased exponentially with the number of bits and reached around 10% at six bits. For the A/D converter, we observed that the network was tolerant to quantization down to 11 bits, where misclassifications started occurring. After eight bits, the error passed 10% and was near 50% around five–six bits which was unacceptable. Fig. 3 shows the results of the sine generator calculations and simulations, while Fig. 4 shows the results for the A/D converter along with the results predicted by the expressions of Xie and Jabri [8], [9].

It should be noted that the simulation results in the A/D converter are not continuously decreasing. This is understandable since these are statistical results and cannot be expected to agree with theoretical behavior exactly. Since we are measuring percentage misclassifications, this lack of agreement will be greater. In the A/D converter, a pattern is

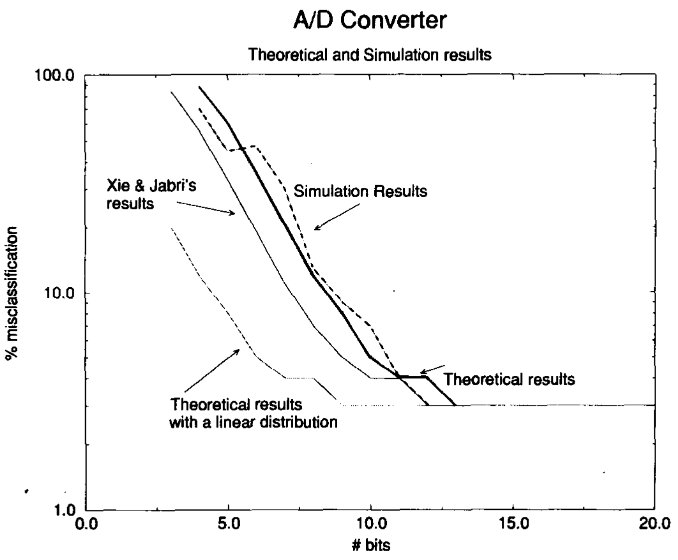

Fig. 4. Comparison of simulation results with theoretical results for the A/D converter.

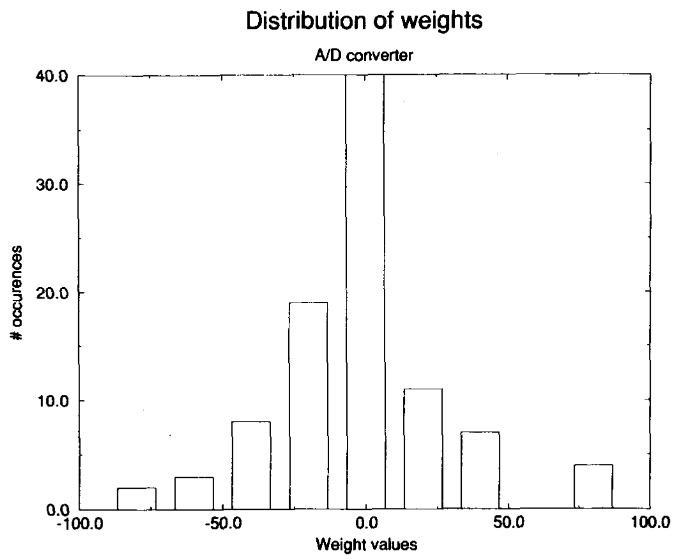

Fig. 5. Distribution of weights in the A/D converter.

considered misclassified whether it has one bit wrong or all four bits wrong. Hence, the error measure allows for this type of behavior. It was observed that SNR values (as in Fig. 3) result in smoother curves. There is a remarkable matching between the theoretical and practical results for the sine generator. This was not the case, however, for the A/D converter originally, where the results were very poor. The reason for this was the assumption that the weights were distributed uniformly. The weight distribution from the simulator is plotted in Fig. 5. This is obviously not a uniform distribution. If a Gaussian distribution was assumed for the weights, the standard deviation in this case is given approximately by

$$\sigma_w = \frac{W_{\max}}{6}.$$

Substituting this value for the variance, the results shown as theoretical results in Fig. 4 were obtained. As can be seen, they present a better match than Xie and Jabri's results. Xie and Jabri's results were not used in the sine generator because they are not applicable to function approximation problems as presented. Xie and Jabri calculate the SNR at the input of the neuron, before passing it through the nonlinearity. Thus, their recursive equations can only calculate σ_y^2 and $\sigma_{\Delta y}^2$. For an SNR, however, one needs the neuron outputs, σ_x^2 and $\sigma_{\Delta x}^2$. It should be noted that Xie and Jabri assumed a uniform distribution for the weights in their derivation.

IV. Backpropagation with Quantization

Having defined and predicted the problem of limited resolution in neural networks, one has to find a solution for this problem. The first approach used was to do training using backpropagation, but with the weights quantized. This meant quantizing the weight values after every update. This caused problems, however, in two respects. One was the drastic reduction in learning speed as the resolution worsened, as demonstrated experimentally in [10]. One can live with this problem because it will not affect the forwardpropagation speed or accuracy once the network is trained. The second problem, however, is even worse. During learning, correction steps can be very small, especially when small values for η are used. As learning proceeds, the steps become even smaller. In VLSI (very large scale integration) implementations, an insufficient weight resolution can cause paralysis during the training phase [11]. In fact, once the correction steps have become smaller (in absolute value) than half the weight resolution, no further correction can be applied and the network paralyzes, even though the output error is still too high for correct behavior.

An algorithm called weight perturbation has been proposed for training limited resolution networks. Several modifications of this algorithm have been successfully implemented [12], [13]. These algorithms essentially perturb the weights and calculate the derivative needed for steepest gradient descent. This way, less precision is required for training. These methods, however, are analogous to the MADALINE III algorithm in that the computational complexity depends more strongly on network size as compared to standard backpropagation. Therefore, these methods are not suitable for training large networks. They should be used after a certain point in training is reached via backpropagation. To limit the computational complexity, one must try to go as far as possible in training with backpropagation. To extend backpropagation, we used several heuristic approaches. Analytical expressions characterizing the dynamic behavior of backpropagation are difficult, if not impossible, to develop. First, the training parameters were made as large as possible so that the correction steps would get larger. This was helpful in all cases of paralysis. For low-bit numbers, some training was done without quantization to get an initial estimate of the weight values. Training was then completed on the quantized network. Even these measures, however, were not enough. A small addition was made to the standard backpropagation algorithm as follows.

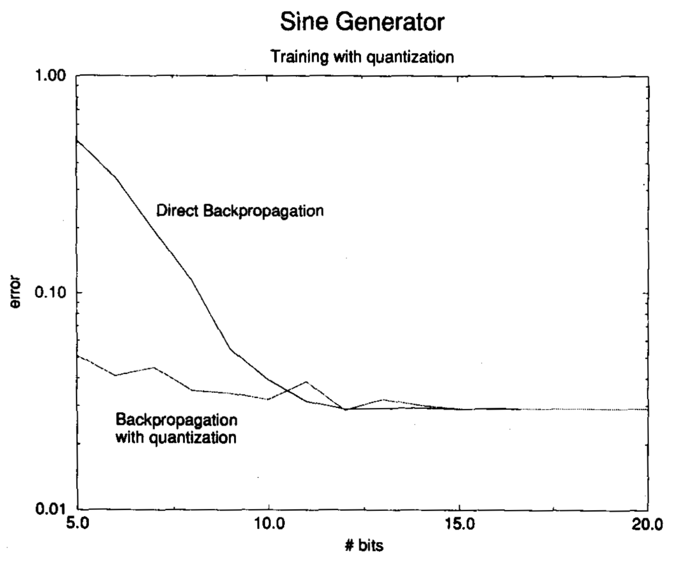

Fig. 6. Comparison of learning with quantization and without quantization for the sine generator. Please note that learning with quantization results in a rather constant error rate while the error for standard backpropagation learning decreases exponentially with the number of bits.

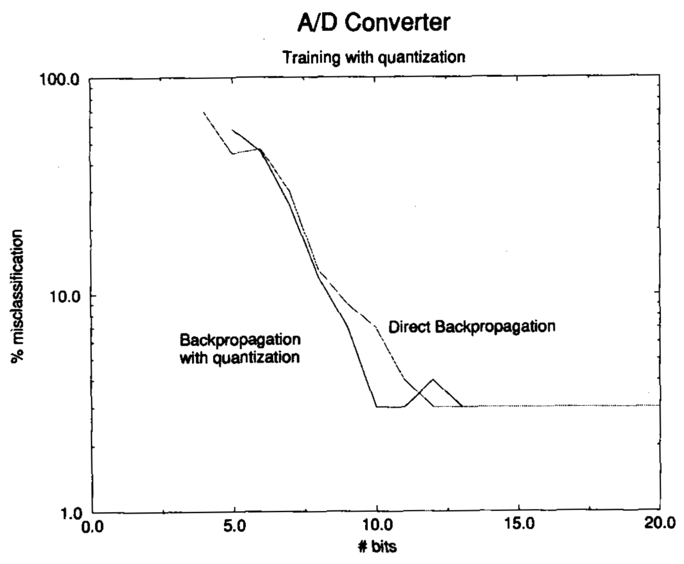

Fig. 7. Comparison of learning with quantization and without quantization for the A/D converter.

When the network became paralyzed, the updates for all the weights were registered. The n largest updates (in absolute value) were chosen (where n is chosen by the user) and the last bits of the weights associated with these updates were flipped in the corresponding direction. As a result, the network was "jolted" out of the local minimum created by the process of quantization. This procedure, which we call "backpropagation with quantization," helped convergence in some cases and reduced the error, although there could be an initial increase in error right after the flipping of the bits. This procedure, however, will not guarantee convergence. The results of backpropagation with quantization given in Figs. 6 and 7 show significant improvement over simple nonquantized training. The effect of backpropagation with quantization, however, differs for our two examples.

For the sine generator, although there are some fluctuations, Fig. 6 shows that backpropagation with quantization produces nearly constant error as the weight resolution (number of bits) increases. We found that the shape of the training surface was different for each resolution, resulting in different error minima of about the same value. The direct backpropagation curve, on the other hand, shows an exponential behavior until 11 bits, after which the error settles to a constant error value. This error value would be present even without quantization. Using backpropagation with quantization gives error values with six-bit resolution which would only be achieved with 10-bit resolution for direct backpropagation. For the A/D converter, which represents pattern classification applications, the improvements produced by backpropagation with quantization, as shown in Fig. 7, are less dramatic. Here, too low a resolution completely paralyzes training and we do not get the constant error of Fig. 6. There is evident improvement, however. Backpropagation with quantization achieves misclassification percentages with 10-bit resolution which could only be achieved with 12-bit resolution using direct backpropagation without quantization. Consequently, using this approach can make the difference between success and failure in many limited resolution systems. If the system being used has few bits representing the weights, the user may be able to achieve convergence using this approach.

V. Conclusion

We have shown how the performance of neural networks trained using conventional, direct backpropagation is seriously worsened by quantization noise. For two examples, representative of function approximation and pattern classification, the performance of neural networks shows a roughly exponential dependence on resolution, the number of bits used to characterize the weights. We found that a minimum resolution of around 10 bits was necessary for acceptable operation, although this depends on the size of the network and the nature of the problem. We have developed an analytical theory for weight quantization which agrees well with simulations for the two examples.

Avoiding quantization noise requires minimum resolutions for weight storage and multiplication which are difficult to achieve when analog neural networks are implemented as integrated circuits. We have introduced a training procedure, "backpropagation with quantization," which helps overcome these limitations. This procedure produces low errors with lower resolution than conventional, direct backpropagation requires.

We have only considered the effects of weight quantization in this paper. Nonlinear synapses or nonideal neuron functions, however, can have comparable influence on neural network performance. Fortunately, our approach to analyzing quantization can be extended to the analysis of nonlinearities [14].

Acknowledgment

The authors would like to thank B. Geuskens for her help in finalizing this manuscript.

REFERENCES

[1] L. W. Massengill, "A dynamic CMOS multiplier for analog VLSI based on exponential pulse-decay modulation," *IEEE J. Solid State Circuits*, vol. 26, pp. 268–276, 1991.

[2] P. Treleaven, M. Pacheco, and M. Vellasco, "VLSI architectures for neural networks," *IEEE Micro. Mag.*, pp. 8–27, Dec. 1989.

[3] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically trainable artificial neural network (ETANN) with 1024 'floating gate' synapses," in *Proc. IEEE/INNS Int. Joint Conf. Neural Networks*, vol. II, Washington, 1989, pp. 191–196.

[4] G. Dündar and K. Rose, "Analog neural network circuits for ASIC fabrication," in *Proc. 5th IEEE ASIC Conf.*, Rochester, 1992, pp. 419–422.

[5] Y. Tsividis and S. Satyanarayana, "Analogue circuits for variable-synapse electronic neural networks," *Electron. Lett.*, pp. 1313–1314, 1987.

[6] O. Rossetto, C. Jutten, J. Herault, and I. Kreuzer, "Analog VLSI synaptic matrices as building blocks for neural networks," *IEEE Micro. Mag.*, pp. 56–63, Dec. 1989.

[7] Intel 80170NX ETANN data sheets, Feb. 1991.

[8] Y. Xie and M. A. Jabri, "Analysis of the effects of quantization in multilayer neural networks using statistical model," *Electron. Lett.*, vol. 27, pp. 1196–1198, 1991.

[9] ——, "Analysis of the effects of quantization in multilayer neural networks using a statistical model," *IEEE Trans. Neural Networks*, vol. 3, pp. 334–338, 1992.

[10] B. W. Lee and S. W. Kim, "Required dynamic range and accuracy of electronic synapses for character recognition applications," in *Proc. IEEE ISCAS*, 1992, pp. 1545–1548.

[11] L. M. Reyneri and E. Filippi, "An analysis on the performance of silicon implementations of backpropagation algorithms for artificial neural networks," *IEEE Trans. Comput.*, vol. 40, pp. 1380–1389, 1991.

[12] A. J. Montalvo, P. W. Hollis, and J. J. Paolos, "On-chip learning with limited precision circuits," in *Proc. Int. Joint Conf. Neural Networks*, 1992, pp. I-192–I-201.

[13] Y. Xie and M. A. Jabri, "On the training of limited precision multilayer perceptrons," in *Proc. Int. Joint Conf. Neural Networks*, 1992, pp. III-942–III-947.

[14] G. Dündar and K. Rose, "Effects of nonlinear synapses on the performance of multilayer neural networks," *Neural Computa.*, to be published.

Günhan Dündar (S'90–M'95) was born in Istnbul, Turkey, in 1969. He received the B.S. and M.S. degrees from Bogazici University, Istanbul, in 1989 and 1991, respectively, and the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1993, all in electrical engineering.

Since Feb. 1994, he has been with the Electrical Engineering Department, Bogazici University. Currently, he is on leave from that position and is with the Turkish Naval Academy, completing his military service. His research interests include analog and digital IC design, neural networks, and image processing.

Kenneth Rose (S'55–M'61–SM'82) received the B.S. degree in engineering physics in 1955 and the M.S. and Ph.D. degrees in electrical engineering in 1957 and 1961, respectively, all from the University of Illinois.

In 1965 he joined the faculty of Rensselaer Polytechnic Institute, Troy, NY, where he is presently a Professor in the Electrical, Computer, and Systems Engineering Department and an active member of Rensselaer's Center for Integrated Electronics and Electronics Manufacturing. His current research interests include VLSI design, interconnect modeling, cryogenic electronics, and neural networks.