# Fault Tolerant Design of Neuro-Processors Using Weight Limitation and Ternary Output

Nobuhiro Tomabechi[*] and Yoshichika Fujioka[*]
[*] Hachinohe Institute of Technology, Japan

***Abstract*** **This paper presents a fault tolerant design of hardware-type neural networks for real time control usage combining the following two methods; (1) a method to reduce the effect of a fault by weight limitation of synapses and (2) a method to reduce the effect of a fault by setting the output of the faulty neuron to the middle level of the ternary logic. Fault simulation is carried out on a numeric pattern recognition system that is implemented using a 3-layered feed-forward neural network. Fault generation is assumed to occur on a neuron rather than an interconnection line. It is demonstrated that a fault tolerant design of neural networks to cover all of the neurons included in the input layer, intermediate layer and output layer can be carried out by combining the weight limitation and the ternary output.**

## I. INTRODUCTION

In this paper, the fault tolerant design of hardware-type neural networks for real time control usage is studied. Concerning fault tolerant design of neural networks, several studies have been reported [1]-[4], however a fault is almost assumed to occur on an interconnection line and is also assumed to occur in the intermediate layer.

This paper presents a fault tolerant design of neural networks combining the following two methods; (1) a method to reduce the effect of a fault by weight limitation of synapses and (2) a method to reduce the effect of a fault by setting the output of the faulty neuron to the middle level of the ternary logic.

Fault generation is assumed to occur on a neuron rather than an interconnection line, that is, all of the interconnection lines connected to the output of the faulty neuron will be failed. Fault generation is also assumed to occur anywhere in the network including the input layer, intermediate layer, and output layer.

Fault simulation is carried out on a numeric pattern recognition system which is implemented using a 3-layered feed-forward neural network.

It is demonstrated that a fault tolerant design of neural networks to cover all of the neurons included in the input layer, intermediate layer and output layer can be carried out by combining the weight limitation and the ternary output.

## II. TARGETED SYSTEMS AND FAULT MODEL

### *A. Targeted Systems*

This paper studies concerning the fault tolerant design of hardware-type neural networks for real time control usage. Three-layered feed-forward neural networks are dealt with. Following 3 character pattern recognition systems are implemented using a 3-layered feed-forward neural network and their fault tolerant abilities are simulated.

1. 7-segment numeric pattern recognition system
2. 12-segment numeric pattern recognition system
3. 12-segment alpha-numeric recognition system

Only the results on the 7-segment system will be shown here, since its input data has the least redundancy and hence it is most severely affected by the fault compared with other two systems.

Fig. 1 shows a model of a neuron. Let us express the input as $I_i$, the weight of a synapse as $W_i$, the threshold as $H$, and the output as $O$.

Then the function of a neuron is expressed by the following equations.

$$x=(\sum_i W_i I_i)-H \qquad (1)$$

$$O=1/(1+e^{-x})$$

In Fig. 2, a 7-segment numeric pattern recognition system is illustrated.

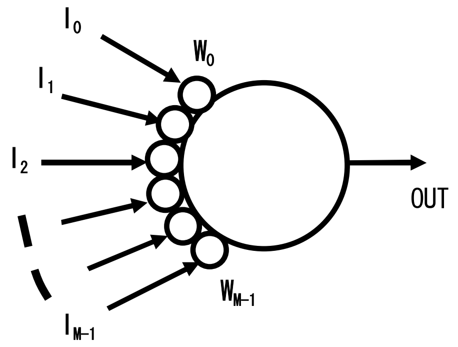


Fig.1 Neuron model

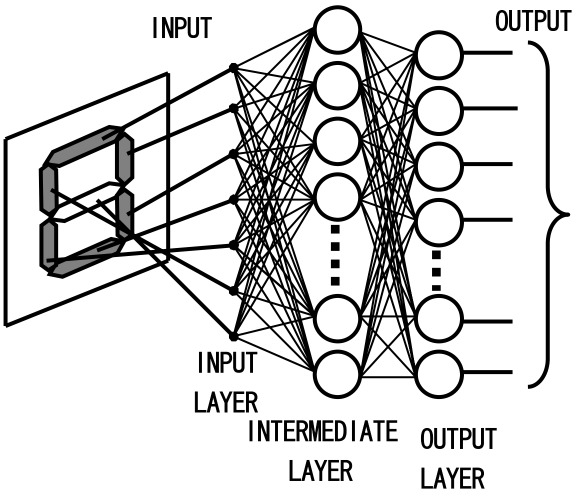


Fig. 2 7-segment numeric pattern recognition system

The learning of the system is carried out by the error back propagation method. The calculation is done by the following equations, where $T_j$ ($j$=0~9), $O^O_j$, $O^I_j$, $W_j$, and $\varepsilon$ denote a teacher signal, the output of a neuron in the output layer, the output of a neuron in the intermediate layer, a weight of a neuron, and the weight changing coefficient, respectively.

$$d^O_j=(O^O_j-T_j)O^O_j(1-O^O_j) \quad \text{for output layer} \quad (2)$$

$$d^I_j=\sum_a d^O_a W^I_{j,a} O^I_j(1-O^I_j) \quad \text{for intermediate layer} \quad (3)$$

$$dW=-\varepsilon\, d_j O_j \quad (4)$$

$$W_j=W_j+dW. \quad (5)$$

Calculations of (1)~(5) are repeated changing input data to all of the numeric patterns. The recognition error, $R$ is given by

$$R=\sum_j (O^o_j-T_j)^2. \quad (6)$$

Calculation of (1)~(6) are repeated until $R$ reaches $R<R_{min}$.

## B. Fault Model

The following model will be adopted here.

(1) A fault occurs on a neuron rather than on an interconnection line, that is, all of the interconnection lines connected to the output of the faulty neuron will fail.

(2) A fault can occur anywhere in the network including the input layer, intermediate layer and output layer.

(3) The output of the faulty neuron is fixed at 0 in the ordinal case or is fixed at 0.5 in the ternary output method. Fault detection will be discussed in another paper.

## C. Estimation of Fault Tolerant Ability

Let us express the number of neurons in the intermediate layer as $I_M$. In general, greater $I_M$ results in greater fault tolerance. $I_M$ will be taken as 6~10.

Let us express input data as $I(i)_k$ and the corresponding output of the output layer as $O(i,j)_k$, where i, k, and j denote the sequential number of input data, the location of the fault, and the terminal number of the output layer, respectively. In normal operation, following relationship stands.

If i=j then $O(i,j)_k \cong 1$ else $O(i,j)_k \cong 0$ where k=0~$I_M$-1. Hence, we can judge that there is no recognition error as follows.

[judgment] If $O(i,i)_k \geq 0.5$ for any i and k, there is no recognition error, otherwise there is recognition error.

Let us refer to O(i, i) as the coincident point output.

To estimate the degree of fault tolerant ability, we will take the distribution of O(i,i) value. CO(N) (N=0~9) denotes the total number of O(i,i) where $N+1>O(i,i) \geq N$, k=0~$I_M$-1 and i=0~9.

The sum of CO(N) (N=0~4) indicates the number of recognition errors and CO(5) indicates the number of critical cases where there is no error but there is a high possibility of generating a recognition error.

## D. Fault Simulation for the System without Fault Protection

As shown later in Table 2, we find O(i,i)<0.5, that is, recognition errors even if $I_M \geq 10$.

Table 1 O(i,i) of the system using S-figure function weight limitation the case of $B=20, W_0=1.5, I_M=10$

| k | O(0,0) | O(1,1) | O(2,2) | O(3,3) | O(4,4) | O(5,5) | O(6,6) | O(7,7) | O(8,8) | O(9,9) |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.93 | 0.86 | 0.88 | 0.90 | 0.85 | 0.87 | 0.95 | 0.91 | 0.91 | 0.93 |
| 1 | 0.81 | 0.95 | 0.94 | 0.89 | 0.94 | 0.95 | 0.81 | 0.91 | 0.79 | 0.92 |
| 2 | 0.82 | 0.85 | 0.96 | 0.93 | 0.93 | 0.71 | 0.94 | 0.78 | 0.90 | 0.93 |
| 3 | 0.93 | 0.89 | 0.96 | 0.93 | 0.86 | 0.94 | 0.74 | 0.76 | 0.73 | 0.92 |
| 4 | 0.87 | 0.95 | 0.96 | 0.94 | 0.94 | 0.93 | 0.92 | 0.95 | 0.96 | 0.95 |
| 5 | 0.88 | 0.96 | 0.81 | 0.93 | 0.88 | 0.94 | 0.93 | 0.93 | 0.95 | 0.79 |
| 6 | 0.93 | 0.94 | 0.96 | 0.77 | 0.96 | 0.95 | 0.93 | 0.95 | 0.81 | 0.86 |
| 7 | 0.93 | 0.95 | 0.93 | 0.91 | 0.85 | 0.93 | 0.95 | 0.91 | 0.78 | 0.78 |
| 8 | 0.94 | 0.88 | 0.96 | 0.78 | 0.96 | 0.93 | 0.86 | 0.75 | 0.91 | 0.82 |
| 9 | 0.94 | 0.95 | 0.96 | 0.93 | 0.90 | 0.84 | 0.82 | 0.76 | 0.74 | 0.84 |

## III. SYSTEM USING WEIGHT LIMITATION

The recognition error by a fault may be generated on the synapses with large weight. Hence, the influences of a fault will be reduced by limiting the weight of synapses in a small range. The weight limitation is effective only for faults on neurons in the intermediate layer, and hence, the weight limitation is applied to the synapses of neurons only in the output layer.

### *A. Weight Limitation Function*

(1) Step function

As the weight limiting function, the step function will be considered first. This is expressed by

$W_0 \geq W \geq W_0,$

where $W$ and $W_0$ denote the weight of a synapse and a limited value, respectively.

The above condition will be realized by stopping the addition of $dW$ to $W_j$ in Eq. 5.

As shown later in Table 2, recognition error can be suppressed by using the step function limitation when $I_M \geq 9$. However, the convergence is not assured, that is, it is frequently observed that $R$ does not continuously decrease and does not reach $R_{min}$.

(2) S-figure function

In order to maintain the convergence, this paper proposes S-figure function, which is expressed bellow, where $W$, $W_0$, and $B$ denote the weight, a limited value, and the base of an exponential function, respectively.

$$\sigma = 1/(1+B^{(W-W0)}) \quad (W \geq 0)$$
$$= 1/(1+B^{(-W-W0)}) \quad (W<0)$$

In Fig.3, S-figure function is shown for the case of $B=20$ and $W_0=1.8$. Weight limitation using $\sigma$ is realized by changing Eq. 4 to Eq. 7.

$$dW = -\sigma\varepsilon\, d_j O_j \qquad (7)$$

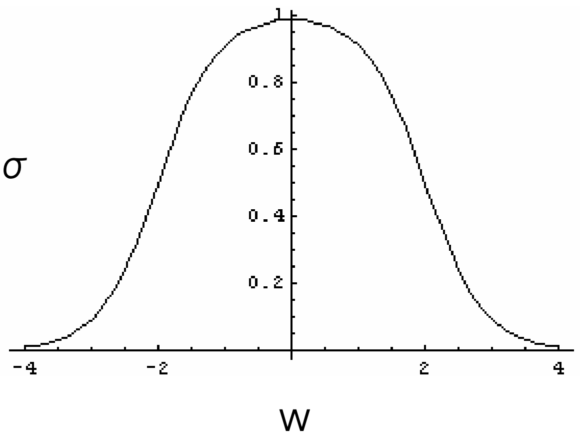


Fig. 3 S-figure function the case of $B=20, W_0=1.8$

### *B. Fault Simulation*

In Table 1, $O(i,i)_k$ ($k=0\sim I_M-1$) is shown for fault injection into neurons in the intermediate layer, where k denotes the location of the faulty neuron.

From Table 1, we see that no recognition error is generated by using S-figure function limitation.

Table 2 distribution of O(i,i) the case of $I_M=10$

| | no limitation | step function limitation | S-figure function limitation |
|---|---|---|---|
| CO(0) | 0 | 0 | 0 |
| CO(1) | 2 | 0 | 0 |
| CO(2) | 1 | 0 | 0 |
| CO(3) | 0 | 0 | 0 |
| CO(4) | 1 | 0 | 0 |
| CO(5) | 2 | 0 | 0 |
| CO(6) | 2 | 6 | 0 |
| CO(7) | 3 | 17 | 14 |
| CO(8) | 7 | 10 | 23 |
| CO(9) | 82 | 67 | 63 |

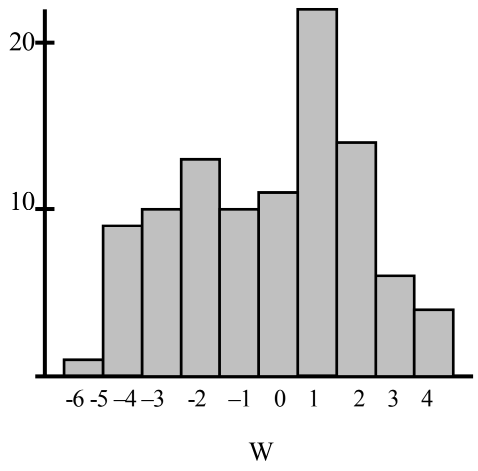
Table 2 shows the distribution of O(i,i), where 3 cases of no weight limitation, step function limitation, and S-figure function limitation are shown. From Table 2, we see that there are 4 recognition errors in the system without fault protection, and also see that the possibility of recognition error in the case of S-figure function is smaller than that of step function.

So we can state that;

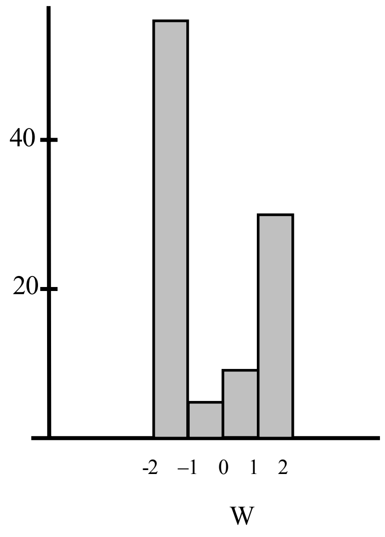
[Result 1] By applying a weight limitation, the recognition error caused by a fault in the intermediate neurons can be suppressed. By applying S-figure function limitation, the possibility of generating a recognition error can be reduced compared with the case of applying the step function limitation.

[Consideration 1] Since the S-figure function has not only a sharp rise and fall shape but also continuity, both the weight limitation and the convergence are satisfied.

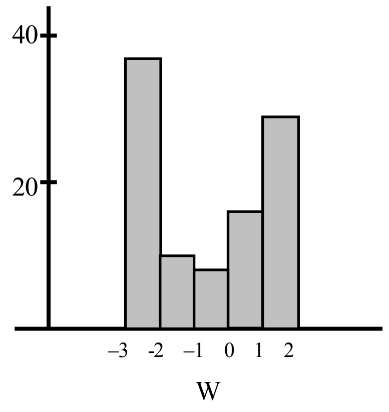
In Fig. 4 (a), (b) and (c), examples of the weight distribution are shown for the case respectively, without weight limitation, with the step function limitation, and with the S-figure function limitation. We see that the S-figure function limitation leads to a more uniform distribution than that of the step function limitation.



(a) Without weight limitation



(b) Step function limitation



(c) S-figure function limitation

Fig.4 weight distribution

If we have a weight limitation function to lead to a more flat distribution, the convergence will be more improved.

## IV. SYSTEM USING TERNARY OUTPUT

The weight limitation is effectively applied only to a fault in the intermediate layer and not to a fault in other layers. We propose a method in which the output of the faulty neuron is fixed at 0.5. This method is applicable to neurons in all layers.

When the normal output is 0, the output value is not changed by fault injection and nothing will occur. On the other hand, when the normal output is 1, output value 1 is changed to 0 by fault injection. In the latter case, the influence of the fault can be reduced to one-half by setting the output of the faulty neuron at 0.5.

### A. *Fault Tolerance of Input Layer*

There are only connecting points instead of neurons in the input layer. However, we will deal with the situation that the output value of a faulty connecting point is fixed at 0.5.

In Table 4, fault simulation of the numeric pattern recognition system is shown. From Table 4, we can state that;

[Result 2] By setting the faulty output at 0.5, recognition error can be greatly reduced, but it cannot be reduced perfectly to 0.

[Consideration 2] The reason why no recognition error cannot be realized is as follows.

As shown in Fig. 5, if segment A is faulty, then the numeric pattern for number 8 and that for number 9 cannot be distinguished, and also if segment B becomes faulty, the numeric pattern for number 0 and that for number 8 cannot be distinguished. To reduce the recognition error to 0, another numeric pattern system other than a 7-segment one, for example a 12-segment one, must be used.

### B. *Fault Tolerance of Output Layer*

By using ternary output, a fault generated in the output layer is easily processed.

1. When the normal output of the faulty neuron is 0, there will be another output whose value is nearly equal to 1, so the faulty output 0.5 can be neglected.
2. When the normal output of the faulty neuron is 1, there will be no output whose value is greater than 0.5, so the faulty output 0.5 is recognized as 1.

Table 4 O(i,i) distribution using ternary output for input layer the case of $I_M$=10

| | ordinary output (faulty output is 0) | ternary output (faulty output is 0.5) |
|---|---|---|
| CO(0) | 10 | 0 |
| CO(1) | 5 | 0 |
| CO(2) | 1 | 1 |
| CO(3) | 2 | 1 |
| CO(4) | 0 | 2 |
| CO(5) | 2 | 7 |
| CO(6) | 5 | 10 |
| CO(7) | 4 | 8 |
| CO(8) | 8 | 11 |
| CO(9) | 33 | 30 |

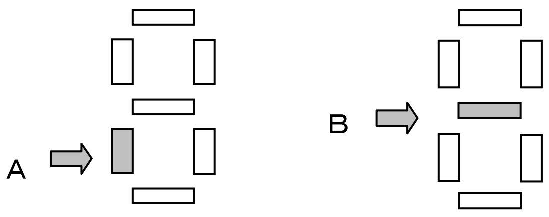


Fig. 5 undistinguished faults

### C. *Fault Tolerance of Intermediate Layer*

Table 5 shows the distribution of O(i,i) when a fault is injected into the intermediate layer for both the case of ternary output only and the case of combining ternary output and weight limitation. From Table 5, we conclude the following.

[Result 3] By using ternary output, the fault tolerance ability is greatly enhanced. When $I_M \geq 14$, recognition error can be suppressed. However, there is the possibility of easily generating recognition errors since CO(5) is not 0.

[Result 4] By combining ternary output and weight limitation, recognition error can be completely avoided in the entire range of $I_M$. In addition, there is less possibility of generating recognition error since CO(5) is 0.

[Consideration 3] Result 3 and Result 4 imply that weight limitation and ternary output enhance fault tolerance not only individually but also cooperatively.

Table 5 O(i,i) distribution using ternary output for intermediate layer the case of $I_M$=14

| | ordinary output (faulty output is 0) | | ternary output (faulty output is 0.5) | |
|---|---|---|---|---|
| | no weigh limitation | weight limitation | no weigh limitation | weight limitation |
| CO(0) | 0 | 0 | 0 | 0 |
| CO(1) | 1 | 0 | 0 | 0 |
| CO(2) | 2 | 0 | 0 | 0 |
| CO(3) | 1 | 0 | 0 | 0 |
| CO(4) | 4 | 0 | 0 | 0 |
| CO(5) | 2 | 0 | 4 | 0 |
| CO(6) | 6 | 2 | 6 | 0 |
| CO(7) | 7 | 16 | 14 | 15 |
| CO(8) | 18 | 23 | 40 | 53 |
| CO(9) | 99 | 99 | 76 | 72 |

## V. CONCLUSIONS

This paper has presented a fault tolerant design of hardware-type neural networks combining the following two methods; (1) a method to reduce the effect of a fault by weight limitation of synapses and (2) a method to reduce the effect of a fault by setting the output of the faulty neuron to the ternary output.

It is demonstrated that a fault tolerant design of neural networks that covers all of the neurons included in the input layer, intermediate layer and output layer can be achieved by combining the weight limitation and the ternary output.

Next subjects will be studied in the future.

(1) trade off of the fault tolerance between the weight limitation and the ternary output

(2) realization of a more effective weight limitation function

## REFERENCES

[1] Y. Tan and T. Nanya, “Fault-tolerant back-propagation model and its generation ability”, Digest IJCNN, pp. 2516-2519, 1993.

[2] N. C. Hammadi and H. Ito, "A learning algorithm for fault tolerant feedforward neural networks", IEICE Trans. INF. & SYST., Vol. E80-D, No. 1, pp. 21-27, 1997-1.

[3] N. Tomabechi, “Hierarchical redundancy design of WSI oriented high-speed neuro-processors”, IEICE Japan Trans. D-I, Vol. J81-D-I, No. 7, pp. 933-936, 1997-7.

[4] M. Nishigaki, T. Tsuzuki and M. Soga, "A fault tolerant learning algorithm considering the worst-case fault of neural networks", Proc. 2000 IEEE Int. Symp. on Intelligent Signal Processing and Communication Systems, pp. 1045-1050, 2000-11.