Training Algorithms for Limited Precision Feedforward Neural Networks

Yun Xie
Department of Electronic Engineering,
Tsinghua University,
Beijing 100084, P.R.China
Marwan A. Jabri
Department of Electrical Engineering
The University of Sydney
N.S.W. 2006, Australia

SEDAL Technical Report No. 1991-8-3

Abstract

In this paper we analyse the training dynamics of limited precision feedforward multilayer perceptrons implemented in digital hardware. We show that special techniques have to be employed to train such networks where each variable is quantised to a limited number of bits. Based on the analysis, we propose a Combined Search (CS) training algorithm which consists of partial random search and weight perturbation and can easily be implemented in hardware. Computer simulations were conducted on IntraCardiac ElectroGrams and sonar reflection pattern classification problems. The results show that using CS, the training performance of limited precision feedforward MLPs with 8 to 10 bit resolution can be as good as that of unlimited precision networks. The results also show that CS is insensitive to training parameter variations.

1 Introduction

When neural networks are to be used on limited precision digital hardware, problems may arise in their training because all network parameters and variables are quantised. The characteristics of limited precision networks may be heavily affected by the quantisation and their training may become much harder.

Several researchers have conducted simulations that implement existing training algorithms using limited precision [1,2,3]. However, few attempts have been made to develop training algorithms specially geared for limited precision.

The work described in this paper was conducted to investigate the performance and the training dynamics of digitally implemented feedforward MLPs where all the variables are represented by a limited number of bits in fixed point format.

The paper is structured as follows. In Section 2, the quantization effects on the training dynamics are analyzed. In Section 3, the Combined Search training algorithm (CS) for limited

precision neural networks is presented. The performance of the algorithm is analyzed and implementation issues are discussed. In Section 4, we present extensive simulations on IntraCardiac ElectroGrams (ICEG) and sonar reflection pattern recognition problems, and compare the performance of our algorithms against several others including backpropagation. We also discuss the sensitivity of CS to its training parameters; In Section 5 the hardware implementation of CS is discussed briefly.

Throughout the paper, and unless indicated otherwise, we use the term MLP to indicate a feedforward multi-layer perceptron network.

2 Quantization Effects on Neural Network Training

In [4], we have used statistical models to analyze the *static* effects of quantization on MLPs. As for the dynamics of the training process, the major effect of quantization (especially when small numbers of bits are used) is that small variations in the error function are quantized to zero. This leads to the formation of a large number of plateaus in the error surface in addition to those that already exist (see Figure 1). These plateaus can be absolutely flat as far as quantisation is concerned, and make the training much harder because they trap the training procedure.

Trapping occurs more frequently if the training algorithm only makes use of the information in the vicinity of the current point in the weight space and especially if it based on the differentiable property of the error function. Therefore, if as few bits as possible are to be used for the quantisation, while ensuring successful training and performance, training could not solely rely on the local information such as the gradient of the error function. Methods performing non-local search in the weight space have to be incorporated.

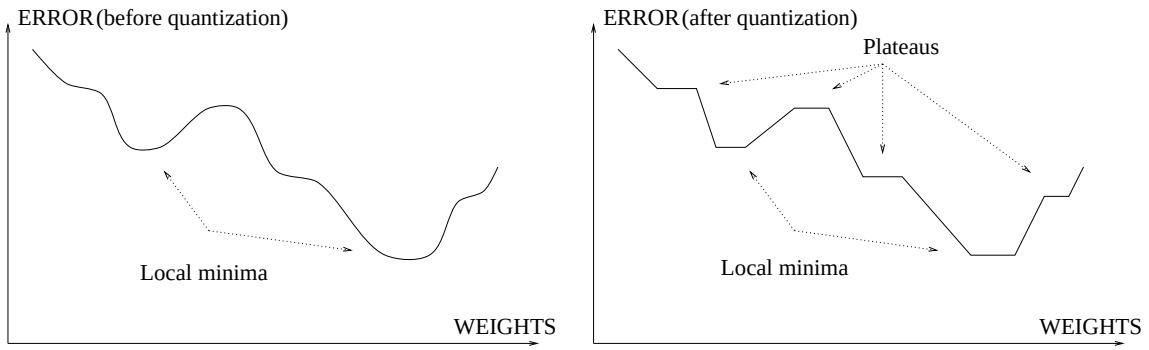

Figure 1: The error function before and after quantization

In this paper, the terms "local search algorithm" and "non-local search algorithm" are used to differentiate between algorithms that use information from a close vicinity as opposed to a wider search area.

3 The Combined Search Algorithm for Training Limited Precision MLPs

It is desirable that limited precision training algorithms possess the following properties:

- would require few computations to deal with limited precision training as compared to the unlimited precision case,
- can escape plateaus and local minima, and
- be simple for hardware implementation.

Backpropagation (BP) is the most popular training algorithm for MLPs [5]. It is efficient in training but difficult to implement in hardware. This is due to its high precision requirements (12 to 16 bits) as it involves a relatively large number of computations in the forward and backward paths [3,1,6].

For hardware implementation simplicity, the Weight Perturbation algorithm (WP) [6] is more attractive. This was originally proposed for analog neural networks. The idea is that by injecting a small perturbation ΔW_{ij} to weight W_{ij}, the partial derivative $\partial Error/\partial W_{ij}$ can be approximated by the measurement $(Error(W_{ij}+\Delta W_{ij})-Error(W_{ij}))/\Delta W_{ij}$, where $Error(.)$ is the error (cost) function to be minimized during training. Therefore the weights can be updated according to:

$$W_{ij}^{new} = W_{ij}^{old} + \eta(Error(W_{ij}) - Error(W_{ij} + \Delta W_{ij}))/\Delta W_{ij} \quad (1)$$

where η is the learning rate.

WP, although not as efficient as BP, requires only feedforward propagation during training, and as a result it is easier to implement in hardware and is less sensitive to quantization errors.

There are two serious problems for BP and WP when they are implemented with limited precision.

- The training convergence is more sensitive to the learning rate than in the case of unlimited precision. The reason is that the learning rate should be small enough to avoid dramatic changes to the weights in a single iteration, and large enough to avoid the quantisation of the weight change to zero.
- BP and WP are local search algorithms. As indicated in Section 2, their performance degrades with lower bit resolutions and they are likely to be trapped on plateaus.

To overcome these problems, a combination of local and non-local search algorithms has to be employed. The training procedure we are proposing in this paper combines such techniques and hence its name, the Combined Search algorithm (CS).

The algorithm makes use of Modified Weight Perturbation (MWP) and Partial Random Search (PRS). In order to keep the weights out of saturation, neurons are equipped with a gain control which is adjusted during training. We describe below the algorithm in detail.

A flow diagram of the Combined Search algorithm is shown in Figure 3. In the diagram, "Weight(I,J)" is the Jth weight connecting the (I-1)th and the Ith layer of neurons; "Gain(I)" is the gain of the neurons in the Ith layer; "L" is a constant larger than 1 (see Section 3.1.2); "Error" is the cost function to be minimized; "Nlayers", "Nweight[I]" and "Nepochs" respectively represent the number of layers, weights in Ith layer and epochs.

The algorithm makes use of Modified Weight Perturbation (MWP) and Partial Random Search (PRS). In order to keep the weights out of saturation, neurons are equipped with a gain control which is adjusted during training. We describe below the algorithm in detail.

```
CombinedSearch{
    do{

        PartialRandomSearch();
        ModifiedWeightPerturbation();
    }while(terminating conditions are not met)
}

PartialRandomSearch{
    do{
        for(I<Nlayers and J<Nweight[I]){
            if(absolute value of Weight[I,J] exceeds the bound)
                        GainAdjustment(I);
            replace Weight[I,J] by a random value;
            if(Error increases)
                        reject the new value of Weight(I,J) and restore the old;
        }
    }while(epochs<MaxNepochs)
}

ModifiedWeightPerturbation{
    do{
        for(I<Nlayers and J<Nweight[I]){
            if(absolute value of Weight[I,J] exceeds the bound){
                        GainAdjustment(I);
            calculate the sign of the partial derivative by perturbing Weight(I,J);
            update Weight[I,J] accordingly by a fixed step;
        }
    }while(epochs<MaxNepochs and Error is decreasing)
}

GainAdjustment(I){
        Gain[I]*L;
        for(J<Nweight[I])
                Weight[I,J]/L;
}
```

Figure 2: Flow Diagram of the Combined Search Algorithm

3.1 Gain Adjustment

The gain of a neuron in a neural network is a constant that amplifies or attenuates the net input to the neuron. For a given neuron, its transfer function $f(.)$, its net input net and g its gain, its output y is $y = f(g * net)$.

It is easy to see that any change to a neuron gain is equivalent to multiplying its input weights with an appropriate value. Therefore, when an unlimited precision neural network is trained, the gains are usually kept constant although gain adaptation may produce some benefits [7].

But in limited precision networks with fixed point representation, the inputs, weights and neurons states are represented by a finite number of bits in fixed point format. Thus, during training, the weights are constrained within a limited range and the gains have to be set appropriately. However, it is impossible to guess *a priori* how large the gains should be to guarantee successful training, gain adaptation provides a natural solution in this problem.

3.1.1 Gains Initialization

For the nth hidden layer, y_i^{n-1} is the input to neuron i, K_{n-1} is the number of input neurons. Δ_{n-1} and N_{n-1} are the quantization width and the number of bits (excluding the sign bit) in the quantization of the outputs of the $(n-1)$th hidden layer and the weights between the $(n-1)$th and the nth hidden layers. For convenience, the largest number represented by these N_{n-1} bits is approximated by $\Delta_{n-1}2^{N_{n-1}-1}$.

Suppose the transfer function is:

$$y = f(x) = \begin{cases} \Delta_n 2^{N_n-1} & g_n * x > \Delta_n 2^{N_n-1} \\ g_n * x & \text{In-between} \\ -\Delta_n 2^{N_n-1} & g_n * x < -\Delta_n 2^{N_n-1} \end{cases}$$

Where g_n is the gain of the neuron.

Inputs and weights are assumed uniformly distributed in $[-\Delta_{n-1}2^{N_{n-1}-1}, \Delta_{n-1}2^{N_{n-1}-1}]$, the variance of y_i^{n-1} is [4]:

$$\begin{aligned} \sigma^2_{y_i^{n-1}} &\stackrel{\text{def}}{=} E\{(y_i^{n-1})^2\} \\ &= \sum_{k=0}^{K_{n-1}-1} E\{(x_k^{n-1})^2 (w_{ik}^{n_1})^2\} \\ &= (K_{n-1}\Delta_{n-1}^4 2^{4N_{n-1}})/144 \end{aligned} \tag{2}$$

We define:

$$\begin{aligned} \max|y_i^{n-1}| &= \sqrt{3\sigma^2_{y_i^{n-1}}} \\ &= \sqrt{(3K_{n-1}\Delta_0^4 2^{4N_{n-1}})/144} \\ &= (\sqrt{K_{n-1}}\Delta_0^2 2^{2N_{n-1}})/(4\sqrt{3}) \end{aligned} \tag{3}$$

Although the absolute value of y_i^{n-1} may be larger than $max|y_i^{n-1}|$, we can still reasonably use $max|y_i^{n-1}|$ as the maximum of $|y_i^{n-1}|$ since more than 90% of y_i^{n-1} falls in the region of $[-max|y_i^{n-1}|, max|y_i^{n-1}|]$ if y_i^{n-1} obeys a normal distribution.

The gains should be initialized, so that after the transfer function, output of a neuron for any of its input are not in the nonlinear region and is strong enough to reflect changes in the weights. So we have:

$$\Delta_n \leq g_n * max|y_i^{n-1}| \leq \Delta_n 2^{N_n-1} \tag{4}$$

Or:

$$g_n = q\frac{\sqrt{3}\Delta_n 2^{N_n-1}}{\sqrt{k_n}(\Delta_{n-1}2^{N_{n-1}-1})^2} \tag{5}$$

Where $q \in [\frac{1}{2^{N_n-1}}, 1]$.

It is easy to see that Equation 5 can be rewritten as:

$$g_n = q\frac{\sqrt{3}O_{max}}{\sqrt{k_n}I_{max} * W_{max}} \tag{6}$$

Where O_{max}, I_{max} and W_{max} are the maximums of the outputs, inputs and the weights respectively.

In the simulations below, the gains are initialized with q being 0.2 and all neurons in the same layer having the same initial gains.

3.1.2 Gain Adaptation Rule

During training, if there is at least one weight to the inputs of a layer reaching the bounds of the dynamic range, the gain of the neurons in that layer are multiplied by a constant larger than one. We call this constant the "gain adjustment coefficient" (L in Figure 3), and it is set to 1.5 in our computer simulations except where specified otherwise. In parallel with the adjustment of the gain of the neurons, all the weights to the inputs of that layer are scaled down accordingly, to maintain unchanged the outputs of that layer. Obviously, the gains are inherently quantized.

Another strategy would be to adjust the gains individually. In this case, the training algorithm may make a better use of the available resolution, especially if the variations of the weights across the neurons are large.

3.2 Modified Weight Perturbation

In Modified Weight Perturbation, the learning rate (see Equation 1) is eliminated and only the sign of $(Error(W_{ij}+\Delta W_{ij})-Error(W_{ij}))/\Delta W_{ij}$ is used to indicate a single positive or negative increment to the weight W_{ij} being modified. The weight update equation is then:

$$W_{ij}^{new} = W_{ij}^{old} + \Delta W_{ij}\text{sign}((Error(W_{ij}) - Error(W_{ij} + \Delta W_{ij}))/\Delta W_{ij}) \tag{7}$$

If the absolute values of weight increments are fixed, the convergence rate may be slower than normal WP as the weight modifications are weaker. However, the convergence of MWP would be accelerated if the increments are initially set to a large value and gradually reduced to the minimum (value of the weight least significant bit).

3.3 Partial Random Search

When, during training, the output errors cannot further be reduced using MWP, this indicates either that the learning process is trapped in a plateau or that a local minimum has been reached. In such cases non-local search techniques have to be used. Our algorithm uses Partial Random Search to escape the plateau or the local minimum. In the Partial Random Search, one weight at a time is replaced by a random value uniformly generated in the predetermined range. If the output error is reduced, the new weight is adopted, otherwise it is rejected and the original weight is restored. Therefore, in each weight update iteration, the search is performed along one dimension of the weight space (and hence the Partiality) but is not limited to the vicinity of the current point in the weight space. Partial Random Search guarantees a certain probability of escaping a local minimum or a plateau at the cost of a slower algorithm (especially if a satisfactory minimum is nearby).

Note in PRS, several weights may be randomly changed at once. In our algorithm however we deal with one weight at a time in order to maintain compatibility with MWP and for the convenience of hardware implementation.

3.4 Implementation of the Combined Search Algorithm

The Combined Search Algorithm applies alternately Partial Random Search and Modified Weight Perturbation for a number of epochs until the convergence criterion is reached.

3.4.1 Search Ranges in the Partial Random Search Algorithm

When, during training, the Partial Random Search Algorithm is applied, random values are generated uniformly in a predetermined range which we refer to as the "search range" in this paper. The determination of this search range is key to successful training using PRS.

In unlimited precision situations, PRS would start with a small search range. As the training proceeds, the weights are changed and the search range is adapted as to maintain it larger than any weight value. The adaptation of the search range during training prevent the weights taking too large values and saturating the network [8].

However, search range adaptation can be achieved indirectly using neuron gain adjustment, a technique that we adopt in our approach. Gain adaptation has been described in Section 3.1 above and is simpler for hardware implementation.

3.4.2 Balance Between PRS and MWP in the Combined Search Algorithm

Another issue in the implementation of the Combined Search algorithm is how to alternate between PRS and MWP during training. The CS strategy used in our simulations alternates between PRS and MWP every 20 training epochs (MaxNepochs is 20). However, when MWP cannot further reduce the output error, the CS algorithm terminates the current MWP cycle and starts the PRS.

4 Experiments

4.1 Network Model and Quantization Method

In the following experiments, only multi-layer feedforward architectures are used. In each architecture, the output of a neuron in the is connected only to the neurons in the next layer. The transfer functions of the hidden and output neurons are sigmoid for the back-propagation algorithm (in order to provide differentiability) The following transfer function is used for the other algorithms:

$$y = f(x) = \begin{cases} 1 & x > 1 \\ x & \text{In-between} \\ 0 & x < 0 \end{cases}$$

All the desired outputs in the following experiments are either 1 or 0. After the network is trained, the output corresponding to an input pattern is considered correct if every bit of the output is within 50% of the desired output.

In the simulations with limited precision, all the input data, weights and neuron states are quantized in a fixed point format in order to fully investigate the performance of digitally implemented networks. Following each update, a value is cast to limited precision by rounding off the value below the precision. The weights of the networks are initialized to random values uniformly distributed in the dynamic range represented by the given number of bits. The specified numbers of bits include the sign bit.

4.2 ICEG Pattern Recognition

The first set of experiments addresses the problem of IntraCardiac ElectroGram (ICEG) classification. Each pattern has 21 inputs. The training set has 278 patterns and the testing set consists of 2463 patterns. All the patterns have to be classified into three classes: ventricular tachycardia (VT); ventricular fibrillation (VF); normal sinus rhythm (NSR). Therefore, the network has three outputs, each representing one of the classes.

The first architecture is a three layer network with 21 input, 10 hidden units and 3 outputs. Ten training trials are performed using different algorithms and various sets of initial conditions. For each trial the same set of initial conditions is used for each of the training algorithms, and every training pattern is applied 900 times. If the performance on the testing set is less than 50% correct, the training is labeled as a "fail" in all the results below.

Figure 3 shows a trial training curves for the five algorithms (with 10 bit resolution). Table 1 is the summary of the training results.

In Table 1, the first two rows show the training results on unlimited precision networks trained using a conjugate gradient algorithm [9] and the Combined Search algorithm.

In the limited precision case, Back-Propagation failed to converge with 10 bit resolution although many different values for the learning rate and momentum were tried.

It is found that Modified Weight Perturbation and Weight Perturbation did not always converge. Out of 10 trials, MWP failed three times and WP once. Therefore, in Table 1, only the best results of these two training algorithms are presented.

The PRS and CS algorithms have successfully trained the network in each of the 10 trials. The CS algorithm, however, achieves the best generalisation results on average (which are quite

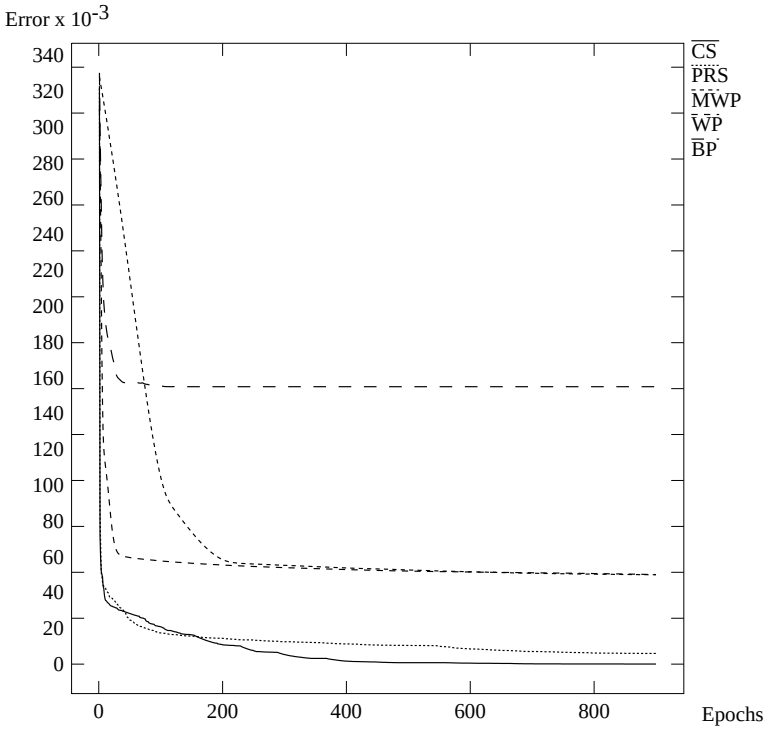

Figure 3: Training curves for ICEG data with 10 bit resolution and one hidden layer

Table 1: Training results on ICEG data with 10 bit resolution and one hidden layer

Training Algorithm	Average Performance on Training Sets (%)	Standard Deviation on Training Sets (%)	Average Performance on Testing Sets (%)	Standard Deviation on Testing Sets (%)
Unlimited OPT	99.2	0.248	91.4	0.609
Unlimited CS	98.9	0.543	90.8	0.785
CS	98.7	0.712	90.0	1.09
PRS	98.3	0.796	89.0	1.17
MWP	88.5(best)		85.2(best)	
WP	97.4(best)		88.4(best)	
BP	fail			

close to the results of the unlimited CS training).

When the resolution is reduced by one bit, we obtain the performance shown in Table 2 and Figure 4. Now, both MWP and WP fail seven out of the ten trials, while the results produced by CS are still satisfactory and are superior to those achieved by PRS.

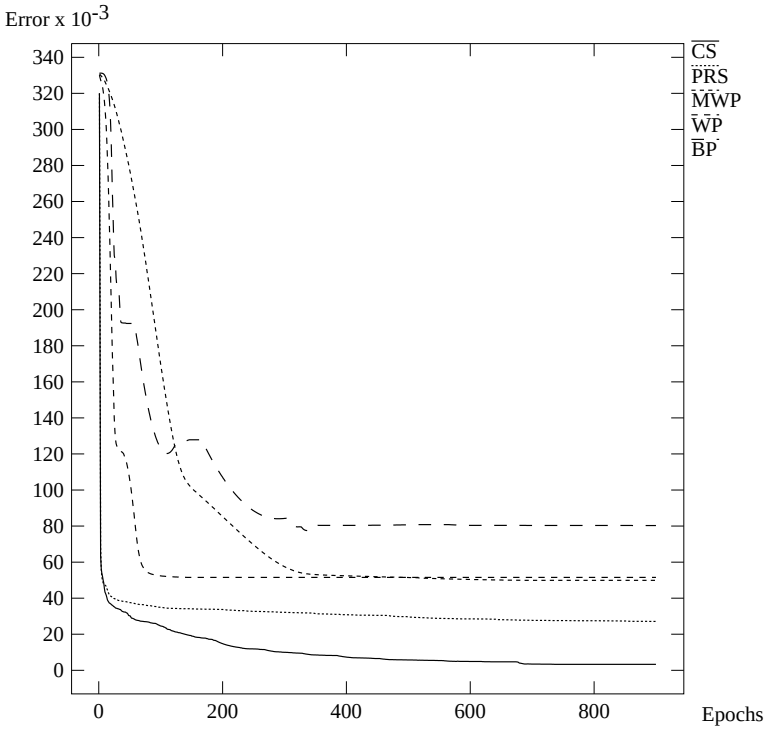

Figure 4: Training curves for ICEG data with 9 bit resolution and one hidden layer

Table 2: Training results on ICEG data with 9 bit resolution and one hidden layer

Training Algorithm	Average Performance on Training Sets (%)	Standard Deviation on Training Sets (%)	Average Performance on Testing Sets (%)	Standard Deviation on Testing Sets (%)
CS	98.4	0.771	89.6	0.978
PRS	98.0	0.856	88.7	1.14
MWP	87.1(best)		83.7(best)	
WP	88.4(best)		85.3(best)	
BP	fail			

The other architecture used for the ICEG classification problem is a four layer net with two hidden layers, each with 5 hidden units. The simulation results are shown in Table 3 where 9 bits are used for quantization.

As the number of hidden layers in the network is increased, all the local search algorithms fail. CS, however, continues to perform almost as well as its unlimited precision version and is superior than PRS. The average performance of CS and PRS are close but the former has a

Table 3: Training results of CS on ICEG data with 9 bit resolution and two hidden layers

Number of Bits	Average Performance on Training Sets (%)	Standard Deviation on Training Sets (%)	Average Performance on Testing Sets (%)	Standard Deviation on Testing Sets (%)
unlimited OPT	98.9	0.215	91.2	0.550
unlimited CS	98.4	0.612	89.9	1.01
CS	98.4	0.746	89.3	1.13
PRS	98.3	0.780	89.1	2.18
MWP	fail			
WP	fail			
BP	fail			

smaller standard deviation.

4.3 Sonar Reflection Recognition

The second set of simulations uses Sonar Reflection Recognition [10]. An input pattern represents a reflection of sonar signals from a metal cylinder or a rock at various angles and under various conditions. The network is expected to classify a pattern as a reflection from the metal cylinder or from the rock. Altogether 208 patterns are available and out of which 111 are reflections from the metal cylinder and the rest are from the rock. As in [10], we use 195 patterns for training and the remaining 13 for testing. Each pattern is a vector of 60 features.

The architecture used in the simulation is a 60 input neurons, 6 hidden units and 2 output units network. The number of quantization bits is 9 for the inputs, weights and neuron outputs.

The network is trained for 15 trials with various sets of initial weights and for each trial 195 patterns are randomly selected for the training and the remaining patterns are used for testing. The "angle" information is not used in our simulations. Each training pattern is applied to the network 1000 times before the training is stopped.

Because of the time-consuming nature of the simulations, only the Partial Random Search and the Combined Search algorithms have been used. Figure 5 and Table 4 show the training curves and performance, respectively.

In order to avoid overtraining, unlimited CS was run for 300 epochs in each trial.

For comparison purposes, the performance achieved by the same architecture as reported in [10] is shown in Table 4. Two differences need to be noted: 1- in the present work 195 patterns were randomly selected for each training trial which may be different from [10]; 2- the results in Table 4 were averaged over 15 training trials.

The results again prove the effectiveness of the Combined Search Algorithm and its advantages over the Partial Random Search Algorithm, especially when the standard deviation is concerned.

The computer simulations on ICEG and sonar reflection pattern recognition clearly suggest that the Combined Search algorithm is an effective algorithm for training limited precision

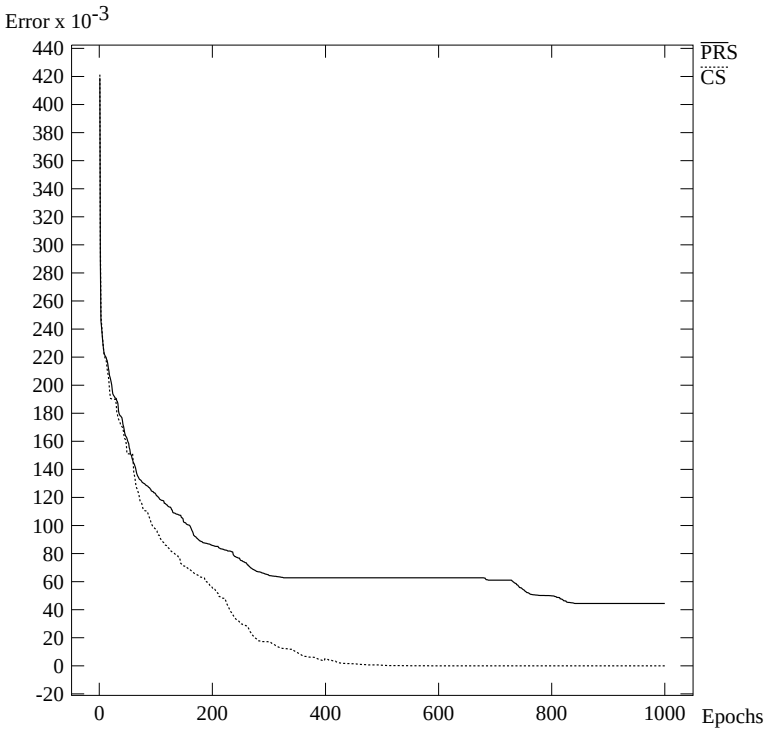

Figure 5: Training curves for the classification of sonar reflections with 9 bit resolution and 6 hidden units

Table 4: Training performance for the classification of sonar reflections with 9 bit resolution and 6 hidden units

Training Algorithm	Average Performance on Training Sets (%)	Standard Deviation on Training Sets (%)	Average Performance on Testing Sets (%)	Standard Deviation on Testing Sets (%)
unlimited CS	99.7	0.370	78.5	10.6
CS	98.9	0.868	82.6	12.2
PRS	89.6	2.31	81.5	12.9
¿From [10]	99.7	0.2	83.5	5.6

multi-layer feedforward neural networks. The Partial Random Search and the Modified Weight Perturbation algorithms do compensate for each other's limitations when combined in the Combined Search algorithm.

4.4 Convergence Speed and Gain Adjustment

The convergence speed of CS and PRS are compared on the training of a one hidden layer architecture of 10 hidden units and 9 bit quantization on the ICEG problem. The training is terminated either when the average output error was below 6×10^{-3} or when the number of training epochs reach 1000.

Out of 10 trials, CS stopped training 8 times within 1000 epoches and the average training epochs of these 8 training sessions is 463; PRS failed to reduce the error below 6×10^{-3} within 1000 epoches. This indicates that CS is more efficient in searching for minima than PRS in terms of convergence speed.

As mentioned in Section 3.1, the gain of the neurons in the network are adjusted during training, and it was shown that the Combined Search algorithm is not sensitive to the initial values of the gains. The experiments show that as long as the initial gain values are 1 to 2 orders of magnitude smaller than the final gains, there is no obvious difference in the training results. This suggests that CS is robust with respect to the gain although the final gains of the neurons in a network are somewhat problem–dependent. This also indicates that if the gain adjustment coefficient is set to 2, 6 to 7 bits are sufficient for its control.

The gain adjustment coefficient, which was set to 1.5 in most of the computer simulations in this paper, is independent of the problem to be solved by the network. A too large value can make the training unstable, especially when low precision is used. The reason is that when the gains are adjusted, although the weights are scaled down accordingly at the same time, the mapping implemented by the network may be altered due to the limited precision. This phenomenon can be seen clearly from the sudden error increases in Figure 6. If the gain adjustment coefficient is very close to 1, the training becomes slow since the growth margin in the PRS's search range is too small for large weights.

In our simulations, it is found that 1.5 is a satisfactory value for the gain adjustment coefficient. Values of 1.2 and 2 were also tried on the ICEG data. Table 5 shows the training results for the single hidden layer architecture with 10 hidden units, where the CS was used and each training session was run for 900 epochs.

Table 5: Training results on ICEG data with 9 bit resolution and one hidden layer

Gain Adjustment Coefficient	Average Performance on Training Sets (%)	Standard Deviation on Training Sets (%)	Average Performance on Testing Sets (%)	Standard Deviation on Testing Sets (%)
1.2	97.3	0.644	89.9	0.935
1.5	98.4	0.771	89.6	0.978
2.0	98.3	0.614	89.7	1.03

The results show that CS gives almost the same performance when the gain adjustment coefficient varies from 1.2 to 2. If it is set to 2 and a binary representation is used, the gain adjustment can be realized just by a "shift" operation.

4.5 The Influence of Bit Resolution on the Performance of the Network

Using the ICEG classification problem and the three layer architecture used in the previous sections, another set of simulations was conducted in which the number of quantization bits was varied.

Table 6 and Figure 6 show the training performance and curves, respectively. The Combined Search is used in this experiment.

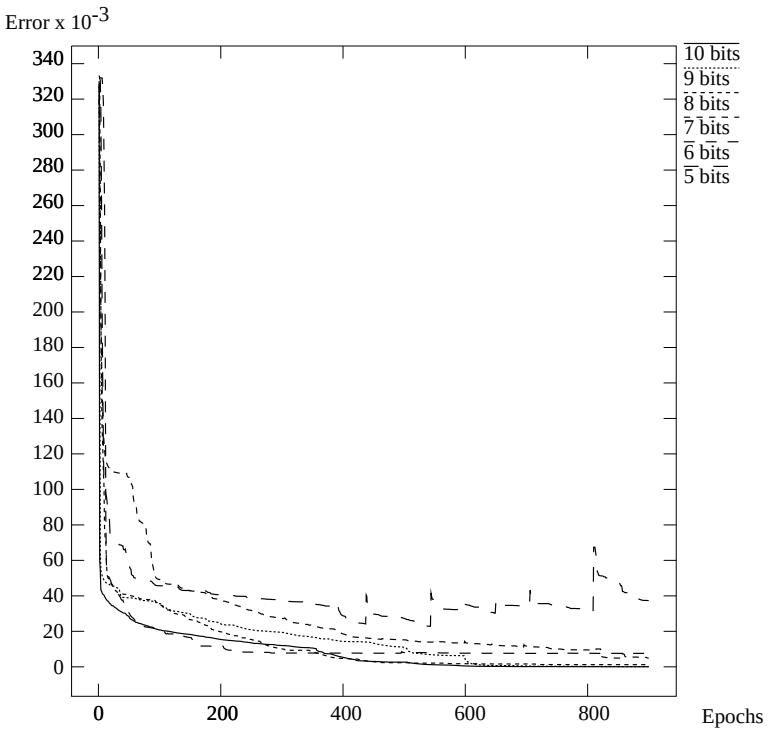

Figure 6: Training curves of CS on ICEG data with one hidden layer

As the number of bits decreases, the performance of the network deteriorates much faster than linearly.

What is not shown in Table 6 is that as the number of bits decreases, CS relies increasingly on the Partial Random Search algorithm during training, while the performance of Modified Weight Perturbation algorithm degrades until its contribution becomes marginal. The reason is that as less bits are used, both the number and the scale of the planes in the error function caused by the quantization increase.

Another interesting phenomenon is that as the number of bits is reduced, an increasing number of hidden units became irrelevant after the network is trained—their outputs are either zero or one for all the input patterns. This is due to the fact that as less bits are used, less information can be extracted by the hidden units from the input data.

Table 6: Training results of CS on ICEG data with one hidden layer

Number of Bits	Average Performance on Training Sets (%)	Standard Deviation on Training Sets (%)	Average Performance on Testing Sets (%)	Standard Deviation on Testing Sets (%)
10	98.7	0.712	90.0	1.09
9	98.4	0.771	89.6	0.978
8	98.5	0.740	89.2	1.46
7	97.5	2.24	87.7	5.29
6	95.7	2.70	86.4	5.85
5	89.2	8.80	81.2	8.63

5 Hardware Implementation of the Combined Search Algorithm

It is shown in [6] that in terms of chip area and hardware complexity, WP is ideal for on-chip training. In addition to the components required for the normal operation of a network, WP requires the following three modules (for on-chip learning):

- Perturbation Generation;
- Error Difference Evaluation;
- Weight write;

Though the analysis in [6] is done for analog VLSI, it applies to digital VLSI as well.
For the Conbimed Search algorithm two parts are required in addition to the three above:

1. a random number generator which can be implemented as a shift register with feedback loops.
2. a gain adjustment module which can be simplified when the gain adjustment coefficient is 2.

6 Conclusions

An efficient and effective limited precision training algorithm is presented in this paper. It combines the Modified Weight Perturbation and the Partial Random Search. A detailed description of training characteristics and comparative simulation results of limited precision multi-layer feedforward neural networks on two pattern recognition problems are also presented along with preliminery testing results on an analog neural network chip.

The following conclusions can be drawn:

- Training algorithms that are effective and efficient for unlimited precision networks may not be suitable for limited precision cases. Some need very high precision computation and are hard to be implemented in hardware.

- The proposed Combined Search Algorithm is an effective training algorithm for limited precision feedforward neural networks. It is simple to use, has stable performance and economic for hardware implementation. In the Combined Search Algorithm the Partial Random Search and the Modified Weight Perturbation algorithms do compensate for each other's limitations. Both the training performance and the convergence speed are improved as a result. Although the simulations were conducted only on multi-layer feedforward neural networks, basically there is no difficulty in applying the Combined Search Algorithm to other types of supervised network models.

- Gain adjustment is essential in training limited precision neural networks. The simulation results show that 6 to 7 bits are sufficient for the control of the gains.

In our experiments, only one kind of combinations of local and non-local search methods are studied. In fact, it is possible to combine various local and non-local search methods in different ways in order to meet various requirements in performance and implementation. The applications of such algorithms are not restricted to limited precision neural network training [8].

7 Acknowledgment

This work was supported in part by the Australian Research Council and the Australian Telecommunication and Electronic Research Board. One of us (YX) acknowledges the support of National Education Committee of P.R.China.

References

[1] J. L. Holt and J. Hwang. Finite precision error analysis of neural network hardware implementation. *Technical report: FT-10, Dept. of E.E., University of Washington*, 1991.

[2] M. Hoehfeld and S. Falhman. Learning with limited numerical precision using the cascade-correlation algorithm. *CMU-CS-91-130, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pa 15213*, May 3, 1991.

[3] P. W. Hollis, J. S. Harper, and J. J. Paulos. The effects of precision constraints in a backpropagation learning network. *Neural Computation*, 2(3), 1990.

[4] Y. Xie and M. A Jabri. Analysis of effects of quantization in multilayer neural networks using statistical model. *IEEE Transactions on Neural Networks*, 3(2), March, 1992.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propogation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing:Explorations in the Microstructure of Cognition*. MIT Press, 1986.

[6] M. A. Jabri and B. Flower. Weight perturbation: An optimal architecture and learning technique for analog vlsi feedforward and recurrent multi-layer networks. *IEEE Transactions on Neural Networks*, 3(1), 1992.

[7] J. K. Kruschke and J. R. Movellan. Benefits of gain: Speeded learning and minimal hidden layers in back-propagation networks. *IEEE Transactions on Systems, Man and Cybernetics*, 21(1), 1991.

[8] P. D. Wasserman. *Neural Computing: Theory and Practice*. Van Nostrand Reinhold, New York, 1989.

[9] E. Barnard and R. Cole. A neural-net training program based on conjugate-gradient optimization. *OGC Tech. Report No.CSE 89-014, Carnegie Mellon University*, 1989.

[10] R. P. Gorman and T. J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1:75–89, 1988.