Learning Performance of Frequency-Modulation Digital Neural Network with On-Chip Learning

Hiroomi Hikawa, Oita university, Oita, Japan

Abstract

New digital architecture of the frequency-based multilayer neural network (MNN) with on-chip learning is proposed. As the signal level is expressed by the frequency, synaptic multiplier is replaced by a simple frequency converter. Furthermore, the neuron unit uses a voting circuit as the nonlinear adder to have better nonlinear activating function. The back-propagation algorithm is modified for the on-chip learning.

The proposed MNN architecture is implemented on field programmable gate array (FPGA) and the various experiments are conducted to test the performance of the system. The experimental results show that the proposed neuron has a very good nonlinear function owing to the voting circuit. The learning behavior of the proposed MNN is also tested by experiments, which show that the proposed MNN has good learning performance and generalization capabilities.

1. Introduction

One of the effective approach for the hardware implementation of neural network is the pulse stream based architecture which uses stochastic computing [1]-[2]. The stochastic computing is performed with basic logic gates using random pulse sequences as inputs. Synaptic multiplication is performed with a simple AND gate. In stochastic digital neurons, pulses from different synapses are OR-ed together, which provide pulsed nonlinearity. The pulsed nonlinearity is based on statistical saturation and the activation function is easily realized. However, the drawback is that the activation function provided by the pulsed nonlinearity is almost fixed and the accuracy of the pulse mode computing is inferior to that of fully digital arithmetic operation [3].

In this paper a new digital architecture of the multilayer neural network (MNN) with on-chip learning, based on frequency modulation (FM) is proposed. As the signal level is expressed by the frequency, synaptic multiplier is replaced by a simple frequency converter. The synapse unit uses a direct digital frequency synthesizer (DDFS) as the frequency converter. The DDFS is much simpler than numerical multiplier. The proposed neuron unit performs nonlinear addition on the weighted neuron outputs. In order to improve the accuracy of neuron output, a voting circuit is employed for the addition. Furthermore, the voting circuit is enhanced so that the nonlinear activation function is adjustable.

The most important feature of neural networks is their learning ability. Size and real-time considerations show that on-chip learning is necessary for wide range of applications. To provide the on-chip learning, the back-propagation algorithm is modified to have pulse-mode operation. The proposed MNN is implemented on filed programmable gate array (FPGA), and the performance of the MNN is verified by experiments.

2. Multilayer Neural Networks

The operation of the MNN is divided into two phases, i.e., learning phase and retrieving phase. During the learning phase, weights are adjusted to perform a particular application and the learning phase consists of forward operation and backward operation. In the forward operation, the output of the network is calculated from input data and the learning algorithm is performed during the backward operation. In the retrieving phase, the same operation as the forward operation is executed.

2.1. Forward operation

During the forward operation, data from neurons of a lower layer is propagated forward to neurons in the upper layer via feed-forward connection network. Let $o_k^{(s)}$ denote the output of k-th neuron of the s-th layer, then the computation performed by each neuron is

$$H_k^{(s)} = \sum_{j=1}^{N_{s-1}} w_{kj}^{(s)} o_j^{(s-1)} + \theta_k^{(s)} \tag{1}$$

$$o_k^{(s)} = f(H_k^{(s)}) \tag{2}$$

where layers are numbered from 0 to M, $H_k^{(s)}$ is the weighted sum of the k-th neurons in the s-th layer and $w_{kj}^{(s)}$ is the synaptic weight. The output $o_k^{(s)}$ of the neuron is obtained by computing an activation function $f(\cdot)$ on the weighted sum. Usually sigmoid function is used as the nonlinear activation function $f(\cdot)$.

2.2. Back-propagation algorithm

Training algorithm is performed in the backward operation. The back-propagation algorithm is the most widely used training algorithm in MNN. First, the forward operation is executed to obtain the ouput response against the input training pattern. Then error between the training data and the actual ouput value is propagated in backward, and the error is used to update the synaptic weights. The back-propagation algorithm is expressed by,

$$\sigma_k^{(s)} = \begin{cases} t_k - o_k^{(s)} & s = M \\ \sum_{j=1}^{N_{s+1}} w_{kj}^{(s+1)} \delta_j^{(s+1)} & s = 1, \cdots M-1 \end{cases} \tag{3}$$

0-7803-4859-1/98 $10.00©1998 IEEE

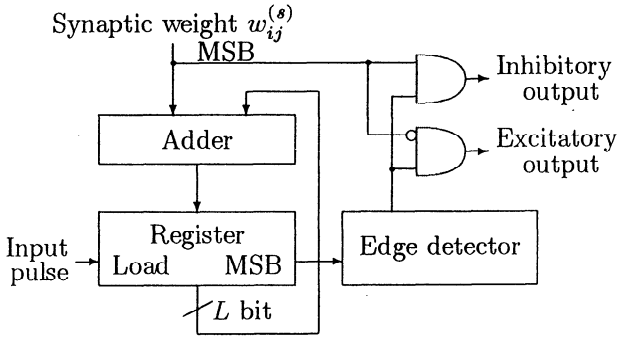

Figure 1: Synapse unit with DDFS

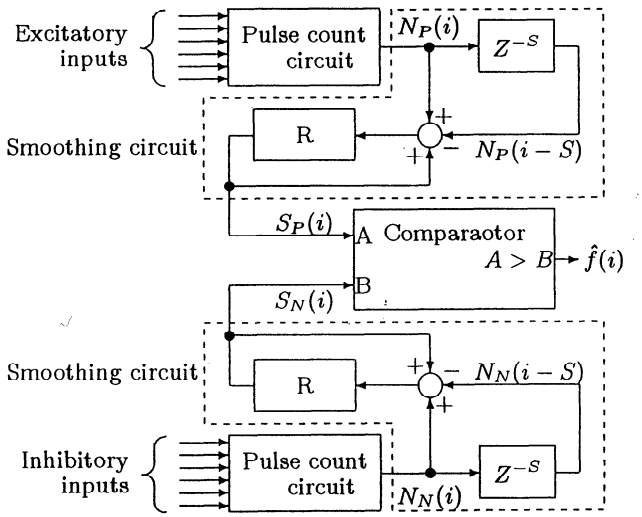

Figure 2: Voting neuron with smoothing circuit

$$\delta_k^{(s)} = \sigma_k^{(s)} f'(H_k^{(s)}) \qquad s = 1, \cdots, M \tag{4}$$

$$\Delta w_{kj}^{(s)} = \eta \delta_k^{(s)} o_j^{(s-1)} \tag{5}$$
$$k = 1, \cdots N_s \quad j = 1, \cdots N_{s-1}$$

where, $f'(\cdot)$ is the derivative of the activation function.

3. Frequency Based Multilayer Neural Network

Proposed MNN uses two computational elements, one is a synapse unit and the other is a neuron unit. The synapse unit performs the synaptic weight multiplication and the neuron unit performs nonlinear addition on the weighted neuron outputs. These units also provide the on-chip learning capability.

3.1. Synapse unit

In the synapse unit, neuron output is multiplied by a synaptic weight. As the proposed network uses frequency to represent the signal levels, the multiplier is replaced by a programmable frequency converter. Block diagram of the synapse unit is depicted in Figure 1. The synapse unit uses a direct digital frequency synthesizer (DDFS) as the frequency converter. The DDFS is much simpler than numerical multiplier. DDFS consists of two parts: an adder and a register. The adder gives an increment of K_W to the register in every sampling period of T_I that is the reciprocal of the input frequency f_I. The most significant bit (MSB) of the register is taken as the output of the DDFS. Thus the output cycle is equal to the interval at which the content of the register exceeds 2^L and overflows occur. Therefore, the frequency of register's MSB is given by:

$$f_{MSB} = \frac{K_W}{2^L} f_I \tag{6}$$
$$0 \le K_W < 2^{L-1} \text{ (MSB of } K_W \text{ is 0)}$$

$$f_{MSB} = \frac{2^L - K_W}{2^L} f_I \tag{7}$$
$$2^{L-1} \le K_W < 2^L \text{ (MSB of } K_W \text{ is 1)}$$

where L is the bit length of the register. Equations (6) and (7) show that the maximum frequency of the DDFS is the half of input frequency, i.e., the valid weight value is between 0.0 and 0.5. To enhance the weight range the edge detector is employed, which detects the low-to-high and high-to-low transitions of the MSB and gives output pulse at both edges of the MSB signal. The synapse unit has two kind of output signals, i.e., excitatory (positive) and inhibitory (negative) outputs. The DDFS output is selected as the excitatory signal when the MSB (sign bit) of K_W is 0, otherwise it is used as the inhibitory signal. Hence the valid weight range is between -1.0 and 1.0. The weight value is represented in two's complement format.

3.2. Neuron unit

In the neuron unit weighted neuron outputs in the lower layer are summed up and the output is generated using the activation function $f(\cdot)$. In the stochastic neuron [2], excitatory and inhibitory synaptic outputs are summed by OR gates. Then these signals are used to generate the neuron output. The problem is that the neuron output pulse is canceled whenever the inhibitory pulse is applied. For example, a single inhibitory pulse prevent the neuron from generating outpout pulse even though the number of excitatory input pulse is much more than that of inhibitory pulse. To alleviate the problem a voting circuit is employed as the nonlinear adder in the proposed MNN. The voting circuit gives a single output pulse when the number of excitatory pulses exceeds the number of inhibitory pulse. Block diagram of the proposed neuron unit is depicted in Fig. 2. The voting circuit consists of a comparator and a pulse count circuit which counts the number of '1's in the input signals. As well as the stochastic neuron, the voting neuron takes advantage of the *statistical saturation*, which provides the nonlinear addition to realize the nonlinear activation function $f(H_k)$. As long as pulses are not frequent (low signal level), the pulse count at the output of

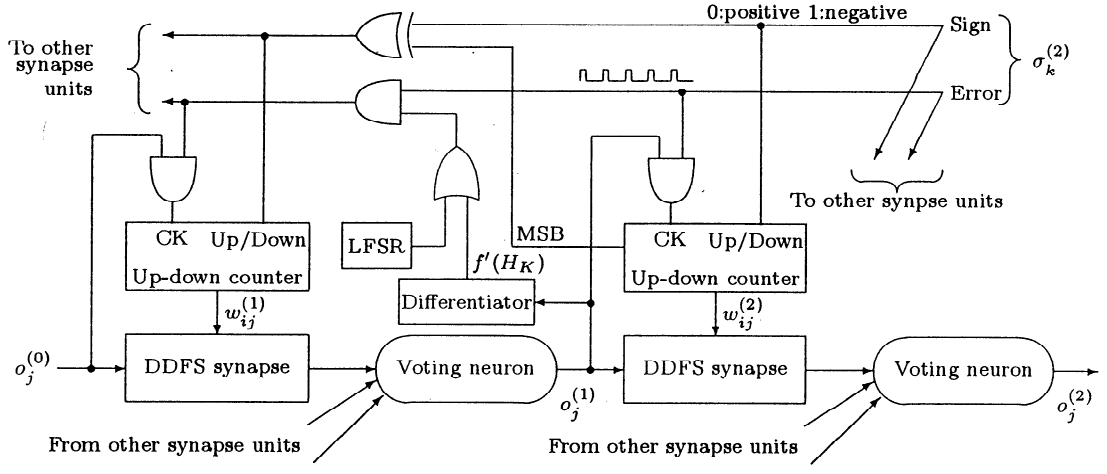

Figure 3: Configuration of the proposed MNN

the voting circuit equals the net sum of individual pulse counts at the inputs. However, when pulses become more frequent (higher signal level), the chance of pulses masking each other becomes significant. Total pulse count then saturates to a maximum. The drawback of this simple method is that the shape of the activation function is almost fixed [3]. To change the characteristic of the non-linear function, a smoothing circuit is added between the pulse count circuit and the comparator. The output of the smoothing circuit $S_P(i)$ and $S_N(i)$ are:

$$S_P(i) = \sum_{k=0}^{S-1} N_P(i-k) \quad (8)$$

$$S_N(i) = \sum_{k=0}^{S-1} N_N(i-k) \quad (9)$$

where, $N_P(i)$ and $N_N(i)$ are the number of the excitatory pulse at i$-th$ sample and the number of the inhibitory pulse at i$-th$ sample, respectively. The neuron output at i$-th$ sample $\hat{f}(i)$ is,

$$\hat{f}(i) = \begin{cases} 1 & S_P(i) > S_N(i) \\ 0 & S_P(i) \leq S_N(i) \end{cases} \quad (10)$$

3.3. On-chip learning

To provide the on-chip learning, the back-propagation algorithm is modified to have pulse-mode operation for the effective hardware implementation. System configuration of the proposed MNN architecture with on-chip learning is depicted in Fig. 3. The upper half of Fig. 3 is the on-chip learning circuit, i.e., the hardware implementation of the back-propagation algorithm.

As well as the forward signals, the error terms $\sigma_k^{(s)}$ and $\delta_k^{(s)}$ in (3) are represented by pulse signals. The error term is propagated through two signal lines, `error` and `sign`. When the teaching signal and the output signal take different values, error pulse signal is generated and transferred in the `error` line, and the `sign` signal indicates the sign of the error. As shown in Fig. 4, during the learning phase, the MNN uses two kind of time slots, one is for the forward operation and the other is used for the backward operation. Pulse signals related to the forward operations, such as input, synapse output and the neuron's output pulse sequence are placed in the forward time slot. Then, the error pulse signals are generated and they are propagated to the lower layer using the backward time slot.

The back-propagation algorithm uses the derivative of the activation function. A pulse differentiator is employed to generate the derivative $f'(H_k)$. As shonw in Fig. 4, the differentiator gives output pulse every time it finds the head and the last of pulse stream. It is reported that the learning performance of the MNN with back-propagation algorithm can be improved by adding an offset $G(< 1.0)$ to $f'(H_k)$ [4]. A random pulse generator is used to add the offset G. A pseudo

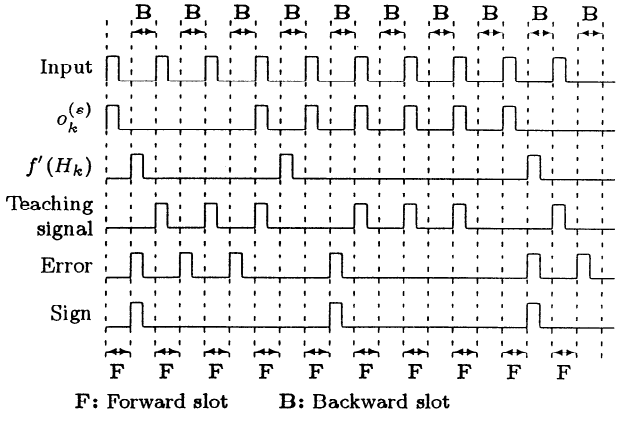

Figure 4: Signals in the proposed MNN

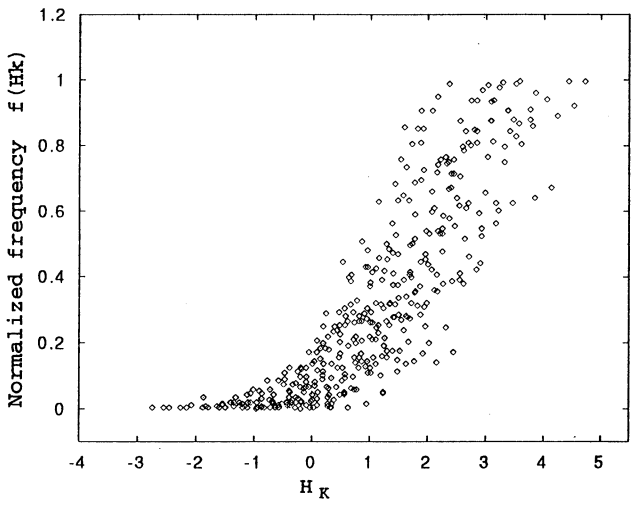

Figure 5: The nonlinear functions of stochastic neurons with DDFS weights.

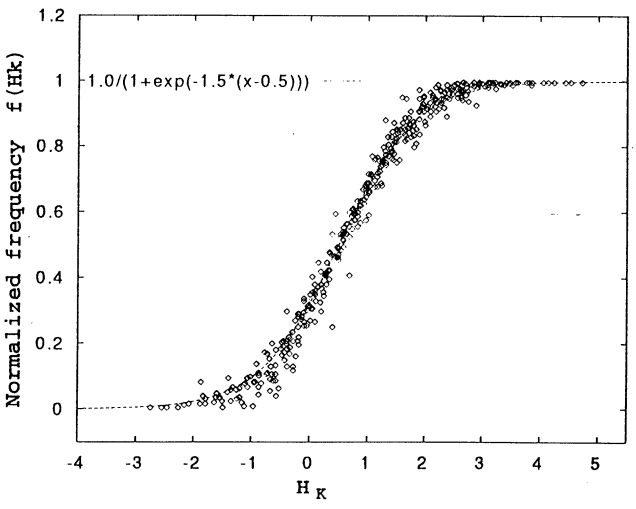

Figure 6: The nonlinear functions of voting neurons with DDFS weights.

random sequence of bits with the probability $P(1) = G$ of having a '1' in the sequence is added to $f'(H_k)$ by using OR gate. Linear feedback shift register (LFSR) is employed for the generation of the pseudo random sequence. These signals are used to update the up-down counters that contain synaptic weights. To calculate $w_{kj}^{(s+1)}\delta_j^{(s+1)}$ in (3), instead of the actual weight value, the sign bit (MSB) of the weight is multiplied by the error term $\delta_j^{(s+1)}$. As the sign bit takes two values (zero for positive, one for negative), the multiplication is significantly simplified. The circuit to perform the operation is a single exclusive-OR gate, which inverts the `sign` signal when the MSB of the weight is one. The multiplication $\sigma_k^{(s)} f'(H_k^{(s)})$ in (4) and the multiplication of $\delta_k^{(s)} o_j^{(s-1)}$ in (6) are realized by logical-AND gates using stochastic multiplication.

4. Experiments

The proposed MNN is implemented on FPGA[1] and experiments are conducted to verify the feasibility of the proposed architecture. The weight value is expressed in 9-bit signed fixed-point format and the size of register used in the DDFS is 9-bit. The experiments with the stochastic weight multiplier and neuron are also performed. The stochastic random weight multiplier uses the random pulse generator based on 8-order LFSR and the random weight circuit takes 9-bit weights (the additional bit is a sign bit).

4.1. Neuron characteristics

First, the characteristics of the activation function are measured. The relation between H_k and $f(H_k)$ of the stochastic neuron and the voting neuron are depicted in Figures 5 and 6. The voting neuron used here does not include the smoothing circuit. It is very clear that the characteristic of the voting neuron is much better than that of the stochastic neuron. The characteristic of the proposed voting neuron has a very smooth, sigmoid-like function, whose output variance is much smaller than that of the stochastic neuron.

The second experiments measure the effect of the smoothing circuit on the neuron's non-linear function. The neuron characteristics are measured using the same configuration as the previous experiments. The experimental results with $S = 8$ and $S = 24$ are depicted in Fig. 7. For the comparison, neuron characteristics with the stochastic weight multiplier are also tested. These figures show that the effect of the smoothing circuit is more significant with the DDFS-type synapse unit. Even though the DDFS weight values are restricted between -1.0 and 1.0, the flexibility of design is greatly improved with this enhancement.

4.2. On-chip learning performance

The pulse differentiator employed to generate $f'(H_k)$ for the back-propagation algorithm finds the head and the last of the pulse train and generates output pulses. Using the neuron signal shown in Fig. 6 as the input, the differentiator output is shown in Fig. 8, which is very close to the derivative of the nonlinear characteristic shown in Fig. 6.

The proposed MNN is trained to perform simple binary logic functions. The MNN used for the experiment has three neurons in the input layer, three neurons in a hidden layer and a single output neuron. The input and the hidden layer both include the offset neuron which always gives '1' output so that the weights connected to the offset neuron act as the offset θ. DDFS synapse unit is used for weighting the neuron output. The parameter S used in the smoothing circuit is eight. The whole MNN circuit and a trainer unit is implemented on a single FPGA.

[1]XILINX, XC4013

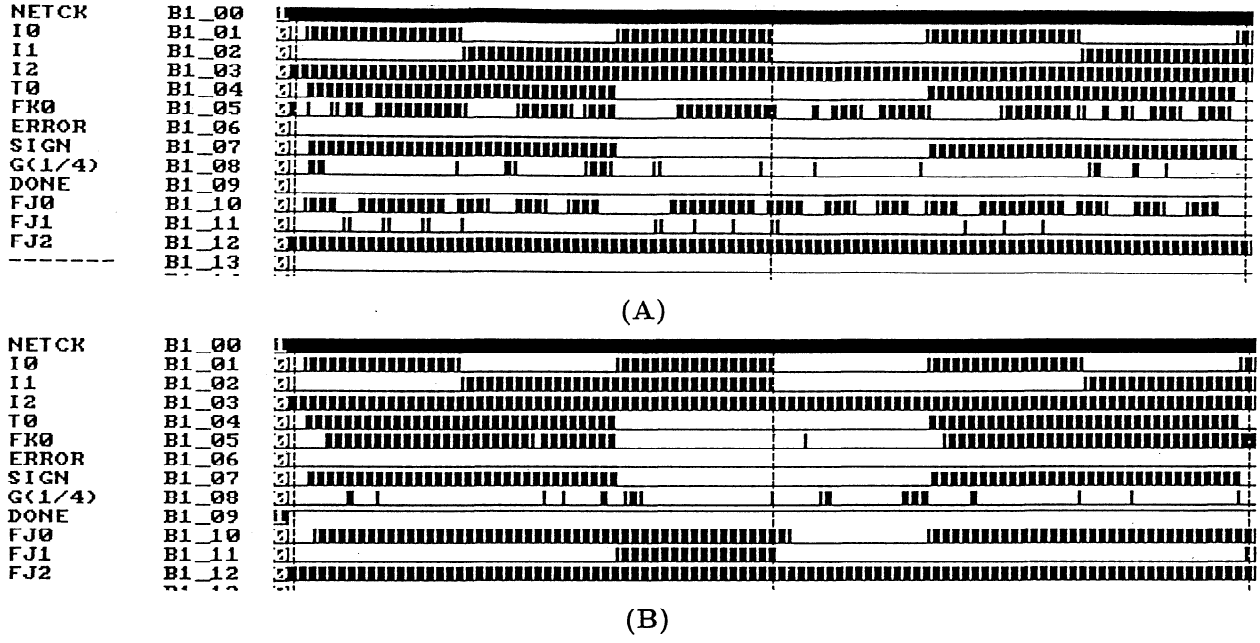

Figure 9: Signals in the MNN learning logic functions, (A) before learning, (B) after learning (exclusive-OR function).

Signals in the experimental MNN during the learning are depicted in Fig. 9. In the figure, a simple logic function (exclusive-OR) is used as the target function. I0, I1 are the input signals, T0 is the teaching signal, and FK0 is the output of the MNN ($o_0^{(2)}$). As the figures show, the proposed MNN successfully learned the exclusive-OR function.

Next experiment is the problem of classifying points inside and outside a shaped region [5]. The MNN used in this experiment has two inputs and one output, which given the x, y coordinates of a point within the square region bounded by $0 \leq x < 1$ and $0 \leq y < 1$, is trained to recognize if the given point lies inside or outside a particular shaped region. The network output is zero if the point lies outside, one if it lies inside. The network has three neurons in its input layer, and a single hidden layer that contains five neurons. The output layer has a single neuron. The third neuron in the input layer and the fifth neuron in the hidden layer are offset neurons to provide offset θ. The whole network and the trainer unit takes two FPGAs. Using a training data set, the network learned to classify the given coordinate (x, y) by itself using its on-chip learning mechanism. The training data set consists of 128-data which is randomly selected from the original data set (256 data) and after the learning, all 256 reference points are fed to the network to test the generalization capability. Fig. 10 shows the training sets and the response of the MNN after 8192 iterations of training. Output levels are represented by the diameter of black circles. As the parameter S increases, the output pattern becomes clear shape and with $S = 24$, the output plot resembles very much the target shape even though the training data is incomplete. Thus, these experiments indicate that the proposed MNN has very good generalization capability as well as the on-chip learning mechanism is very functional.

5. Conclusion

New digital architecture of the frequency-based MNN with on-chip learning has been discussed. The proposed MNN architecture is implemented on FPGAs and the various experiments are conducted to test the performance of the system. First, the neuron characteristics are measured by the experiments, and the results show that the proposed MNN has a very good nonlinear function owing to the voting circuit. The learning behavior of the on-chip learning capability of the MNN is also tested by experiments, which show that the proposed MNN has good learning performance and generalization capabilities.

References

[1] G. Moon, M. E. Zaghloul, and R. W. Newcomb, "VLSI Implementation of Synaptic Weighting and Summing in pulse Coded Neural-Type Cells," *IEEE Trans. on Neural Networks*, vol. 3, no. 3, pp.394-403, May. 1992.

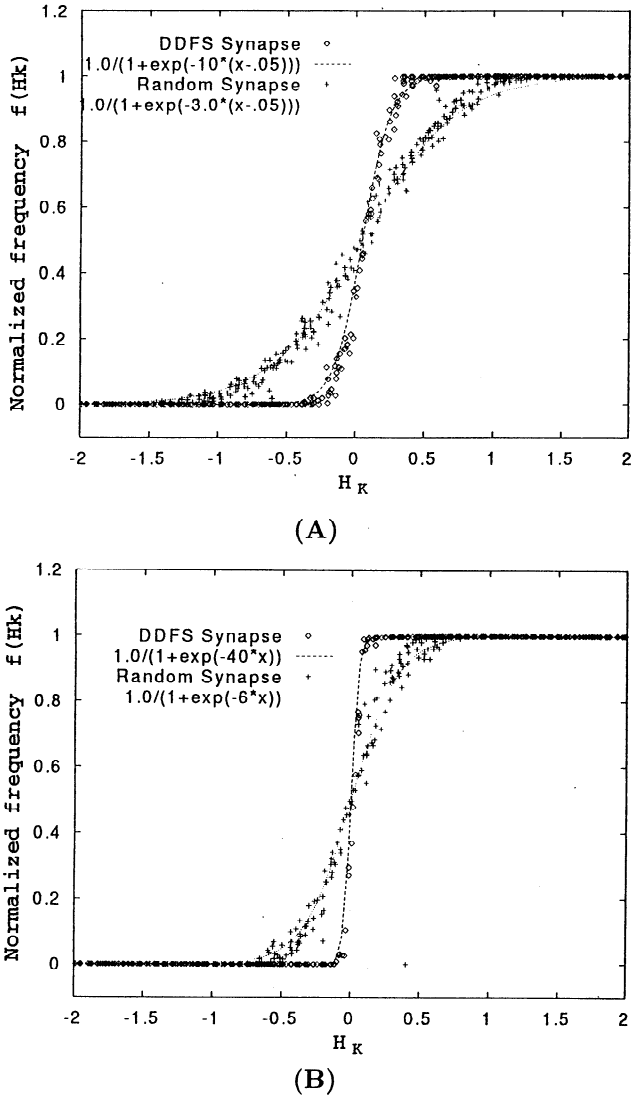

Figure 7: The nonlinear functions of the voting neuron (A) $S = 8$, (B) $S = 24$.

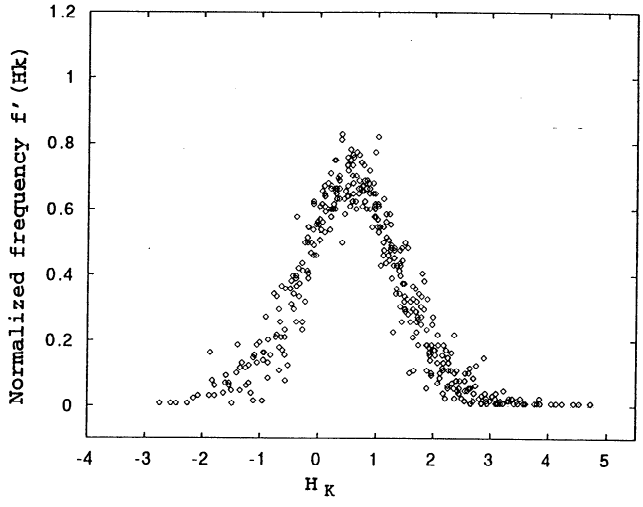

Figure 8: The differentiator output.

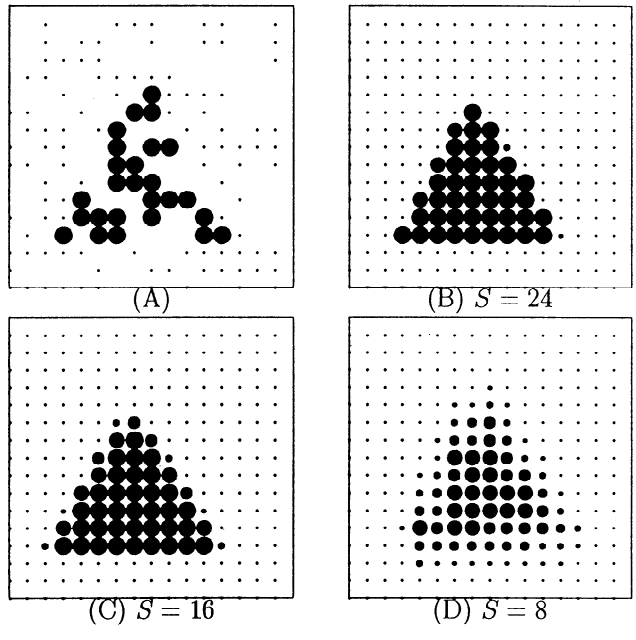

Figure 10: Training data and the response in the classifying problem, (A) training data (128 pixels), (B) after the learning (S=24), (D) after the learning (S=16), (F) after the learning (S=8).

[2] Y. C. Kim, M. A. Shanblatt, "Random noise effects in pulse-mode digital multilayer neural networks," *IEEE Trans. on Neural Networks*, Vol. 6, No. 1, pp.220-229, January 1995.

[3] Leonardo M. Reyneri, "A performance analysis of pulse stream neural and fuzzy computing systems," *IEEE Trans. on CAS*, Vol. 42, No. 10, pp.642-660, Octover 1995.

[4] H. Hikawa, "Improvement on the learning performance of multiplierless multilayer neural network," *Proc. IEEE ISCAS'97*, Vol.1, pp.641-644, 1997.

[5] M. Marchesi, G. Orlandi, F. Piazza and A. Uncini, "Fast neural networks without multipliers," *IEEE Trans. on Neural Networks*, Vol.4, No.1, Jan. 1993.