

# Parameter Quantization Effects in Gaussian Potential Function Neural Networks

ERKAN KARAKUŞ<sup>(1)</sup>, ARIF SELÇUK ÖĞRENCİ<sup>(2)</sup>, GÜNHAN DÜNDAR<sup>(1)</sup>

<sup>(1)</sup>Department of Electrical and Electronics Engineering  
Boğaziçi University  
80815-Bebek, Istanbul  
TURKEY

<sup>(2)</sup>Department of Electronics Engineering  
Kadir Has University  
Vefabey Sok. 5, 80810-Gayrettepe, Istanbul  
TURKEY

*Abstract:* - In hardware implementations of Gaussian Potential Function Neural Networks (GPFNN), deviation from ideal network parameters is inevitable because of the techniques used for parameter storage and implementation of the functions electronically, resulting in loss of accuracy. This loss in accuracy can be represented by quantization of the network parameters. In order to predict this effect, theoretical approaches are proposed. One-input, one-output GPFNN with one hidden layer have been trained as function approximators using the Gradient Descent algorithm. After the training, the network parameters (means and standard deviations of the hidden units and the connection weights) are quantized up to 16-bits in order to observe the percentage error on network output stemming from parameter quantization. Simulation results are compared with the predictions of the theoretical approach. Consequently, the behaviour of the network output has been given with combined and separate parameter quantizations. Moreover, given the allowed percentage error for the network, a method is proposed where the minimum number of bits required for quantization of each parameter could be determined based on the theoretical predictions.

*Key-Words:* - Gaussian potential function neural networks, weight quantization, training.

## 1 Introduction

One of the challenging problems encountered in the design of neural network circuitry is the storage of parameters. Due to the techniques used for parameter storage such as digital memories, capacitors, and transistors, deviation from ideal network parameters is inevitable. As an example, quantization of ideal network parameters for easier storage may lead to a remarkable level of inaccuracy in hardware implementation of neural networks. Learning algorithms that are efficiently implemented on general-purpose digital computers encounter great difficulties while mapped onto VLSI hardware: deviations of parameters may significantly affect the performance of the trained system unless the more costly “chip in the loop” training is performed. A serious problem here is to predict the upper bound of inaccuracy to ensure a certain performance during the training stage. The remedy is to investigate the parameter quantization effects on neural network output and to provide an analytic solution based on

probabilistic analysis for prediction of these effects. Such an analysis for multilayer perceptron neural networks has been carried out by several researchers [1, 2]. In this work, parameter quantization effects will be investigated and predicted for GPFNN.

GPFNN are considered in virtue of their wide usage for both supervised and unsupervised types of learning. They are used for function approximation where the highly nonlinear Gaussian basis functions provide good locality for incremental learning. GPFNN are also used where a similarity measure has to be computed such as in classification and self-organization problems. Section 2 investigates the parameter quantization effects for a specific network, and a theoretical approach based on a probabilistic analysis is provided in order to predict network error resulting from parameter quantization. Next, results obtained for both experimental and theoretical methods are given where the network parameters (mean and standard deviation parameters of Gaussian activation function and the weights on the

output synapses,) are quantized up to 16-bits to observe the network performance. In the conclusion, results are summarized and an optimization method is proposed, in which, given the percentage error, the optimum number of bits for each parameter quantization is found using the predictions of the theoretical approach.

## 2 The Effects of Neural Network Parameter Quantization

A single input - single output Gaussian Potential Function Network (GPFN) is considered for the analysis of parameter quantization on network performance. The Gradient Descent algorithm is used to train the network as a function approximator. After the training, the network parameters (which are the means and standard deviations of the hidden units and the connection weights) are quantized to observe the error on network output stemming from parameter quantization. This is performed separately on all parameters first, then together. Then, theoretical and experimental percentage error (PE) on the output is calculated for each parameters quantization. For this purpose, a theoretical approach will be derived to predict the network error resulting from parameter quantization.

In the following,  $x_1^0$  is the input to the neural network,  $y_i^0$  is the input to each hidden unit,  $x_i^1$  is the output of each hidden unit,  $w$  are the weights,  $y^1$  is the output of the neural network. The weights on the synapses connecting the input node to hidden nodes are equal to one, hence

$$y_i^0 = x_1^0 \quad (1)$$

$$x_i^1 = f(y_i^0) = e^{-(y_i^0 - m_i)^2 / s_i^2} \quad (2)$$

Then, the network output is,

$$y^1 = \sum_{i=0}^{N-1} w_{i1} x_i^1 \quad (3)$$

The experimental percentage error calculation will be carried out by the following formula,

$$PE = \frac{\sigma_{noise}}{\sigma_{signal}} \quad (4)$$

where  $\sigma_{noise}$  is the standard deviation of the quantization error which is considered as a noise on desired signals.

In the following sections, derivation of the standard deviation of the error and of the desired output will be given for each parameter perturbation of the network, namely the mean, standard deviation, and weights.

### 2.1 Weight Quantization Effects

The weight values are assumed to have a uniform distribution between  $-max(w)$  and  $max(w)$  where  $max(w)$  is the maximum weight value obtained from simulation results [1, 2]. Distribution parameters of connection weights are found as follows [2]:

$$m_w = 0 \quad (5)$$

$$\sigma_w^2 = \frac{1}{2 \max(w)} \int_{-\max(w)}^{\max(w)} w^2 \cdot dw = \frac{\max(w)^2}{3} \quad (6)$$

The weight quantization effect on the network output can be expressed by the following:

$$y^1 = \sum_{i=0}^{N-1} (w_{i1} + \Delta w_{i1}) x_i^1 \quad (7)$$

and the error on the network output becomes,

$$\Delta y^1 = \sum_{i=0}^{N-1} \Delta w_{i1} x_i^1 \quad (8)$$

where  $\Delta w_{i1}$  is the quantization error (noise) of the weights. Quantization error has a uniform probability distribution between  $-\Delta_w/2$  and  $+\Delta_w/2$

where  $\Delta_w = \frac{2w_{max}}{2^B - 1}$  is the quantization increment

with  $w_{max}$  being the upper bound of the quantization interval and  $B$  is the number of bits used in quantization. The distribution parameters of quantization error are found as follows [2]:

$$m_{\Delta w} = 0 \quad (9)$$

$$\sigma_{\Delta w}^2 = \frac{1}{\Delta_w} \int_{-\frac{\Delta_w}{2}}^{\frac{\Delta_w}{2}} \Delta^2 d\Delta = \frac{\Delta_w^2}{12} \quad (10)$$

The variance of the error in network output is

$$V(\Delta y) = V(\Delta w) \sum_{i=0}^{N-1} E(x_i^1)^2 + E(\Delta w)^2 \sum_{i=0}^{N-1} V(x_i^1) \quad (11)$$

Since  $E(\Delta w) = 0$ , the variance of the error becomes,

$$V(\Delta y) = V(\Delta w) \sum_{i=0}^{N-1} E(x_i^1)^2 \quad (12)$$

Similarly, variance of the actual network output is,

$$V(y_{real}^1) = V(w) \sum_{i=0}^{N-1} E(x_i^1)^2 + E(w)^2 \sum_{i=0}^{N-1} V(x_i^1) \quad (13)$$

Since  $E(w) = 0$ , then

$$V(y_{real}^1) = V(w) \sum_{i=0}^{N-1} E(x_i^1)^2 \quad (14)$$

The percentage error due to weight quantization can be expressed as

$$(PE)_w = \sqrt{\frac{V(\Delta y)}{V(y_{real}^1)}} = \sqrt{\frac{V(\Delta w)}{V(w)}} \quad (15)$$

and using (6) and (10), PE is found as follows:

$$(PE)_w = \frac{w_{max}}{\max(w)(2^B - 1)} \quad (16)$$

## 2.2 Mean Value Quantization Effects

The input of the network is assumed to be time, varying in the interval  $[0, 2L]$ , where  $2L$  is the period in which the function is to be approximated by the neural network. Hence, it is conceivable that the network input and mean values of the hidden units will have a uniform probability distribution between 0 and  $2L$  (this assertion will be modified in Section 3, in order to get an upper bound estimate for the quantization effect): mean and standard deviation of the network input and mean (center values) above can easily be found as:

$$m_{x_i^0} = m_m = L \quad (17)$$

$$\sigma_{x_i^0}^2 = \sigma_m^2 = \frac{1}{2L} \int_0^{2L} (m-L)^2 dm = \frac{L^2}{3} \quad (18)$$

and distribution parameters of quantization error,  $\Delta m$ , are similar to those of  $\Delta w$ .

In order to eliminate the exponential term in the Gaussian activation function for easier mathematical manipulation, Taylor series expansion of  $e^a$  around  $a=0$  will be used, then the activation function becomes

$$f(y_i^0) = 1 - \frac{(y_i^0 - m_i)^2}{s_i^2} + \frac{(y_i^0 - m_i)^4}{2s_i^4} \quad (19)$$

The mean quantization effect on the network output can be expressed in the following form:

$$y^1 = \sum_{i=0}^{N-1} w_i \cdot (x_i + \Delta x_i) \quad (20)$$

Finally one finds the error in the network output as:

$$\Delta y^1 = \sum_{i=0}^{N-1} w_i \left\{ \begin{array}{l} 1 - \frac{(y_i^0 - (m_i + \Delta m_i))^2}{s_i^2} + \frac{(y_i^0 - (m_i + \Delta m_i))^4}{2s_i^4} \\ -1 + \frac{(y_i^0 - m_i)^2}{s_i^2} - \frac{(y_i^0 - m_i)^4}{2s_i^4} \end{array} \right\} \quad (21)$$

Variance of a function of  $n$  independent random variables, say  $Z = H(X_1, \dots, X_n)$  with  $E(X_i) = \mu_i$  and  $V(X_i) = \sigma_i^2$ , is given by the following:

$$V(Z) \approx \sum_{i=1}^n \left( \frac{\partial H}{\partial X_i} \right)^2 \sigma_i^2 \quad (22)$$

where all the partial derivatives are evaluated at the point  $(\mu_1, \dots, \mu_n)$ . By applying (22), the standard deviation of the error becomes,

$$\sigma_{\Delta y^1} = \sqrt{V(\Delta y^1)} = N \cdot w \left[ \frac{2}{s} (L - \bar{m}) - \frac{2}{s} (L - \bar{m})^3 \right] \frac{\Delta m}{\sqrt{12}} \quad (23)$$

where parameters with overbars indicate the variables' expected values. In Section 3, it will be shown how to determine the expected values.

According to (14),  $E(x_i^1)$  is required to be calculated in order to find the variance of the output obtained without parameter quantization. The expected value of hidden unit output,  $E(x_i^1)$ , is found according to the approach derived in [2]. First, the cumulative distribution function for  $x$  is derived. Then the pdf of  $x$  is found and the expected value of  $x$  becomes,

$$E[x] = \frac{s}{4L} \int_0^1 \frac{1}{\sqrt{-\ln x}} dx \quad (24)$$

After some mathematical manipulation this yields

$$E[x] = \frac{s\sqrt{\pi}}{4L} \quad (25)$$

Then, substituting (25) into (14), the standard deviation of the output becomes,

$$\sigma_{y_{real}^1} = \sigma_w \sqrt{\sum_{i=0}^{N-1} \left( \frac{s_i \sqrt{\pi}}{4L} \right)^2} \quad (26)$$

Finally, by substituting (26) into (4), percentage error (PE) is found as follows,

$$(PE)_m = \frac{\frac{\Delta m}{\sqrt{3}} N \cdot w \left[ \frac{1}{s} (L - \bar{m}) - \frac{1}{s} (L - \bar{m})^3 \right]}{\sigma_w \sqrt{\sum_{i=0}^{N-1} \left( \frac{s_i \sqrt{\pi}}{4L} \right)^2}} \quad (27)$$

## 2.3 Quantization Effects of Standard Deviation

Similar to mean quantization, quantization effect of standard deviation on the network output can be expressed in (16) and distribution parameters of quantization error,  $\Delta s$  are similar to those of  $\Delta w$ .

By using the Taylor approximation, error on the network output can be expressed by,

$$\Delta y^1 = \sum_{i=0}^{N-1} w_i \left\{ \begin{array}{l} \frac{(y_i^0 - m_i)^2}{s_i^2} - \frac{(y_i^0 - m_i)^2}{(s_i + \Delta s_i)^2} + \\ \frac{(y_i^0 - m_i)^4}{2(s_i + \Delta s_i)^4} - \frac{(y_i^0 - m_i)^4}{2s_i^4} \end{array} \right\} \quad (28)$$

and by applying (22), the standard deviation of error on the network output is found as,

$$\sigma_{\Delta y^1} = \sqrt{V(\Delta y^1)} = N \cdot w \left[ \frac{2}{s} (L - \bar{m})^2 - \frac{2}{s} (L - \bar{m})^4 \right] \frac{\Delta s}{\sqrt{12}} \quad (29)$$

Finally, by substituting the equations (26) and (29) into (4), percentage error (PE) is found as follows,

$$(PE)_s = \frac{\frac{\Delta_s}{\sqrt{3}} N \cdot w \left| \frac{1}{s} (L - \bar{m})^2 - \frac{1}{s} (L - \bar{m})^4 \right|}{\sigma_w \sqrt{\sum_{i=0}^{N-1} \left( \frac{s_i \sqrt{\pi}}{4L} \right)^2}} \quad (30)$$

When all parameters, mean, standard deviation and weights, are quantized together, each quantization will contribute to the error on the network independent of each other [1]. Hence, a worst case (upper bound) estimate of the percentage error caused by quantization of all parameters will be the sum of percentage errors caused by quantization of each parameter:

$$PE \leq (PE)_w + (PE)_m + (PE)_s \quad (31)$$

### 3 Simulation Experiments

After having derived the theoretical methods for the prediction of network parameter quantization effects, comparison of the theoretical results to the simulation results will be presented. The GPFNN used in the simulation has been trained as a function approximator. Simulations are carried out for several periodic ( $\sin(x)$ ,  $\sin(x) \cdot \cos(x)$ , etc. over  $[0, 2\pi]$ ) and non-periodic functions ( $f(x) = x(x-1)(x-1.5)(x-2.5)(x-3)$  over  $[0, 3]$ ) for different numbers of hidden units. In the following, results for  $\sin(x) \cdot \cos(x)$  and for the polynomial with  $N=14$  hidden units are given. The neural networks are trained by software using 32-bit precision for a rms error of less than 0.1%.

#### 3.1 Results for $\sin(x) \cdot \cos(x)$ :

For the theoretical results, two different approaches will be used for the function  $\sin(x) \cdot \cos(x)$  (and similar periodic functions). In the following, the expected values for the parameters, weight and mean, obtained from (5) and (17) will not be directly used for the terms  $\bar{w}$  and  $\bar{m}$ , since this will cause (27) and (30) to be equal to zero. It is obvious that this would happen if all the Gaussians were identical to each other and equally spaced. Instead, they will be evaluated on two different sets for the terms,  $\bar{w}$ ,  $\bar{m}$  and  $\bar{s}$  in order to compute the upper bound for the possible percentage error. In the first approach, the terms  $\bar{w}$  and  $\bar{s}$  are directly evaluated from the training results as  $\bar{s} = \text{mean}(s)$  and  $\bar{w} = \text{mean}(w)$  where  $\text{mean}(s)$  and  $\text{mean}(w)$  are the arithmetical mean of standard deviation and center values of hidden units obtained from simulation results.  $\bar{m}$  is chosen as follows: Assume that the interval of interest is covered by  $N$  Gaussians in such a way that the center

of each one coincides with its neighbouring Gaussian's tail at the level of only  $1/e^2 \approx 10\%$  of the neighbour's maximum. Hence, the maximum distance between the centers of two Gaussians becomes  $\bar{s}\sqrt{2}$ . Therefore, the maximum deviation of the average of the centers can be taken as  $\bar{s}\sqrt{2}$ , implying that

$$\bar{m} = \pi \mp \bar{s}\sqrt{2} \quad (32)$$

This value of  $\bar{m}$  also agrees well with the average of the centers obtained from simulations,  $\text{mean}(m)$ . Substituting (32) into (27) and (30) yields,

$$(PE)_m = \frac{\frac{\Delta_m}{\sqrt{3}} \cdot N \cdot w \cdot \frac{\sqrt{2}}{s}}{\frac{\max(w)}{\sqrt{3}} \sqrt{\sum_{i=0}^{N-1} \left( \frac{s_i \sqrt{\pi}}{4L} \right)^2}} \quad (33)$$

$$(PE)_s = \frac{\frac{\Delta_s}{\sqrt{3}} \cdot N \cdot w \cdot \frac{2}{s}}{\frac{\max(w)}{\sqrt{3}} \sqrt{\sum_{i=0}^{N-1} \left( \frac{s_i \sqrt{\pi}}{4L} \right)^2}} \quad (34)$$

In the second approach,  $\bar{w}$  and  $\bar{s}$  are chosen empirically. From several simulation runs with employing different number of hidden units, it has been observed that

$$\bar{w} = \frac{\max(w)}{25N} \quad (35)$$

Intuitively,  $\bar{s}$  should also be given by the formula

$$\bar{s} = k \frac{L}{N} \quad (36)$$

where  $k$  is a measure of the "complexity" of the problem. As the function exhibits more fluctuations, i.e. becomes more "complex", the standard deviations of the Gaussians have to be smaller for attaining the required locality and preserving the generalization ability of GPFNN. For the function  $\sin(x)$ ,  $k=2/3$  and for  $\sin(x)\cos(x)$ ,  $k=1/2$ . Finally,  $\bar{m}$  is calculated as given by (32).

#### 3.2 Results for the polynomial

The polynomial  $f(x) = x(x-1)(x-1.5)(x-2.5)(x-3)$  is not symmetrical with respect to any point in  $[0, 3]$ . It also has a non-zero average value, thus, the expected value of the weight values will not be zero. Then it becomes necessary to consider the variance of the hidden units' outputs in (14). We can assume, however, that the weight values are uniformly distributed between the maximum weight value,  $\max(w)$ , and the minimum weight value,  $\min(w)$ , obtained from simulation results. Then, the

distribution parameters of connection weights are found as follows:

$$m_w = \frac{\min(w) + \max(w)}{2} \quad (37)$$

$$\sigma_w^2 = \frac{(\max(w) - \min(w))^2}{12} \quad (38)$$

Using (12) and (14) percentage error for weight quantization can be expressed as

$$(PE)_w = \sqrt{\frac{V(\Delta w) \sum_{i=0}^{N-1} E(x_i^l)^2}{V(w) \sum_{i=0}^{N-1} E(x_i^l)^2 + E(w)^2 \sum_{i=0}^{N-1} V(x_i^l)}} \quad (39)$$

Since  $E(w) \neq 0$ , the variance of hidden units' output,  $V(x)$ , has to be calculated to be used in (39):

$$V(x) = \int_a^b e^{-\frac{(y-m)^2}{2\sigma^2}} - E(x) dx \quad (40)$$

where  $a$  and  $b$  are the lower and upper bounds of the interval of the function respectively. For our problem, this is obtained after some mathematical manipulation as

$$V(x) = s\sqrt{\pi} \left( erf(\sqrt{2}x_2) - erf(\sqrt{2}x_1) \right) - \frac{2\pi\sigma^2}{L} (erf(x_2) - erf(x_1)) + \frac{2\pi\sigma^2}{4L_2^2} (b-a) \quad (41)$$

where  $x_2 = (b-m)/\sigma$ ,  $x_1 = (a-m)/\sigma$ , and  $L_2 = (b+a)/2$ . The percentage error for the mean value and standard deviation quantization can be found similar to (33) and (34) where  $\bar{w}$ ,  $\bar{m}$  and  $\bar{s}$  are computed using the two approaches as in the previous example.

$$(PE)_m = \frac{\frac{\Delta_m}{\sqrt{3}} N \left| \frac{\bar{w}}{s} \frac{(L_2 - \bar{m})}{-2} \left( 1 - \frac{1}{-2} \frac{(L_2 - \bar{m})^2}{s} \right) \right|}{\sqrt{V(w) \sum_{i=0}^{N-1} E(x_i^l)^2 + E(w)^2 \sum_{i=0}^{N-1} V(x_i^l)}} \quad (42)$$

$$(PE)_s = \frac{\frac{\Delta_s}{\sqrt{3}} N \left| \frac{\bar{w}}{s} \frac{(\pi - \bar{m})^2}{-3} \left( 1 - \frac{1}{-2} \frac{(\pi - \bar{m})^2}{s} \right) \right|}{\sqrt{V(w) \sum_{i=0}^{N-1} E(x_i^l)^2 + E(w)^2 \sum_{i=0}^{N-1} V(x_i^l)}} \quad (43)$$

In the next column, effects of parameter quantization are given as the graph of the "percentage error" versus the "number of bits used", in logarithmic scale for the  $\sin(x) \cdot \cos(x)$  problem. For mean and standard deviation quantization, both theoretical approaches are compared to the simulation results. During the simulations, first, each parameter of interest is quantized separately (Figures 1, 2, 3.) Then, all parameters are quantized in Fig. 4.

All graphs displayed exponential behaviour, that is why a normalized logarithmic scale is used.

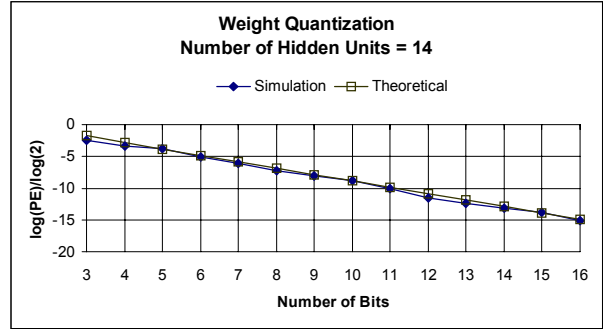


Fig. 1 Weight Quantization Error

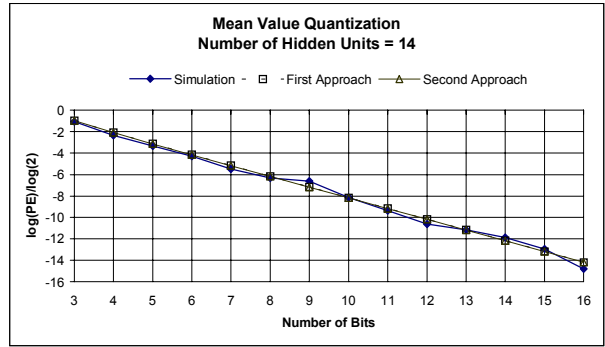


Fig. 2 Mean Quantization Error

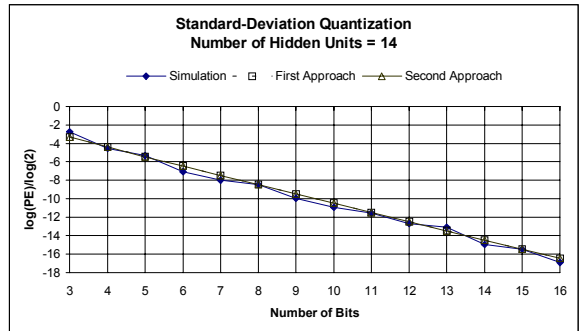


Fig. 3 Standard Deviation Quantization Error

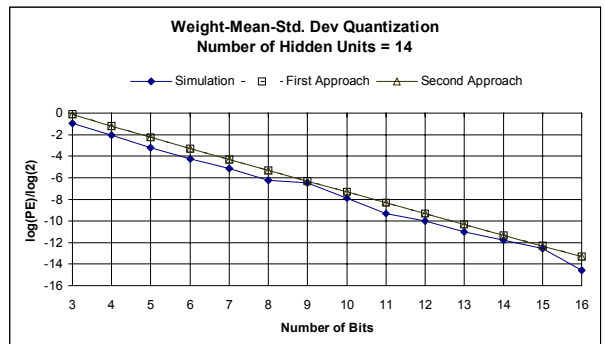


Fig. 4 Weight-Mean-Standard Deviation Quantization Error

## 4 Results and Conclusion

In this work, the GPFN was trained with the Gradient Descent algorithm for several different function approximation examples. In order to observe the network performance affected by parameter quantization, network parameters, which are the connection weights, means and standard deviation of hidden units, were disturbed with up to 16-bit quantization noise. This was performed on all parameters first separately, then together and it was observed that the performance of the resultant network was degraded by quantization noise and showed an exponential dependence on resolution. The same behaviour has also been reported in [2] for multilayer perceptron neural networks.

Two theoretical approaches are derived in order to predict the behaviour of network performance affected by parameter quantization noise based on statistical analysis of network parameters. Both approaches for theoretical predictions almost coincided. This indicates that the estimation proposed for the placement of Gaussian hidden units for the functions is in accordance with the real case. It was also seen that both theoretical approaches agreed well with simulations. It has been also shown in the simulations that the network performance exhibits exponential dependence on resolution and was degraded with decreasing resolution.

It was observed that quantization noise on connection weights and means of the hidden units plays an important role on network performance. The performance was satisfactory only with 8 or greater number of bits quantization on these parameters. On the other hand, the network performance did not degrade markedly with standard deviation quantization as compared to other parameter quantizations. Similar precision requirements are also reported in [3].

Given the required percentage error of the network, minimum number of bits for each parameter quantization can be determined based on the theoretical results. Table 1 shows an example of such determination based on the results of second theoretical approach obtained by training the network. As can be seen from Table 1, to provide the required performance, the minimum number of bits per parameter quantization can be determined in several ways. For instance, to ensure a percentage error of 2 per cent, while one set of quantization bits would be {8,8,6}, the other one would be {7,10,6}. Moreover, an optimization could be applied to the following cost function that takes the hardware area cost for each parameter into account.

$$Cost = k_1 B_w + k_2 B_m + k_3 B_s \quad (44)$$

where  $k_1$ ,  $k_2$ , and  $k_3$  are the area requirements for each weight that may depend on the implementation style. By minimizing the cost, optimum number of bits for each parameter can be found. As an example, the area requirement per bit for each parameter is the same in a DAC (Digital to Analog Converter) style parameter storage system. Hence the cost to be minimized is simply the sum of the bit counts for the parameters [4].

The main outcomes of the approach here can be stated as follows: time can be saved for predicting the effects of quantization by employing the semi-analytic approach. These results can be used in designing the architecture of the hardware. Furthermore, these results give some insight on how the quantization of the parameters affect the overall behaviour of the network. However, small modifications are necessary depending on the characteristics of the function to be approximated. As a future study, the techniques developed for predicting the network error against parameter quantization in approximation problem could be extended to other kinds of problems.

Table 1 Determination of minimum number of bits per parameter for a given percentage error for  $\sin(x)$ .

Approx. Total % Error	Weight		Mean		Standard Deviation	
	PE	No of Bits	PE	No of Bits	PE	No of Bits
~10%	1.00%	>=7	7.50%	>=5	1.50%	>=5
	4.50%	>=5	4.00%	>=6	1.50%	>=5
~2%	0.60%	>=8	0.90%	>=8	0.80%	>=6
	1.00%	>=7	0.20%	>=10	0.80%	>=6

### References:

- [1] Xie, Y. and M.A. Jabri, "Analysis of the Effects of Quantization in Multilayer Neural Networks Using a Statistical Model," *IEEE Trans. on Neural Networks*, Vol. 3, No: 2, 1992.
- [2] Dündar, G. and K. Rose, "The Effects of Quantization on Multilayer Neural Networks," *IEEE Trans. on Neural Networks*, Vol. 6, No: 6, 1995, pp. 1446-1451.
- [3] Murray, A. and L. Tarassenko, Pulse Stream VLSI for RBF Neural Networks, Final Report, <http://www.ee.ed.ac.uk/~neural/research/RBFproject/>, 1999
- [4] Çevikbaş, İ.C., A.S. Öğrenci, S. Balkır, and G. Dündar, "VLSI Implementation of GRBF Networks," *Proc. of ISCAS'2000*, Geneva, May 2000.