

Analog Feedforward Neural Networks with Very Low Precision Weights

Shahram Abdollahi Alibeik, Farid Nemati, Mehrdad Sharif-Bakhtiar

Department of Electrical Engineering

Sharif University of Technology

P.O. Box 11365-9363, Azadi Ave.

Tehran, IRAN

abdolla2@ee.sharif.ac.ir, neamati@ee.sharif.ac.ir

ABSTRACT

An Off Chip training algorithm for feedforward neural networks is presented. This algorithm has been successfully used to train networks with weight precision as low as 1 bit. The effect of reducing the weight precision on the generalization ability of the network is presented. The network performance, in the presence of hardware non-idealities, has also been investigated. It is shown that a network with low precision weights can well tolerate the effect of hardware non-idealities if the network is properly trained.

1. Introduction

Feedforward neural networks have been successfully used in many pattern recognition applications. It has been proved that they can approximate any continuous function, to any arbitrary precision, using only one hidden layer with sufficient number of nodes [10]. However, because of the parallel nature of the computations in neural networks, simulation of large networks on serial computers is slow. Some of the key features of neural networks like fault tolerance and the ability to compensate hardware non-idealities, make them attractive for VLSI implementations. Several chips have been introduced, using digital or analog VLSI circuits to implement neural networks [5][6]. Digital circuits are precise and insensitive to hardware non-idealities such as noise, offset and component variation and can be interfaced easily to peripherals. However, digital circuits, in comparison to their analog counterparts, are slower and occupy more area. Analog circuits can be small, simple and fast at the expense of more sensitivity to hardware non-idealities. On the other hand, considering the fault tolerance feature of neural networks, it is expected that the sensitivity of analog circuits to hardware non-idealities would not be restrictive. Therefore, when designing hardware for analog neural networks the allowable limits of non-idealities are to be known.

The training method chosen, has a great influence on these limits and the minimum allowable precision of weights. The existing training methods, according to the hardware used to perform the required computations, can be divided into three categories, namely: On Chip training, Partial On Chip training and Off Chip training methods.

In On Chip training method, all the computations including forward and backward computations in Back Propagation algorithm, are

performed by analog hardware. This method, though fast, needs high precision for weight representation (usually more than 10 bits¹) [2][3]. This leads to difficulties with the design of the high performance circuit components. The convergence of the training algorithm, depends also on the low offset of the backward computation circuitry [1][2]. It is possible to increase the complexity of the training algorithm to relax the required weight precision [3][4], at the price of more complex hardware. In general, this method is not appropriate for analog implementations due to its high precision requirements.

In Partial On Chip training method, the forward computations are performed by analog hardware while a serial computer performs the other computations with high precision. As a result, this method is slower than On Chip training method during the training phase. In many applications, such as pattern recognition, the training is performed prior to the actual use of the chip and once the chip is trained, it can potentially operate at high speed. This method can compensate for the hardware non-idealities and unlike the previous method, it is insensitive to the offset of the analog circuits [5]. Using this method, weight precisions as low as 5 or 6 bits have been reported[5].

In Off Chip training method, a computer performs all the computations with high precision and the computed value for the weights are used in the analog hardware of the forward path. The hardware non-idealities that can not be simulated by computer (e.g., component variation), can be put into account by performing a final adjustment through a Partial On Chip training [6]. The use of computer for training provides the ability of devising algorithms for very low precision weight representation [7][8]. Therefore, simple and small analog circuits can be utilized for

¹ In this article, sign bit is not counted but is always present.

multiplication and other operations and the resulting hardware can be highly integrated and will be fast. However, hardware simplification through the reduction of the weight precision, in an Off Chip training method, can not be efficiently done unless the following questions are answered.

- 1- Using a proper training algorithm, how much can the weight precision be reduced ?
- 2- How does the reduction of weight precision affect the generalization ability of the network?
- 3- How much can the trained network tolerate the non-idealities in the actual analog hardware?

2. Training Algorithm

In Off Chip training specially when resolution is very low, the incremental change in weights obtained from simple Back Propagation learning methods decrease gradually and will become smaller than one quantization step. This will stop the training progression, i.e., weights will be trapped in one of the spurious local minima caused by weight quantization. Several solutions addressing this problem have been suggested, such as adaptive changing of learning rate in order to escape from local minima [4][8]. However, to avoid failure, a definite method for changing the learning rate is not given.

A good training method to overcome the above problem, should be successful in difficult training tasks and must be efficient enough to keep the number of hidden nodes as low as possible. An algorithm with these properties is given in the following steps.

STEP a) Initially the network is trained with floating point weight precision using Back Propagation Training Algorithm. Conjugate gradient method is used to speed up the convergence. It is also necessary that the desired outputs of the network be equal to the asymptotic values of the activation function. Our experiments showed that this plays an essential role in successful network training and reducing the network sensitivity to the quantization of the weights.

STEP b) The weights are quantized to n bits, after being scaled appropriately by multiplying the gain of the activation functions by a scale factor and dividing the weights of the input branches of the nodes by the same scale factor. A common method for optimizing scale factors is to minimize the distance between the quantized and original weight vectors[8]. In our algorithm, however, these scale factors are found such that the error of the network on the training set is minimum after scaling and quantization of the weights.

STEP c) The network is then trained using a modified version of Back Propagation, QGDR (Quantized Gradient Descent Rule), so as to avoid

spurious local minima. This algorithm makes use of the sign of the gradient vector obtained from Back Propagation. Each weight (W_i) is changed in the opposite direction of dE/dW_i for an amount equal to one quantization step. This change is accepted if it decreases the network error (E), otherwise it is rejected. This process is applied to all of the weights in the network until the network error becomes less than a predefined value or no further changes in the weights are made in one epoch. Our experiments showed that QGDR can find a good result in the neighborhood of the minimum found in STEP (a), in contrast to the adaptive learning rate method which has the potential to move the weights away from initial minimum. QGDR is also much faster than methods like Simulated Annealing or Blind Random Search.

STEP d) The number of bits, n , representing quantized weights, is decreased by one and STEPs (b) and (c) are repeated until the desired weight precision is reached.

3. Test Vehicles

To test the performance of the training algorithm discussed in the previous section, the following problems were used.

Problem 1 - 6-Bit Parity Problem: A seven input-one output network with one hidden layer was used for which one of the inputs is always set to one so as to implement the bias terms. Experiments were done separately for 10 and 15 nodes in the hidden layer. The training set had 64 members, containing all possible states for the input. With the parity problem, output changes as the input is changed by one bit and also the training set contains all of the possible input states. This makes the parity problem a good vehicle for testing the power of a training algorithm specially when the number of bits representing quantized weights is to be low.

Problem 2 - A simplified version of the Persian (Arabic) digit recognition problem: In this problem a network with one hidden layer was utilized which had 100 input nodes corresponding to the 10×10 matrix representing the digits. The 10 output nodes of the network were interpreted on Winner Take All basis. The number of hidden nodes were 10, 15 and 20 for three separate groups of experiments. The training set contained 30 members with 3 samples for each digit. Although this problem is fairly simple and can be trained by simpler training algorithms, it was selected in order to test the generalization ability of the network trained with the new algorithm. The test set contained 1200 members, obtained by adding 5, 10, 15 and 20 percent noise, 10 times, to each member of the training set.

Table 1: The Training Results
(ϵ stands for numbers less than $1E-4$)

	Problem 1				Problem 2					
	10 Hidden Nodes		15 Hidden Nodes		10 Hidden Nodes		15 Hidden Nodes		20 Hidden Nodes	
Weight bits	Error	No. of Faults	Error	No. of Faults	Error	No. of Faults	Error	No. of Faults	Error	No. of Faults
Float	ϵ	0	ϵ	0	ϵ	0	ϵ	0	ϵ	0
6	ϵ	0	ϵ	0	ϵ	0	ϵ	0	ϵ	0
5	ϵ	0	ϵ	0	ϵ	0	ϵ	0	ϵ	0
4	ϵ	0	ϵ	0	ϵ	0	ϵ	0	ϵ	0
3	$6.1E-4$	0	ϵ	0	ϵ	0	ϵ	0	ϵ	0
2	$1.6E-3$	0	ϵ	0	ϵ	0	ϵ	0	ϵ	0
1	0.5	1	ϵ	0	$4.3E-3$	0	$1.3E-3$	0	ϵ	0

4. Training Results

In all of our experiments, the activation function was taken as $f(x) = \tanh(cx)$ where c was unity in the beginning of STEP (a) of the training algorithm. Also, the following definition was used for Error (Cost Function) :

$$\text{Error} = 0.5 \times \sum_i \sum_j (\text{Actual Output}_{ij} - \text{Desired Output}_{ij})^2$$

(i = All members of the training set, j = All network outputs)

The number of bits, representing the weights, were taken equal to 6, after the completion of STEP (a). This number was gradually reduced to one by successive application of STEPs (b) and (c). In each case the training was stopped, if the error became less than $1E-4$.

The training results for problems 1 and 2 (Table 1) for different number of hidden nodes show that in both problems, the ability to learn the training set, is quite satisfactory, even with 1 bit precision for weights. It can also be concluded from Table 1 that the learning ability of the network improves as the number of hidden nodes is increased. For both

problems, the final error on the training set for 1 bit weight precision, can become as low as the final error for floating point weight precision, if the number of hidden nodes is large enough.

The results of testing the network of problem 2 with the test set and for different number of hidden nodes are given in Fig. 1. Fig. 1 shows that decreasing the weight precision, reduces the generalization ability of the network. But this reduction in the generalization ability is negligible for weight precisions as low as 2 bits. Fig. 1 also shows that despite the improvement in learning ability gained by increasing the number of hidden nodes, the generalization ability does not necessarily improve. In fact, there is an optimum value for the number of hidden nodes for which the generalization power of the network is maximum.

5. Simulation of Hardware Non-Idealities

Analog hardware suffer from several non-ideal effects such as noise and offset. Thus, when a neural network is implemented using analog hardware, questions such as the following arise: How these non-idealities affect the network performance? To what extent are these effects tolerable? Does reducing the number of bits representing quantized weights have a significant effect on the network tolerance to non-idealities?

To answer these questions, non-ideal effects were applied to each of the trained networks of previous section (both with 15 hidden nodes) and the changes in the network performance on training set and test set were investigated. The major non-ideal effects in analog hardware, i.e., non-linearity, noise, offset and component variation, were modeled as follows.

a) Non-linearity - Non-linearity was modeled by applying function $\tanh(\beta x)/\beta$ to the network inputs and the multiplier outputs. This function has unity

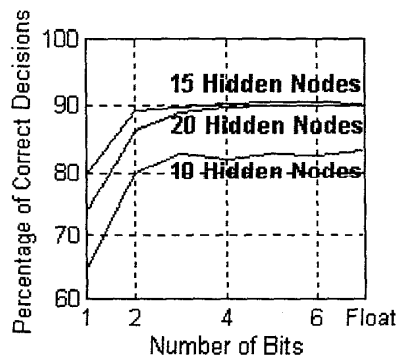


Fig. 1. Generalization Ability in Problem 2

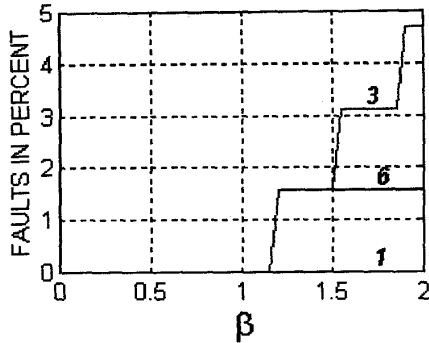


Fig. 2. Nonlinearity in Problem 1

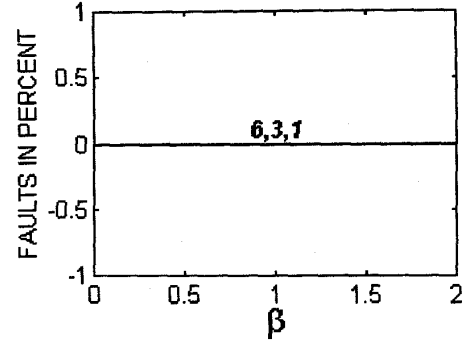


Fig. 3. Nonlinearity in Problem 2

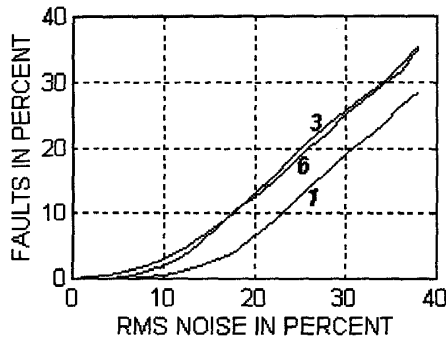


Fig. 4. Noise in Problem 1

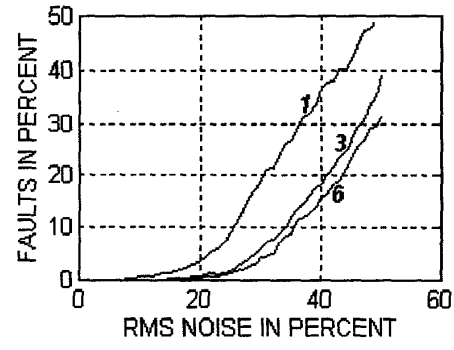


Fig. 5. Noise in Problem 2

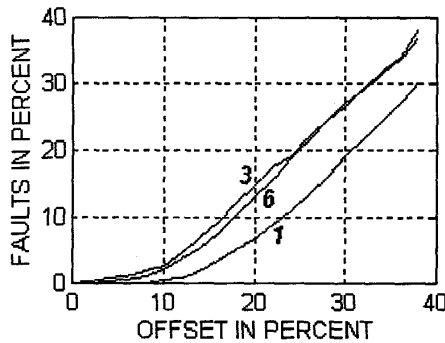


Fig. 6. Offset in Problem 1

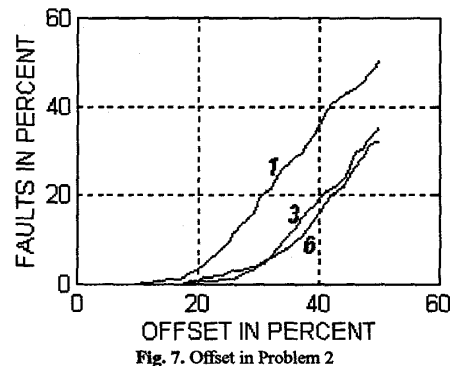


Fig. 7. Offset in Problem 2

gain at $x=0$ for all values of β and its non-linearity increases as β is increased (more than 50% for $\beta = 2$).

b) Noise - Noise was simulated by a random variable with uniform distribution and zero mean for which its standard deviation (Noise RMS value) was taken as a simulation parameter. This random variable was added to the network inputs to investigate the network behavior as a function of the Input Equivalent Noise.

c) Offset - Numbers with a constant magnitude and random sign was added to each network input to model the component offsets referred to the input.

d) Component Variation - Component variation was modeled by considering a random

coefficient with unity mean for each multiplier. The standard deviation of this random variable was a simulation parameter and is interpreted as component variation percentage.

The results of the experiments are shown in Fig. 2 through Fig. 9. In all of these figures, the vertical axis is in terms of fault percentage on the training set and the horizontal axis is in terms of one of the non-idealities mentioned above. Each figure contains three curves corresponding to 6, 3 and 1 bit precision for weights as specified for each curve. In these experiments, when the non-ideal effects were modeled as random variables, the results of ten experiments were averaged and used as the final result.

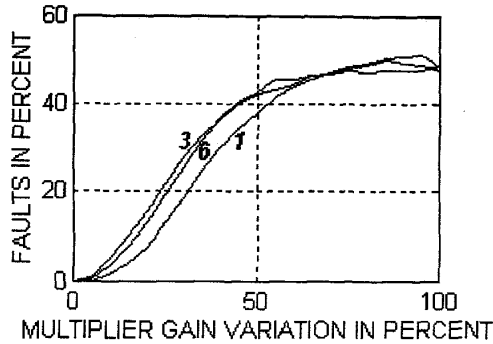


Fig. 8. Component Variation in Problem 1

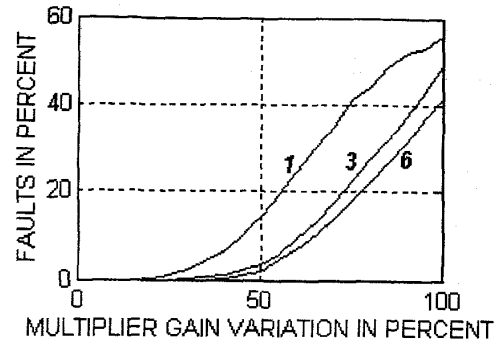


Fig. 9. Component Variation in Problem 2

As shown in Fig. 2 and Fig. 3, the effect of non-linearity is negligible and in the worst case the fault percentage on the training set is less than 5% (for 3 bits). For noise and offset (Fig. 4 to Fig. 7), when less than 12% for problem 1 and 20% for problem 2, fault percentage on the training set is less than 5% for weight precision down to 1 bit. Fig. 8 and Fig. 9 show that large component variations also can well be tolerated by the network. Fig. 8 and Fig. 9 show that the component variation can be higher than 10% while the fault percentage on the training set remains less than 5% with weight precision down to 1 bit.

The results of the experiments on the generalization ability of the network in presence of non-idealities are given in Table 2. As shown in Table 2, the generalization ability of the network on the test set does not degrade for the large values of the non-idealities applied to the network.

Therefore, it can be concluded that the reduction of weight precision has a negligible effect on the non-ideality tolerance of the network, if the lower precision network can be trained to the same error level as the higher precision network. Failure in properly training the low precision network not only limits the generalization ability of the network but it also increases the sensitivity of the network to the hardware non-idealities.

6. References

[1] - "The Effects of Analog Hardware Properties on Back propagation Networks with On-Chip Learning",

- B.K. Dolenko & H.C. Card, ICNN 1993, Page 110
 [2] - "On-Chip Learning in the Analog Domain with Limited Precision Circuits", A.J. Montalov & P.W. Hollis & J.J. Paulos, IJCNN 1992, Page I - 196
 [3] - "Learning with Limited Numerical Precision Using the Cascade Correlation Algorithm", M. Hoehfeld & S.E. Fahlman, IEEE Transactions on Neural Networks, July 1992, Page 602
 [4] - "Effects of Weight Discretization on the BP Learning Method: Algorithm Design and Hardware Realization", D.D. Caviglia & M. Valle & G.M. Bisio, IJCNN 1990, Page II - 631
 [5] - "Back-Propagation Learning and Non-idealities in Analog Neural Network Hardware", R.C. Frye & E.A. Rietman & C.C. Wong, IEEE Transactions on Neural Networks, January 1991, Page 110
 [6] - "Analog CMOS Implementation of a Multilayer Perceptron with Nonlinear Synapses", J.B. Bolt & W. Guggenbuhl, IEEE Transactions on Neural Networks, May 1992, Page 457
 [7] - "A Digital Multilayer Neural Network with Limited Binary Expressions", K. Nakayama & S. Inomata & Y. Takeuchi, IJCNN 1990, Page II - 587
 [8] - "Design of Multilayer Neural Networks with Power-of-Two Weights", M. Marchesi & N. Benvenuto & G. Orlandi, ISCAS 1990, Page 2951
 [9] - "Analysis of the Effect of Quantization in Multilayer Neural Networks Using a Statistical Model", Y. Xie & M.A. Jabri, IEEE Transactions on Neural Networks, March 1992, Page 334
 [10] - "Introduction to the Theory of Neural Computation", J. Hertz & A. Krogh & R.G. Palmer, John Wiley & Sons, 1991

Table 2: Effect of Hardware Non-idealities on Generalization Ability in Problem 2
 (The table shows the percentage of correct decisions on test set)

Weight Resolution	Ideal Hardware	Non-linearity ($\beta = 1$)	Noise (10 %)	Offset (10 %)	Component var. (20 %)
6	90.5	90.5	90.2	88.7	88.2
3	89.9	89.9	88.2	88.0	88.3
1	80.1	80.1	76.2	77.2	76.1