THE EFFECTS OF QUANTIZATION ON MULTI-LAYER FEEDFORWARD NEURAL NETWORKS

MINGHU JIANG* and GEORGES GIELEN

Department of Electrical Engineering, ESAT-MICAS, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium

*jiang.mh@tsinghua.edu.cn

In this paper we investigate the combined effect of quantization and clipping on multi-layer feedforward neural networks (MLFNN). Statistical models are used to analyze the effects of quantization in a digital implementation. We analyze the performance degradation caused as a function of the number of fixed-point and floating-point quantization bits in the MLFNN. To analyze a true nonlinear neuron, we adopt the uniform and normal probability distributions, compare the training performances with and without weight clipping, and derive in detail the effect of the quantization error on forward and backward propagation. No matter what distribution the initial weights comply with, the weights distribution will approximate a normal distribution for the training of floating-point or high-precision fixed-point quantization. Only when the number of quantization bits is very low, the weights distribution may cluster to ±1 for the training with fixed-point quantization. We establish and analyze the relationships for a true nonlinear neuron between inputs and outputs bit resolution, the number of network layers and the performance degradation, based on statistical models of on-chip and off-chip training. Our experimental simulation results verify the presented theoretical analysis.

Keywords: Quantization; weight clipping; neural networks; statistical model; noise to signal ratio.

1. Introduction

Reducing the number of quantization bits in multi-layer feedforward neural networks (MLFNN) affects either their circuit size or response time, and diversifies their application to areas such as pattern recognition or approximation of functions, which may lead to an increased interest in task-specific hardware implementation of neural networks. Generally, a smaller number of quantization bits reduces VLSI surface area, while a larger number of quantization bits gives better accuracy, but increases the required complexity and cost of the hardware as well as the response time of the neural network. Hardware implementations of neural networks are characterized by low numerical precision. However, most of the standard...
algorithms for training of MLFNN are not suitable for networks with quantization because they are based on gradient descent and require a high accuracy of the network parameters.\textsuperscript{11} Several weight-quantization techniques have been developed to reduce the required accuracy further without deterioration of the network performance.\textsuperscript{2,12,16,17} These algorithms offer hardware-friendly learning rules in case of limited accuracy and further improve the performance of neural networks.

The purpose of this work is to analyze in detail the impact of quantization on the performance of MLFNN for different probability distributions. We probe into how to make the effects of quantization as small as possible; how many bits are allowed for training and recall; how many layers for the MLFNN are really required to represent physical states, parameters, and variables in order to ensure a certain training and nonlinear ability of a given standard. The main contributions of this paper are the following. We combine the effect of quantization and clipping, and analyze the performance degradation caused by fixed-point and floating-point quantizations, which to the best of our knowledge has not been reported so far. We adopt the uniform and normal probability distributions to analyze a true nonlinear neuron; we use a realistic sigmoid or hyperbolic tangent function as quantization error analysis for a nonlinear neuron. We derive in detail the effect of the quantization error on forward and backward propagation, and our method is less limited and based on more in-depth mathematical analysis. We compare the training performances with and without weight clipping, and derive the effect of the quantization error on forward and backward propagation in detail; analyze the tendency of the weights distribution for the floating-point and fixed-point quantization training; establish and analyze the relationships for a true nonlinear neuron between inputs and outputs bit resolution, number of network layers and performance degradation, based on statistical models of on-chip and off-chip training.

The paper is organized as follows. Section 2 demonstrates the theoretical analysis of quantization and its potential problems in MLFNN. From statistical models, we establish the relationships among inputs and outputs bit resolution, number of layers and performance degradation for both fixed-point and floating-point quantizations. In Sec. 3, experimental results are presented. The simulation results verify and confirm the theoretical analysis. Section 4 provides conclusions. Finally, the mathematical proofs are contained in the appendices.

2. Quantization Analysis of MLFNN
Assuming that $X^{(l)}$ ($l = 1, 2, \ldots, L - 1$) expresses the state vector of nodes in layer $l$ with a total $N_l$ nodes, and $X^{(1)}$ is the input vector, $X^{(L)}$ and $t$ are respectively the actual and target output vectors which are desired to be binary ($-1$ and $1$). The activation function is a hyperbolic tangent. $w^{(l)}_{j,i}$ is the synaptic weight connecting the $i$th node of layer $l-1$ to the $j$th node of layer $l$ in the $L$ layers of the MLFNN. There are $P$ training sample sets. $x^{(l)}_{N_l+1} = 1$ and $w^{(l+1)}_{j,N_l+1} = \theta^{(l+1)}_{j,N_l+1}$ denote a neuron’s bias and threshold, respectively. The number of quantization
bits after and before a neuron in layer \( l \) is further denoted by \( M_{\text{out}}^{(l)} \) and \( M_{\text{in}}^{(l)} \), respectively. \( \Delta \) is the quantization width, \( \Delta x \) and \( \Delta w \) are the quantization noises, and are independent of each other and of all inputs and outputs. The error due to finite precision depends on the number representation used: computation of floating-point is of higher precision than that of fixed-point at the same word length; however the implementation of fixed point is simpler and its implementation cost in hardware is lower. For fixed-point quantization, the rounding error is uniformly distributed in \([-\Delta/2, \Delta/2]\), its mean is zero and its variance is \( \sigma^2_\Delta = \Delta^2/12 \). For floating-point quantization, the numbers are represented as \((-1)^S \times 2^{c} \times M\), where \( 1/2 \leq M < 1 \), \( S \) is a sign bit, \( c \) is the exponent bits of a non-negative integer, \( M \) is a pure binary fraction. As the dynamic range of floating-point quantization is very large, overflow does not easily occur. After floating-point quantization, the weight or the signal \( x \) becomes \( Q[x] = x(1 + \varepsilon) = x + e \), where \(-\Delta < \varepsilon < \Delta \) is a relative error, with zero mean and variance \( \sigma^2_e = \Delta^2/3 \).

### 2.1. The fixed-point quantization analysis of MLFFNN

By combining the inputs to a neuron, the output of node \( i \) of layer \( l \) can be written as:

\[
x^{(l)}_i = f(y^{(l)}_i) = f \left( \sum_j w^{(l)}_{i,j} x^{(l-1)}_j \right) = \tanh \left( \lambda \sum_j w^{(l)}_{i,j} x^{(l-1)}_j / (N_{l-1} + 1) \right)
\]

where \( 2 < l \leq L; j = 1, 2, \ldots, N_l; i = 1, 2, \ldots, (N_{l-1} + 1) \); \( \lambda \) is a constant decided by experiment.

Equation (1) shows that when the neuron’s fan-in is very large, the internal activation \( y^{(l)}_i \) may be very large, which may cause saturation of analog circuits and numerical overflow in digital systems. The normalization by the compression factor \( \lambda / (N_{l-1} + 1) \) makes \( y^{(l)}_i \) unrelated to the fan-in number and can decrease the chance that the network’s output falls into a false saturated state. Moreover, it increases the ability of escaping from a false saturated output state or of breaking away from local minima.\(^{11,14}\) After quantization \( y^{(l)}_i \) is as follows:

\[
Q[y^{(l)}_i] = \sum_{j}^{N_{l-1}+1} (w_{i,j} + \Delta w_{i,j}) (x_j + \Delta x_j) = \sum_{j=1}^{N_{l-1}+1} w^{(l)}_{i,j} x^{(l-1)}_j + \Delta y^{(l)}_i = y^{(l)}_i + \Delta y^{(l)}_i
\]

where \( \Delta y^{(l)}_i \) is the quantization error of \( y^{(l)}_i \). This quantization error is equal to:

\[
\Delta y^{(l)}_i = \sum_{j}^{N_{l-1}+1} (\Delta w_{i,j} x_j + w_{i,j} \Delta x_j)
\]

where \( j = 1, 2, \ldots, N_{l-1}+1; i = 1, 2, \ldots, N_l \) and the second-order items are omitted.

Combining Eqs. (1) and (2), we obtain the variance of \( y^{(l)}_i \):

\[
\sigma_{y^{(l)}_i}^2 = \sum_{j=1}^{N_{l-1}+1} E\{w^2_{i,j} x^2_j\} = (N_{l-1} + 1) \sigma_{x^{(l-1)}}^2 \sigma_{w^{(l)}}^2
\]
and the variance of \( \Delta y_{i}^{(l)} \):

\[
\sigma^{2}_{\Delta y_{i}^{(l)}} = \sum_{j=1}^{N_{l}+1} \{E\{w_{i,j}^{2}\} \sigma^{2}_{\Delta x_{j}^{(l-1)}} + E\{x_{j}^{2}\} \sigma^{2}_{\Delta w_{i,j}}\} = 2(N_{l-1} + 1)(\sigma^{2}_{x_{i}^{(l-1)}} + \sigma^{2}_{w_{i}^{(l)}}) \sigma^{2}_{\Delta w_{i}^{(l)}} . \tag{4}
\]

Combining Eqs. (3) and (4), we obtain the noise-to-signal ratio (NSR) of \( y_{i}^{(l)} \) as:

\[
\text{NSR}_{y_{i}^{(l)}} = \frac{\sigma^{2}_{\Delta y_{i}^{(l)}}}{\sigma^{2}_{y_{i}^{(l)}}} = \frac{(N_{l-1} + 1)(\sigma^{2}_{x_{i}^{(l-1)}} + \sigma^{2}_{w_{i}^{(l)}}) \sigma^{2}_{\Delta w_{i}^{(l)}}}{(N_{l-1} + 1) \sigma^{2}_{x_{i}^{(l-1)}} \sigma^{2}_{w_{i}^{(l)}}} = \frac{2}{2^{2M^{(l)}}} . \tag{5}
\]

In order to make the analyzed results more reliable and to more accurately predict the quantization error and the properties of the network, the true function of a nonlinear neuron (i.e. a realistic hyperbolic tangent function as activation function) is used with two different probability distributions. The quantization error \( \Delta x_{i}^{(l)} \) after the neuron for output node \( i \) of layer \( l \) of the FNN is approximately expressed by a first-order Taylor expansion of the activation function, giving:

\[
\Delta x_{i}^{(l)} \approx \frac{\partial x_{i}^{(l)}}{\partial y_{i}^{(l)}} \Delta y_{i}^{(l)} = \lambda (1 - (x_{i}^{(l)})^{2}) \Delta y_{i}^{(l)}/(N_{l-1} + 1) \leq \lambda \Delta y_{i}^{(l)}/(N_{l-1} + 1) . \tag{6}
\]

Therefore, we obtain the variance of \( \Delta x_{i}^{(l)} \) as:

\[
\sigma^{2}_{\Delta x_{i}^{(l)}} = \left( \frac{\partial x_{i}^{(l)}}{\partial y_{i}^{(l)}} \right)^{2} \sigma^{2}_{\Delta y_{i}^{(l)}} = \lambda^{2}(1 - (x_{i}^{(l)})^{2})^{2} \sigma^{2}_{\Delta y_{i}^{(l)}/(N_{l-1} + 1)^{2}} = \lambda^{2}(1 - (x_{i}^{(l)})^{2})^{2} (\sigma^{2}_{w_{i}^{(l)}} \sigma^{2}_{\Delta x_{i}^{(l-1)}} + \sigma^{2}_{x_{i}^{(l-1)}} \sigma^{2}_{\Delta w_{i}^{(l)}})/(N_{l-1} + 1) . \tag{7}
\]

Equation (7) is different from Xie’s linear analysis and Dundar’s nonlinear analysis, as shown in Fig. 1. This previous research into the sensitivity and quantization error analysis of neural networks used only linear neurons with uniform probability distribution or simple nonlinear neurons; hence the limits of the rigid boundaries apply. In our paper we derive the effect of the quantization error on forward and backward propagation in detail and our method uses less limits and more in-depth mathematics. We know that the closer the neuron output is to the saturation state, the more compressed its quantization error will be. Almost no noise will occur when the activation function is over-saturated. The variance of the output error depends on the derivative of the neuron’s output (i.e. the deepness of saturation) and the variance of \( \Delta y_{i}^{(l)} \).
If the internal activation $y_i^{(l)}$ complies with a uniform distribution, then according to Appendix 1, the NSR of output node $i$ of layer $l$ of the FNN after and before the neuron obeys the following relation:

$$\text{NSR}_{x_i^{(l)}} \max = \beta_{y_i^{(l)}} \text{NSR}_{y_i^{(l)}} > \frac{\sigma_{y_i^{(l)}}^2}{\sigma_x^2} + \frac{\sigma_w^2}{\sigma_{y_i^{(l)}}^2} = \text{NSR}_{y_i^{(l)}}$$

(8)

where

$$\beta_{y_i^{(l)}} = \frac{1}{1 - \frac{6\lambda^2\sigma_{y_i^{(l)}}^2}{5(n_{l-1}+1)^2} + \cdots} \quad \text{(9)}$$

It is a monotonically increasing function of $\frac{\lambda\sigma_{y_i^{(l)}}}{N_{l-1}+1}$ within the limited range, and it is also unrelated to the quantization error. Its maximum and minimum are respectively equal to:

$$\frac{\lambda\sigma_{y_i^{(l)}}}{N_{l-1}+1} \to 0 \quad \text{and} \quad \frac{\lambda\sigma_{y_i^{(l)}}}{N_{l-1}+1} \to \frac{\pi}{2\sqrt{2}}$$

(10)

In the above theoretical analysis, signals, weights and the internal activation of the output nodes are assumed to be uniformly distributed, but in fact this is not always a good assumption. It has been concluded\(^{18}\) that the weight distribution in a neural network resembles a normal distribution when the word width is long enough. This means that there are many weights with small values and only a few weights with large values. If the fan-in number is large enough, then according to the central limit theorem the weighted sum can be looked at as having $N_{l-1}$...
independent random input variables, and the density of their weighted sum can be approximately considered to be a normal distribution. Under this assumption and according to the analysis in Appendix 2, the NSR of output node \( i \) of layer \( l \) of the FNN after and before the neuron obeys the following relationship:

\[
\text{NSR}_{x_l^{(i)}} = \frac{\sigma^2_{\Delta x_l^{(i)}}}{\sigma^2_{x_l^{(i)}}} = \rho_i^{(l)} \frac{\sigma^2_{\Delta y_l^{(i)}}}{\sigma^2_{y_l^{(i)}}} \geq \frac{\sigma^2_{\Delta y_l^{(i)}}}{\sigma^2_{y_l^{(i)}}} = \text{NSR}_{y_l^{(i)}}
\]  

(11)

where

\[
\rho_i^{(l)} = \frac{\lambda^2}{(N_l - 1 + 1) \sigma^2_{y_l^{(i)}}} \int_{-\infty}^{\infty} \frac{4 \exp(-s^2/2)}{(1 + \cosh \left( \frac{2s}{\lambda \sigma_{y_l^{(i)}}} \right))} ds
\]

(12)

It is a monotonically increasing function of \( \frac{\lambda \sigma_{y_l^{(i)}}}{(N_l - 1 + 1)} \) within the limited range, and is also unrelated to the quantization error. Its maximum and minimum are respectively equal to:

\[
\lim_{\frac{\lambda \sigma_{y_l^{(i)}}}{(N_l - 1 + 1)} \to 0} \rho_i^{(l)} = 1 \quad \text{and} \quad \lim_{\frac{\lambda \sigma_{y_l^{(i)}}}{(N_l - 1 + 1)} \to \infty} \rho_i^{(l)} = 4.
\]

(13)

### 2.2. The effect of the quantization error on the forward propagation of MLFNN

For the fixed-point quantization, when \( N_{l-1} \) is small, the probability density function of the random variable \( y_l^{(i)} \) is approximately a uniform distribution. Considering Eq. (8), the maximal NSR of \( x_l^{(i)} \) is characterized by the following recursive formula:

\[
\text{NSR}_{x_l^{(i)}}^{\text{max}} = \frac{\sigma^2_{\Delta x_l^{(i)}}}{\sigma^2_{x_l^{(i)}}} \bigg| \left( \text{NSR}_{x_{l-1}^{(i-1)}} + \frac{\sigma^2_{\Delta w_l^{(i)}}}{\sigma^2_{w_l^{(i)}}} \right) \bigg] = \beta^{(l)} \left( \frac{\sigma^2_{\Delta w_{l+1}^{(i)}}}{\sigma^2_{w_{l+1}^{(i)}}} + \frac{\sigma^2_{\Delta w_{l+2}^{(i)}}}{\sigma^2_{w_{l+2}^{(i)}}} \right) = \beta^{(l)} \left( \frac{\sigma^2_{\Delta w_{l+1}^{(i)}}}{\sigma^2_{w_{l+1}^{(i)}}} \right) = \beta^{(l)} \left( \frac{\sigma^2_{\Delta w_{l+2}^{(i)}}}{\sigma^2_{w_{l+2}^{(i)}}} \right) = \beta^{(l)} \left( \frac{\sigma^2_{\Delta w_{l+3}^{(i)}}}{\sigma^2_{w_{l+3}^{(i)}}} \right) + \cdots
\]

(14)

On the other hand, when \( N_{l-1} \) is large enough, then according to the central limit theorem the probability density function of \( y_l^{(i)} \) approaches a normal distribution. Considering Eq. (11), the NSR of \( x_l^{(i)} \) is then characterized by the following...
giving

According to Eq. (10), if we select

The NSR is enlarged layer after layer. If the internal activation complies with a uniform distribution, then according to Eq. (14), we have

The input errors are propagated feedforward from input layer to output layer. The NSR is enlarged layer after layer. If the internal activation complies with a normal distribution, then according to Eq. (15), we have

The larger the internal activation and the larger the nonlinear compression, the smaller the number of bits \( M^{(l)} \) are needed. Combining Eqs. (17) and (18), we know that \( M^{(l)} \) can decrease 1–1.5 bits per layer depending on the distribution of \( y_j^{(l)} \). If we keep the NSR constant from input layer to output layer, i.e. \( M^{(l)} \) is kept constant, then whenever a layer is added to the network, the precision of
quantization must increase by 1–1.5 bits. The accuracy of the network on which the quantization error of the output has less effect depends mainly on that of the input and hidden layers. As $1 \leq \rho^{(l)}$ and $1 \leq \beta^{(l)}$ are unrelated to the quantization error and enlarged layer by layer, then if we need to reduce the NSR of the output layer, one measure is to reduce the input error; another measure is to decrease the number of layers $L$; the third measure is to depress $\beta^{(l)}$ and $\rho^{(l)}$ (i.e. depress $\frac{\lambda}{N_{l-1}+1}y^{(l)}_i$).

From the above analysis, we know that, no matter what distribution the internal activation of the neuron output complies with, the NSR after the nonlinear neuron is always enlarged. As the NSR increases with the number of layers, more bits are needed to keep the NSR constant. The necessary number of bits can be reduced from layer to layer if the neurons have an effective nonlinear compression; increasing the bits results in an improvement of the NSR which is independent of the fan-in number. If we change the fan-in number, and keep the number of layers constant, then the output NSR remains the same.

It was shown$^3$ that a MLFNN with a single hidden layer with nonlinear sigmoidal neurons and one output layer with linear neurons can uniformly approximate any continuous function to an arbitrary degree of exactness provided that the hidden layer contains a sufficient number of nodes. Concerning the numbers of neurons and weights in the fully connected MLFNN, the weight number is proportional to the square of the number of neurons. For a MLFNN with $L$ layers, its number of neurons and weights are $\sum_{l=2}^{L} N_l$ and $\sum_{l=2}^{L} (N_{l-1} + 1) N_l$ respectively. Because the NSR increases with the layer number at a fixed number of quantization bits, and the ratio is independent of the number of input nodes, increasing the number of layers can improve the nonlinear ability of the network to some extent. The nonlinear ability of a MLFNN with a single hidden layer is enhanced by increasing the number of hidden nodes.$^9$ Therefore, if the balance must be made between the NSR and the number of neurons, weights and layers, we think that a network with a single hidden layer is a better choice in the reality of finite precision.

2.3. The floating-point quantization analysis

We assume that after floating-point quantization the internal activation $y^{(l)}_i$ of output node $i$ of layer $l$ of the FNN is $Q[y^{(l)}_i] = y^{(l)}_i (1 + \varepsilon) = y^{(l)}_i + e^{(l)}_i$. The quantization errors are classified into additive errors and multiplicative errors, and their variance can be written as $\sigma^2_e = \sigma^2_{e(+)} + \sigma^2_{e(\star)}$.

The variance of multiplicative error is equal to:

$$\sigma^2_{e^{(\star)}(+)} = \sum_{j=1}^{N_{l-1}+1} \sigma^2_e \sigma^2_{x} \sigma^2_{w_i} = (N_{l-1} + 1)\sigma^2_x \sigma^2_{w_i} \sigma^2_{w_i}. \quad (19)$$

The variance of additive error is equal to:

$$\sigma^2_{e^{(+)}} = \sum_{j=2}^{N_{l-1}+1} j \sigma^2_e \sigma^2_{x} \sigma^2_{w_i} = \frac{N_{l-1}(N_{l-1}+3)}{2} \sigma^2_x \sigma^2_{w_i} \sigma^2_{w_i}. \quad (20)$$
Combining Eqs. (19), (20) and (3), we obtain the NSR of $y^{(l)}_i$ as:

$$\text{NSR}^{\text{float}}_{y^{(l)}_i} = \frac{\sigma^2_{\epsilon^{(l)}_i}}{\sigma^2_{y^{(l)}_i}} = \frac{N_{l-1}^2 + 5N_{l-1} + 2}{2(N_{l-1} + 1)} \sigma^2_{\epsilon^2} = \frac{N_{l-1}^2 + 5N_{l-1} + 2}{2(N_{l-1} + 1)} \Delta^2 \beta^2.$$  \hspace{1cm} (21)

From Eq. (21), we see that the NSR before the neuron for the case of floating-point quantization is unrelated to the input signals of every layer but it is a monotonically increasing function of the number of input nodes of this layer. The more the number of input nodes, the more the number of additive and multiplicative items, and the larger the additive error and the multiplicative error. The NSR after the neuron is the same as that in the case of fixed-point quantization, but enlarged by a factor $\beta^2_i$ or $\rho^2_i$. The NSR of the output layer is also unrelated to the number of layers. Equation (21) holds only when the dynamic range or the number of exponent bits of the floating-point representation is large enough, and therefore no clipping in the computations occurs and only the error of the limited mantissa can be considered. However, in practice the number of exponent bits of the floating-point representation is also limited, and therefore in practice clipping and overflow always exist. If $c$ increases with 1 bit, then the dynamic range of the floating-point representation will double and will be wider than that of the fixed-point representation.

### 2.4. The effect of the quantization error on backpropagation

Feedforward networks use mostly the backpropagation (BP) algorithm of gradient descent\textsuperscript{15} for training. The goal of the training is to find the set of weights which corresponds to a global minimum of the error energy function, or at least the weights which give an error lower than some tolerable limit. There are mainly three kinds of training manners in the presence of finite precision:

1. On-chip training means quantization and clipping of the weights at each iteration.
2. Chip-in-the-loop learning: The neural network hardware is only used in the forward propagation of the training; the calculation of the new weights is done off-chip on a computer, which downloads the updated weights onto the chip after each training iteration.
3. Off-chip training means that its training process is performed on a high-precision computer; then the weights are quantified and downloaded on the chip, the recall is performed for the forward propagation.

The approach of off-chip training creates potentially dangerous problems, i.e. the probability of getting stuck into local minima increases: due to training off-chip and then weight clipping for recall, new local minima may appear on the border of the allowed region in the weight space. But sometimes the network can leap over the local minimum by the process of quantization. This can help convergence in some cases, however the procedure will not guarantee convergence.$^5$ A feasible approach
to finding good weights is on-chip training which repeats the clipping of the weights at each iteration of the gradient method. Clipping during training results in neural networks with significantly better performances for recall, as this kind of training is shown to be more robust in performance for the data of training and test.

Because most computations are required in the training phase, the effects of quantization are more significant in this phase. In order to adapt the learning of new patterns very quickly, all computations must be attempted to be implemented on-chip. The on-chip updates are constrained by a finite precision range which departs to some extent from the true gradient descent. This is not just an approximation of the gradient descent, but most of the time has a significantly different behavior, particularly when the learning rate is not small enough with respect to the number of training patterns. The experimental results showed that slower learning appeared when more weights reached the limits of the clipping function. The performance with clipping during training is lower than that without clipping. As the effect of the gradient descent on training in computations with finite precision will be difficult to measure, the statistical evaluation of the weight updates do not effectively determine a propensity of the network to learn.

If we adopt on-chip training, then the weights of the output layer are updated according to the gradient descent algorithm as:

\[ Q[w_{j,i}^{(L)}(k + 1) - w_{j,i}^{(L)}(k)] = Q[\eta^{(L)}Q[\delta_j^{(L)}]x_i^{(L-1)}] \] (22)

where

\[ Q[\delta_j^{(L)}] = Q[\lambda Q[(t_j - x_j^{(L)})/(1 - (x_j^{(L)})^2)]/N_{L-1} + 1] \] (23)

and \( j = 1, 2, \ldots, N_L \), \( i = 1, 2, \ldots, N_{l-1} + 1 \). \( \eta \) is the learning rate and \( k \) is the iteration number. The weights of the remaining layers are updated as:

\[ Q[w_{j,i}^{(l)}(k + 1) - w_{j,i}^{(l)}(k)] = Q[\eta^{(l)}Q[\delta_j^{(l)}]x_i^{(l-1)}] \] (24)

where

\[ Q[\delta_j^{(l)}] = Q \left[ \lambda Q[(1 - (x_j^{(l)})^2)] : Q \left[ \sum_{r=1}^{N_{l-1}} w_{r,j}^{(l+1)} Q[\delta_r^{(l+1)}] \right] / (N_{l-1} + 1) \right] \] (25)

and \( 2 \leq l \leq L - 1, j = 1, 2, \ldots, N_l, r = 1, 2, \ldots, N_{l+1} \), \( i = 1, 2, \ldots, N_{l-1} + 1 \).

Assuming that \( x_i^{(l-1)} |_{\text{max}} = 1 \), for the output layer, combining Eqs. (22) with (23), we have that, if and only if

\[ Q \left[ \eta^{(L)} \lambda |t_j - x_j^{(L)}|(1 - (x_j^{(L)})^2) \right] / N_{L-1} + 1 > \Delta / 2 \] (26)

holds at each iteration, the weight updates can be performed. The random value \( |t_j - x_j^{(L)}|(1 - (x_j^{(L)})^2) \) in Eq. (26) can be obtained by using Anand’s method in
our quantization analysis. The conditional expectation in the first few iterations of backpropagation after some simple derivations is:

\[
E_{w^{(L)}}[d_j - x_r^{(L)}(1 - (x_r^{(L)})^2)] = \left[1 - \sum_{r=1}^{N_{L-1}} \frac{\Delta^2 2^{2^M}}{12} \left(\frac{\lambda x_r^{(L-1)}}{(N_{L-1} + 1)}\right)^2\right] < 1. \tag{27}
\]

Combining Eqs. (26) with (27), at least when \(Q\left[\frac{\eta^{(L)}}{N_{L-1} + 1}\right] > \Delta/2\) is satisfied, the iteration can be performed.

For the \(L - 1\) layer, combining Eqs. (24) with (25), we have that, if and only if

\[
Q \left[\frac{\eta^{(L-1)} \lambda^2 (1 - (x_r^{(L-1)})^2) x_i^{(L-2)}}{(N_{L-1} + 1)(N_{L-2} + 1)} \sum_{r=1}^{N_{L+1}} w_{r,j}^{(L)}(t_r - x_r^{(L)})(1 - (x_r^{(L)})^2)\right] > \Delta/2 \tag{28}
\]

holds at each iteration, the weight updates can be performed. For the same reason, the conditional expectation of the random variable \(\sum_{r=1}^{N_{L+1}} w_{r,j}^{(L)}(t_r - x_r^{(L)})(1 - (x_r^{(L)})^2)\) in the first few iterations of backpropagation satisfies:

\[
E_{w^{(L)}} \left\{\sum_{r=1}^{N_{L+1}} w_{r,j}^{(L)}(t_r - x_r^{(L)})(1 - (x_r^{(L)})^2)\right\} = \frac{N_{L} \lambda x_j^{(L-1)}}{(N_{L-1} + 1)} \frac{\Delta^2 2^{2^M}}{12}. \tag{29}
\]

Substituting Eqs. (29) into (28), we have

\[
Q \left[\frac{\eta^{(L-1)} \lambda^3 N_{L}(1 - (x_j^{(L-1)})^2) x_i^{(L-1)} x_r^{(L-2)}}{(N_{L-1} + 1)^2(N_{L-1} + 1)} \frac{\Delta^2 2^{2^M}}{12}\right] > \Delta/2. \tag{30}
\]

Due to the normalized internal activation in forward propagation, a compress factor \(\lambda/(N_{l-1} + 1)\) will be generated by the derivative of activation in a gradient descent during the BP process. By comparing Eqs. (26) with (30), we find that the decrease of the weight updates by the compress factor will have an influence for the output and the other layers. From Eqs. (23) and (25), we know that the delta’s decrease of the output layer is caused by the derivative of the internal activation in the layer, and the delta in the remaining layers is related to the derivative of the activation in this layer and the following layers. Therefore, the range of the weight updates is cumulatively decreased by the factor from the output to the input layer. In order to cancel these kinds of effect and improve the training performance, \(\eta^{(L)}\) is selected to be inversely proportional to the factor. If \(\eta^{(L)}\) is selected too large, the weight updates are very large, then the weights are quantized and clipped at each iteration. When the weights reach the border of \([-W_{\text{max}}, W_{\text{max}}]\) during the training, and the gradient is directed out of \([-W_{\text{max}}, W_{\text{max}}]\), then the weights are updated along the projection of the gradient onto the borderline of \([-W_{\text{max}}, W_{\text{max}}]\), instead of along the gradient itself. In this case, as there exists a quantization error, the weights are unlikely to be updated towards the direction of gradient optimization. \(^{19}\)

If \(\eta^{(L)}\) on the other hand is selected too small, then the rate of the weight updates is too slow (perhaps because the weight updates are less than the quantization width), and the network training paralyses. When the training fails to converge,
then the weight updates are smaller than the quantization width, which prevents the weights from changing; further, there may be weight clipping or insufficient weight resolution.

During the training process with finite precision, the BP algorithm is highly sensitive to weights and training fails when the weight accuracy is worsened. The weight updates are very small for a large size of network. When they are smaller than the quantization width, this prevents the weights from changing and thus the network paralyses. It is the main reason to restrict the implementation of large-scale neural networks in the presence of limited accuracy.

3. Experiments

We carried on several simulation experiments to evaluate and confirm our above theoretical results. Two different performance measurements have been used: the first one is the minimal RMS error for an approximation problem, and the second one is the misclassification of patterns for a classification problem.

3.1. Noisy XOR problem

Noisy XOR data used 2000 training patterns generated from four normal distributions with mean vectors and covariance matrices, as shown in Table 1 and Fig. 2, corresponding to two classes and four clusters which consist of 500 random sample sets. 250 sample sets were trained by a two-layer FNN (one single hidden layer, the structure of the network is (2+1)-(15+1)-1, here +1 denotes the bias node) and by a three-layer FNN (two hidden layers, the structure of the network is (2+1)-(8+1)-(6+1)-1)) into two classes, the remaining 250 sample sets were used to classify.

Weight matrices were initialized with random values, uniformly distributed in the range [-1.0;+1.0]. The learning rate $\eta^{(t)}$ was set within $[0.01N_{l-1},0.25N_{l-1}]$ for the two-layer FNN and $[0.008N_{l-1},0.15N_{l-1}]$ for the three-layer FNN, which was proportional to the fan-in number of the layer and the highest learning rate that permitted convergence (i.e. did not lead to oscillations), $\lambda = 1$. The training parameters were made as large as possible so that the weight updates would get larger. Training was stopped when the misclassification for the training set was unchanged or after 10,000 epochs.

<table>
<thead>
<tr>
<th>Class 1</th>
<th>Class 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Vector</td>
<td>Mean Vector</td>
</tr>
<tr>
<td>Covariance Matrix</td>
<td>Covariance Matrix</td>
</tr>
<tr>
<td>-0.9891</td>
<td>-0.9191</td>
</tr>
<tr>
<td>0.4048</td>
<td>0.4894</td>
</tr>
<tr>
<td>-0.0145</td>
<td>-0.0173</td>
</tr>
<tr>
<td>0.5684</td>
<td>0.6554</td>
</tr>
<tr>
<td>0.9249</td>
<td>0.6228</td>
</tr>
<tr>
<td>-0.0173</td>
<td>0.0493</td>
</tr>
<tr>
<td>0.0277</td>
<td>0.0493</td>
</tr>
<tr>
<td>0.5684</td>
<td>0.3963</td>
</tr>
</tbody>
</table>
Quantization on Multi-Layer Feedforward Neural Networks

Fig. 2. Noisy XOR problem, corresponding to two classes and four clusters.

Fig. 3. (a) Initial weights with uniform distribution in the range \([-1.0, +1.0]\); (b) weight distribution after training with infinite precision; (c) weight distribution with 15/1 bits floating-point quantization by on-chip training; (d) weight distribution with 16 bits fixed-point quantization by on-chip training.
Figure 3(a) shows that the initial weights were uniformly distributed in the range \([-1.0, +1.0]\). Figure 3(b) shows that even if the initial weights comply with a uniform distribution and the maximal fan-in number was equal to 15 (i.e. is not too large) for the two-layer FNN, the trained weights which are mainly distributed within the range \([-10, +10]\) can be seen to have approximately a normal distribution using infinite precision. Figure 3(c) shows the weight distribution of the floating-point quantization in which the mantissa was 15 bits and the exponent was 1 bit after on-chip training. Figure 3(d) shows the weight distribution with 16 bits fixed-point quantization in which, after on-chip training, it is observed that many weights are distributed near +1 or −1 due to clipping, showing that the dynamic range of the floating-point quantization is wider than that of the fixed-point quantization. The weight distribution of the floating-point quantization approaches approximately a normal distribution.

Figure 4 shows the misclassification of the test set as a function of the number of bits for the floating-point quantization (the exponent is equal to 1 bit). It is observed that the misclassification percentage of the two-layer FNN for on-chip training or chip-in-the-loop training decreases dramatically when going from 5 to 11 bits, and reaches around 17–19% at 11 bits. The misclassification percentage of the three-layer FNN for on-chip training or chip-in-the-loop training decreases dramatically when going from 7 to 13 bits, and reaches around 14–15% at 13 bits. Above 15 bits, the misclassification of all training manners decreases slowly, and reaches around 14–18% at 23 bits. For off-chip training, the range of change of

![Figure 4](image-url)
the misclassification percentage is less than that of on-chip training. Below 8 bits the two-layer FNN has a lower misclassification than the three-layer FNN. Above 13 bits the two kinds of on-chip training have a lower misclassification than off-chip training.

Figure 5 is the misclassification with different number of bits of the fixed-point quantization. Comparing Fig. 4 with Fig. 5, with fewer bits of quantization the two-layer FNN has a better performance than the three-layer FNN, because of the higher NSR of the three-layer FNN. When the number of quantization bits is increased, the misclassification of the three-layer FNN improves, and above 16 bits the misclassification of the three-layer FNN is a little better than that of the two-layer FNN. Because the exponent $c (=1)$ of the floating-point quantization is very low, overflow easily occurs. In this case, while the floating-point quantization has similar features as the fixed-point quantization, the dynamic range of the former is wider than that of the latter, and the former has a better performance than the latter, especially for lower numbers of quantization bits.

3.2. Function approximation

To verify the theoretical evaluation of MLFNN with finite-precision computations as presented in this paper, we used a function-mapping problem. The input to the network was a random number in the interval $[-1, +1]$ and the target was the mapping of the $0.7 \tanh(2x)$ curve. Both two-layer and three-layer network
structures (1+1)-(10+1)-1 and (1+1)-(6+1)-(5+1)-1 were used to simulate the approximation of the function. The training data were constructed from 25 pairs of uniformly distributed data within the interval range \([-1, +1]\), and the test data were 25 pairs of randomly selected data which differed from the training data. The weight matrices were initialized with random values uniformly distributed in the range \([-1.0, +1.0]\). The training stopped when the minimal RMS error for each network output was reduced to \(\text{RMS}_{\text{min}} = 0.0027\) or after 3000 epochs. The learning rate \(\eta^{(t)}\) was set within \([0.1N_{l-1}, 0.3N_{l-1}]\) for the two-layer FNN and \([0.05N_{l-1}, 0.1N_{l-1}]\) for the three-layer FNN, \(\lambda = 1\). Figure 6(a) shows that the initial weights were uniformly distributed in the range \([-1.0, +1.0]\). Figure 6(b) shows that the weight distribution, trained with infinite precision, can be considered approximately as a normal distribution. Figure 6(c) shows the weight distribution of the floating-point quantization in which the mantissa was 11 bits and the exponent was 1 bit after on-chip training. Figure 6(d) shows the weight distribution with 12 bits fixed-point quantization after on-chip training. Combining Figs. 6(c) and 6(d), it is observed that the weights near 0 increase due to clipping and quantization.
Figure 7 shows a comparison of the minimal RMS error for the function approximation problem when using different bits of fixed-point quantization, indicating that on-chip training has similar performance as chip-in-the-loop training. The two-layer FNN on-chip training has a lower minimal RMS error than the three-layer FNN below 12 bits. But when the number of quantization bits is above 12, the three-layer FNN has a lower minimal RMS error than the two-layer FNN, because the three-layer FNN has a higher nonlinear ability. The minimal RMS error can effectively be decreased using a higher number of quantization bits. For off-chip training the minimal RMS error is not sensitive to changing the number of quantization bits. Above 16 bits the minimal RMS error for on-chip training is lower than that of off-chip training. Figure 8 shows the comparison of the minimal RMS error for the function approximation problem when using a different number of floating-point quantization bits, indicating that on-chip training has similar performance as chip-in-the-loop training. The two-layer FNN on-chip training has a lower minimal RMS error than the three-layer FNN below 16 bits. But when the number of quantization bits is above 24, then the three-layer FNN has a lower minimal RMS error than the two-layer FNN. For off-chip training, the minimal RMS error is not too sensitive to changing the number of quantization bits. Comparing Figs. 4, 5, 7 and 8, it can be concluded that the performance of off-chip training at low number of bits is better than that of on-chip training. Because quantization and clipping are performed at each iteration of on-chip training, the performance of training is
poor at lower numbers of bits. But at high numbers of bits on-chip training enables weights better suited to training and testing, thus improving the performance of the network.

3.3. Classification for Fisher’s Iris data

We now show the performance comparison of different numbers of quantization bits based on Fisher’s Iris data. The classifier used the two-layer FNN (the structure of the network is (4+1)-(17+1)-3)) and the three-layer FNN (the structure of the network is (4+1)-(7+1)-(6+1)-3)), respectively. Fisher’s Iris data set contains 150 patterns belonging to three classes, for each of which there are 50 exemplars and each input is a four-dimensional real vector. The original patterns were translated and scaled such that each component of the input vector lies within the range $[1^{-1.0}; +1.0]$. From each class 30 samples were used for training, and all 50 samples were used for testing. Training was stopped when the number of misclassifications was unchanged or after 2000 epochs. The weight matrices were initialized with random values uniformly distributed in the range $[-1.0, +1.0]$. The learning rate $\eta^{(t)}$ was set within $[0.05N_{l-1}, 0.2N_{l-1}]$ for the two-layer FNN and $[0.01N_{l-1}, 0.08N_{l-1}]$ for the three-layer FNN, $\lambda = 1$.

Figure 9(a) shows that the initial weights are uniformly distributed in the range $[-1.0, +1.0]$. Figure 9(b) shows that, even when the initial weights comply with a uniform distribution and the maximal fan-in number is equal to $17 + 1$ for the
two-layer FNN, the trained weights, which are mainly distributed in the range $[-2.5, +2.5]$, can be approximately looked as a normal distribution when using infinite precision. Figure 9(c) shows the weight distribution after on-chip training for the case of floating-point quantization in which the mantissa was 19 bits and the exponent was 1 bit. Figure 9(d) shows the weight distribution after on-chip training in the case of 20 bits fixed-point quantization, in which it is observed that most weights are distributed near +1 or −1 due to clipping. It is shown that the dynamic range of the floating-point quantization is wider than that of the fixed-point quantization. When the mantissa and exponent are large, the weight distribution of the floating-point quantization approaches the normal distribution of infinite precision.

Figure 10 shows the misclassification for different numbers of quantization bits for the MLFNN on-chip training. It was observed that the misclassification percentage of the fixed-point quantization decreases dramatically from 10 to 12 bits, and reaches around 10% at 12 bits; from 12 to 32 bits, the misclassification decreases slowly, and reaches around 8% at 32 bits. Below 28 bits the two-layer FNN has a lower misclassification percentage than the three-layer FNN, because in this case
the NSR of the three-layer FNN is higher. Above 24 bits both networks have almost the same misclassification error. The floating-point quantization has a better performance than the fixed-point quantization. The performance of the floating-point quantization (exponent is equal to 1 bit) on average is 5% improved compared to that of the fixed-point quantization with the same number of bits. Some results of the floating-point quantization for larger exponents (mantissa bits/exponent bits) are shown in Table 2. It is observed that an increasing number of exponent bits can enlarge the dynamic range of computation, resulting in a lower number of mantissa bits and a better performance.

Comparing Figs. 3, 6 and 9, we conclude that no matter what distribution the initial weights comply with, the weights distribution will approximate a normal distribution for the training with floating-point or high-precision fixed-point quantization. Only when the number of quantization bits is very low, the weights distribution may cluster to ±1 in the case of training with fixed-point quantization.

<table>
<thead>
<tr>
<th>Architecture/ % Misclassification</th>
<th>4/6</th>
<th>6/5</th>
<th>8/4</th>
<th>Infinite Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 – (17 + 1) – 3</td>
<td>3.33</td>
<td>4.0</td>
<td>3.33</td>
<td>2.0</td>
</tr>
<tr>
<td>4 – (7 + 1) – (6 + 1) – 3</td>
<td>8.0</td>
<td>6.0</td>
<td>4.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>
4. Conclusions

The noise-to-signal ratio (NSR) increases with the number of layers of the multi-layer feedforward neural network (MLFNN), indicating that when a layer is added, more bits are needed to keep the performance constant. No matter what distribution the internal activation of the outputs comply with, after the signal passes through the nonlinear neuron, the NSR is enlarged. Although the nonlinear neuron plays a compression role for signal and noise, the signal is compressed more than the quantization noise. The necessary number of bits can be reduced from layer to layer if the neurons have an effective nonlinear compression. Increasing the number of bits results in an improvement of the NSR, and the ratio is independent of the fan-in number. If we need to reduce the NSR after a neuron, one measure is to reduce the input error; another measure is to decrease the number of layers $L$; the third measure is to use less $\sigma_x^2$ and $\sigma_w^2$, or less $\lambda/(N_{l-1} + 1)$, which can effectively reduce the output NSR. The dynamic range of the floating-point quantization is wider than that of the fixed-point quantization, and it has a better performance especially for lower numbers of quantization bits. For a lower number of quantization bits, the two-layer FNN has less misclassifications than the three-layer FNN, because of the higher NSR of the three-layer FNN. No matter what distribution the initial weights comply with, the weights distribution will approximate a normal distribution for the training with floating-point or high-precision fixed-point quantization. Only when the number of quantization bits is very low, the weights distribution may cluster to $\pm 1$ in the case of training with fixed-point quantization. The MLFNN is more sensitive to changing the number of layers for lower numbers of quantization bits. When the number of quantization bits is high, the performance of the three-layer FNN is almost always better than that of the two-layer FNN. The performance of off-chip training at low numbers of bits is better than that of on-chip training, because quantization and clipping are performed at each iteration by on-chip training. The performance of on-chip training is poor at lower numbers of bits. However, at higher numbers of bits on-chip training enables weights better suited for training and testing, thus improving the performance and robustness of network.

Appendix 1: Proof of Eqs. (8) and (9)

Assuming that all of the quantified variables are uniformly distributed, we can easily obtain:

\[ E\{x_i^2\} = E\{w_{i,j}^2\} = \sigma_x^2 = \sigma_w^2 = \int_{-\Delta 2^{M-1}}^{\Delta 2^{M-1}} \frac{x^2}{2 \Delta 2^{M-1}} dx = \frac{\Delta^2 \Delta 2^M}{12} \]

where $i = 1, 2, \ldots, N_l; j = 1, 2, \ldots, N_{l-1} + 1$.

The parameters can be estimated after the nonlinear neuron, if $x_i^{(l)} = f(y_i^{(l)}) = \tanh(\lambda y_i^{(l)}/(N_{l-1} + 1))$. Defining $\max |y_i^{(l)}| = A = \sqrt{3 \sigma_y^2}$, which probability $\left(p_{y_i^{(l)}}\right)$ of $|y_i^{(l)}| > A$ is very low and which is approximately a uniform distribution over the
interval \([-A, A]\), we have the mean of \(x_i^{(l)}\):

\[
E\{x_i^{(l)}\} = \frac{1}{2A} \int_{-A}^{A} \tanh(\lambda y_i^{(l)}/(N_{i-1}+1))dy_i^{(l)} = 0. \tag{A.2}
\]

The variance of \(x_i^{(l)}\) is:

\[
\sigma^2_{x_i^{(l)}} = \frac{1}{2A} \int_{-A}^{A} \tanh^2(\lambda y_i^{(l)}/(N_{i-1}+1))dy_i^{(l)}
\]

\[
= 1 - \frac{(N_{i-1}+1)}{\lambda A} \tanh(\lambda A/(N_{i-1}+1)). \tag{A.3}
\]

Combining with Eqs. (A.2), (A.3), (3), (5) and (7), and when the fan-in number is small and hence the output of the neuron is approximately a uniform distribution, we deduce the NSR of \(x_i^{(l)}\) as:

\[
\text{NSR}_{x_i^{(l)}} = \frac{\sigma^2_{\Delta x_i^{(l)}}}{\sigma^2_{x_i^{(l)}}} = \frac{\lambda^2(1 - (x_i^{(l)})^2)}{1 - \frac{N_{i-1}+1}{\lambda A} \tanh(\lambda A/(N_{i-1}+1))} \frac{\sigma^2_{\Delta y_i^{(l)}}}{\sigma^2_{y_i^{(l)}}} \frac{\sigma^2_{\Delta x_i^{(l-1)}}}{\sigma^2_{x_i^{(l-1)}}}
\]

\[
< \frac{\lambda^2 \sigma^2_{y_i^{(l)}}}{(N_{i-1}+1)^2(1 - \frac{N_{i-1}+1}{\lambda A} \tanh(\lambda A/(N_{i-1}+1)))} \frac{\sigma^2_{\Delta y_i^{(l)}}}{\sigma^2_{y_i^{(l)}}} \frac{\sigma^2_{\Delta x_i^{(l-1)}}}{\sigma^2_{x_i^{(l-1)}}}
\]

\[
= \text{NSR}_{\text{max}}^{x_i^{(l)}} = \beta^{(l)} \frac{\sigma^2_{\Delta y_i^{(l)}}}{\sigma^2_{y_i^{(l)}}} > \frac{\sigma^2_{\Delta y_i^{(l)}}}{\sigma^2_{y_i^{(l)}}} = \text{NSR}_{y_i^{(l)}} \tag{A.4}
\]

where

\[
\beta^{(l)} = \frac{\lambda^2 \sigma^2_{y_i^{(l)}}}{(N_{i-1}+1)^2(1 - \frac{N_{i-1}+1}{\lambda A} \tanh(\lambda A/(N_{i-1}+1)))}. \tag{A.5}
\]

By using a Taylor series expansion of the activation function

\[
\tanh(x) = x - x^3/3 + 2x^5/15 - 17x^7/315 + 62x^9/2835 - \ldots + (-)^{n+1}2^{2n}(2^{2n} - 1)B_n x^{2n-1}/(2n)! + \ldots
\]

where the coefficient \(B_n\) is a \(n\)th-order Bernoulli numeral, and \(|x| < \pi/2\), and assuming that \(\sqrt{3}\lambda \sigma_{y_i^{(l)}}/(N_{i-1}+1) < \pi/2\) (it is very easily satisfied in reality), then Eq. (A.5) can be expressed as:
\[
\beta^{(l)} = \frac{\lambda^2 \sigma^2 y^{(l)}}{(N_{l-1} + 1)^2 \left[ \frac{\lambda^2 \sigma^2 y^{(l)}}{3(N_{l-1} + 1)^2} - \frac{18 \lambda^4 \sigma^4 y^{(l)}}{15(N_{l-1} + 1)^2} + \ldots \right]}
\]

\[
= \frac{1}{6 \lambda^2 \sigma^2 y^{(l)}} > 1. \quad \text{(A.6)}
\]

Equations (A.4) and (A.6) yield the desired result.

**Appendix 2: Proof of Eqs. (11) and (12)**

If the fan-in number is very large, then assuming that \( u = y^{(l)} / \sigma_{y^{(l)}} \) is approximately a normal distribution, according to Eq. (1), we have that \( E[u] = 0 \) and \( \sigma^2_u = 1 \) hold.

The variance of \( x^{(l)}_{y^i} \) can be expressed as:

\[
\sigma^2_{x^{(l)}_{y^i}} = \int_{-\infty}^{\infty} \left( \frac{\lambda y^{(l)}_{y^i}}{(N_{l-1} + 1)} \right)^2 \frac{1}{\sqrt{2\pi} \sigma_{y^{(l)}_{y^i}}} \exp \left( -\frac{(y^{(l)}_{y^i})^2}{2 \sigma^2_{y^{(l)}_{y^i}}} \right) \, dy^{(l)}_{y^i}
\]

\[
= \int_{-\infty}^{\infty} \tanh^2 \left( \frac{\lambda y^{(l)}_{y^i} u}{(N_{l-1} + 1)} \right) \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{u^2}{2} \right) / du. \quad \text{(A.7)}
\]

Assuming that the variance of \( \delta y^{(l)}_{y^i} \) is \( \sigma^2 / \Delta y^{(l)}_{y^i} \), let \( \nu = \frac{\Delta y^{(l)}_{y^i}}{\sigma_{y^{(l)}_{y^i}}} = \frac{1}{\sigma_{y^{(l)}_{y^i}} \sqrt{\text{NSR}_{y^{(l)}_{y^i}}}} \), where \( \text{NSR}_{y^{(l)}_{y^i}} = \frac{\sigma^2_{y^{(l)}_{y^i}}}{\sigma^2_{y^{(l)}_{y^i}}} \) is the NSR of \( y^{(l)}_{y^i} \). Thus we have that \( E[\nu] = 0 \) and \( \sigma^2_\nu = 1 \) hold. The quantization error of \( x^{(l)}_{y^i} \) is equal to:

\[
\Delta x^{(l)}_{y^i} = \tanh \left[ \frac{\lambda}{N_{l-1} + 1} (y^{(l)}_{y^i} + \Delta y^{(l)}_{y^i}) \right] - \tanh \left[ \frac{\lambda}{N_{l-1} + 1} y^{(l)}_{y^i} \right]
\]

\[
= \tanh \left[ \frac{\lambda}{N_{l-1} + 1} \sigma_{y^{(l)}_{y^i}} \left( u + \sqrt{\text{NSR}_{y^{(l)}_{y^i}}} \nu \right) \right] - \tanh \left[ \frac{\lambda}{N_{l-1} + 1} \sigma_{y^{(l)}_{y^i}} u \right].
\]

The variance of \( \Delta x^{(l)}_{y^i} \) can be given as:

\[
\sigma^2_{\Delta x^{(l)}_{y^i}} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left( \tanh \left[ \frac{\lambda}{N_{l-1} + 1} \sigma_{y^{(l)}_{y^i}} \left( u + \sqrt{\text{NSR}_{y^{(l)}_{y^i}}} \nu \right) \right] \right.
\]

\[
- \left. \tanh \left[ \frac{\lambda}{N_{l-1} + 1} \sigma_{y^{(l)}_{y^i}} u \right] \right)^2 \cdot \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{u^2 + \nu^2}{2} \right) \, du \, d\nu.
\]

Assuming that \( \frac{\lambda}{N_{l-1} + 1} \sigma_{y^{(l)}_{y^i}} \sqrt{\text{NSR}_{y^{(l)}_{y^i}}} \ll 1 \) holds, then the variance of \( \Delta x^{(l)}_{y^i} \) can be obtained:

\[
\sigma^2_{\Delta x^{(l)}_{y^i}} = \frac{\lambda^2}{(N_{l-1} + 1)^2 \sigma^2_{y^{(l)}_{y^i}} \text{NSR}_{y^{(l)}_{y^i}}} \int_{-\infty}^{\infty} \frac{4 \exp(-s^2/2)}{\sqrt{2\pi} \left( 1 + \cosh \left( \frac{2\lambda}{N_{l-1} + 1} \sigma_{y^{(l)}_{y^i}} s \right) \right)^2} \, ds. \quad \text{(A.8)}
\]
Combined Eqs. (A.7) with (A.8), the NSR of \( x^{(l)}_i \) obeys the following formula:

\[
\text{NSR}_{x^{(l)}_i} = \frac{\sigma^2_{x^{(l)}_i}}{\sigma^2_{x^{(l)}_i}} = \frac{\lambda^2}{(N_{l-1}+1)^2\sigma^2_{y^{(l)}_i}} \int_{-\infty}^{\infty} \frac{4\exp(-s^2/2)}{\sqrt{2\pi}(1+\cosh \left( \frac{2\lambda}{N_{l-1}+1}\sigma_{y^{(l)}_i} s \right))} ds \text{NSR}_{y^{(l)}_i}
\]

\[
= \rho^{(l)}_i \text{NSR}_{y^{(l)}_i}.
\]  

Equation (A.9) yields the desired result.

References

Quantization on Multi-Layer Feedforward Neural Networks


Minghu Jiang received the Ph.D. degree in signal and information processing from Northern Jiaotong University in 1998. From May 1998 till April 2000, he worked as Postdoctor at the State Key Lab of Intelligent Technology & Systems, at Tsinghua University, Beijing, China. Then he became an associate professor and faculty member of Computer Sciences and Computational Linguistics at Tsinghua University, Beijing, China. From December 2000 till December 2001, he was a postdoctoral researcher in the Department of Electrical Engineering of the Katholieke Universiteit Leuven, Belgium. Currently, he is a professor and Director of the Lab of Computational Linguistics, Department of Chinese Language, Tsinghua University, Beijing. He is the author or co-author of more than 50 published papers.

His research interests include neural networks for signal processing, speech recognition and understanding, natural language processing.

Georges G. E. Gielen received the Ph.D. degree in electrical engineering from the Katholieke Universiteit Leuven, Belgium, in 1990. Currently he is a Professor at the Department of Electrical Engineering at this university. He is the author or co-author of more than 200 published papers in journals and conference proceedings.

Dr. Gielen serves regularly on the program committees of international conferences and on the editorial board of international journals. He is a Fellow of IEEE, a member of the Board of Governors as well as a Distinguished Lecturer of the IEEE Circuits and Systems Society. He was the 1997 Laureate of the Belgian Royal Academy of Sciences, Literature and Arts in the category of engineering sciences, and he received the 1995 Best Paper Award of the John Wiley *International Journal on Circuit Theory*.

His research interests include the design and particularly the computer-aided design of analog and mixed-signal integrated circuits, signal processing and neural networks.