PAPER Special Issue on Fault-Tolerant Computing

A Learning Algorithm for Fault Tolerant Feedforward Neural Networks

Nait Charif HAMMADI† and Hideo ITO††, Members

SUMMARY A new learning algorithm is proposed to enhance fault tolerance ability of the feedforward neural networks. The algorithm focuses on the links (weights) that may cause errors at the output when they are open faults. The relevances of the synaptic weights to the output error (i.e. the sensitivity of the output error to the weight fault) are estimated in each training cycle of the standard backpropagation using the Taylor expansion of the output around fault-free weights. Then the weight giving the maximum relevance is decreased. The approach taken by the algorithm described in this paper is to prevent the weights from having large relevances. The simulation results indicate that the network trained with the proposed algorithm do have significantly better fault tolerance than the network trained with the standard backpropagation algorithm. The simulation results show that the fault tolerance and the generalization abilities are improved.

key words: *feedforward neural network, learning algorithm, relevance of synaptic weights, essential link, open faults*

1. Introduction

Feedforward neural networks (NNs), trained with the backpropagation algorithm have been applied successfully in variety of diverse areas such as speech recognition, optical character recognition, control, and medical analysis [1]. The algorithm seeks to minimize the error in the output of NN as compared to a target, or desired response [2]. Although it was thought that NNs are fault tolerant as they consist of parallel processing elements, the existing learning algorithms do not make optimal use of redundant resources. Recently extensive research has proved that NNs are not intrinsically fault tolerant, and the fault tolerance has to be enhanced by adequate scheme [3], [4]. A number of methods have been proposed to enhance the fault tolerance ability of NNs. The influence of learning rate, training time and training with noisy input data on the performance of the NNs under the existence of fault have been studied [3]. In [5] it was found that training on noisy input data also enhance the fault tolerance ability of NNs. The effect of analog noise injection on the synaptic weights during multilayer neural network training on the fault tolerance property was analyzed [6]. A procedures to build fault tolerant NNs by replicating the hidden units are presented [8], [12], and the minimum redundancy required to tolerate all possible single faults is analytically derived [12]. Using error correcting code, a fault tolerant design which can correct an error at the output layer neuron was presented [7]. A learning algorithm that minimizes the difference between faulty and non faulty networks is proposed, and the close relationship between generalization and fault tolerance abilities is discussed [9].

C.T. Chiru et al. found that links that are highly sensitive have large weights [13]. To enhance the fault tolerance, they proposed to coerce weights to have small values during backpropagation algorithm, and additional hidden neurons are added dynamically to the network to ensure that desired performance can be obtained. However it is not known a priori the best limited range for weights and the dynamically addition of neurons may lead a large number of hidden neurons without being sure that the network uses optimally the initial size.

The ability to perform the generalization is one of the most important properties of NNs [16], [17]. In an attempt to improve the generalization capability, a weight smoothing algorithm was proposed [18], this algorithm can be used when there exists some correlations among neighboring data patterns. The pruning techniques also were proposed to enhance the generalization ability of the NNs [11].

In the conventional backpropagation algorithm, weights are modified in order to decrease the error on the learning pattern without any self-built mechanism that makes weights encoding the similar amount of information. This paper proposes a new learning algorithm to enhance the fault tolerance and the generalization abilities of feedforward NNs. The algorithm focuses on the links (weights) that causes an error at the output when they are open faults. These weights are considered to encode much information. The algorithm estimates the relevances of the synaptic weights to the output error in each training cycle, then the weight which gives the maximum relevance is decreased.

Next section describes the neural network architecture and presents the fault model adopted for NNs. The learning algorithm for fault tolerant NNs is presented in Sect. 3, in Sect. 4 we evaluate the proposed algorithm on a character recognition problem by investigating the NNs fault tolerance and the generalization

Manuscript received May 1, 1996.

†The author is with Graduate School of Science and Technology, Chiba University, Chiba-shi, 263 Japan.

††The author is with the Faculty of Engineering, Chiba University, Chiba-shi, 263 Japan.

capabilities. The influence of the proposed algorithm on the maximum relevance and the weights distribution as compared to that of the standard backpropagation algorithm (SBPA) is analyzed in Sect. 5.

2. Network Architecture and Fault Model

In this section, an outline of a target neural network (NN) and fault model are presented.

2.1 Network Architecture

We consider a feedforward NN with single hidden layer. The input layer neurons are fully connected to the hidden layer neurons which are fully connected to the output layer neurons. The neuron model is shown in Fig. 1. The output o_j of the jth neuron is given by

$$o_j = f\left(\sum_{i=0}^{N} w_{ij}u_i\right), \tag{1}$$

where w_{ij} is the synaptic weight on the connection from the ith neuron to the jth neuron, N is the number of neurons that feed the jth neuron (it is equal to the number of neurons in the previous layer), u_i is the output of the ith neuron, the bias w_{0j} is treated as a synaptic weight connected to a fixed input $u_0 = 1$, and f is the activation function given by

$$f(x) = \frac{1}{1+\exp(-x)}. \tag{2}$$

In the classification task, the activation function of the neurons in the output layer can be replaced by threshold activation after training, and the output o_k is classified as follows:

$$\begin{aligned} o_k &= 1 \quad \text{if} \quad \sigma = \sum_{j=0}^{H} w_{jk}u_j > 0, \\ o_k &= 0 \quad \text{if} \quad \sigma = \sum_{j=0}^{H} w_{jk}u_j < 0, \end{aligned} \tag{3}$$

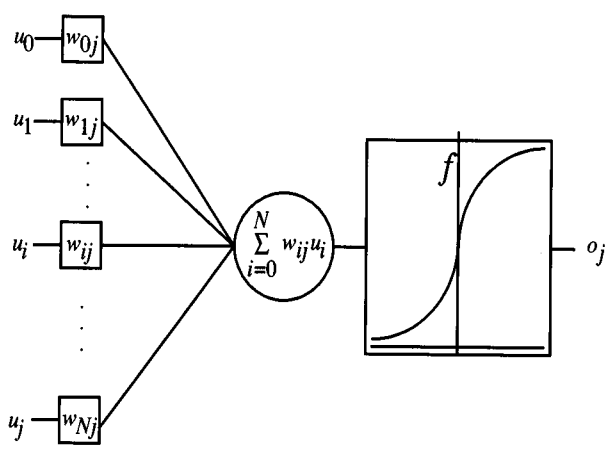


Fig. 1 Neuron model.

where H is the number of neurons in the hidden layer.

The output is considered wrong if it switches from the normal output 1 to 0 or from the normal output 0 to 1, this happens if σ changes its sign.

2.2 Fault Model

A physically plausible type of fault is the loss of connection between two neurons (*open fault*) [8], this relates to the loss of an arc in a directed graph which abstractly represents the topology of NNs [14]. This fault is equivalent to the case when the synaptic weight is set at 0, this fault is also equivalent to the conventional stuck-at-0 type. This fault is assumed in this paper.

Fault tolerance is frequently cited as an important property of NNs [8], however, the loss of single weight is frequently sufficient to completely disrupt a learned function. A connection link is called an *essential link* if its disconnection causes an error at the output [9]. In our simulation, to find the number of essential links, a link is cut (its synaptic weight is set at 0) and all the training patterns are applied to NN. If the network can not recognize all the training patterns, the given link is an essential one.

3. Learning Algorithm for Fault Tolerant Nets

Theoretical background for the proposed learning algorithm is presented first, and the algorithm is presented next.

3.1 Relevance of Synaptic Weights

When a given synaptic weight is stuck-at-faulty value w^f_{ij} (i.e. its value becomes w^f_{ij}), the first manifestation of this fault is observed at the level of the summation of weighted inputs of the neuron j (after the summation operation, it is not possible to know which weight was faulty). The error ε_j at the input summation of the neuron j is given by

$$\varepsilon_j = \sum_{i=0}^{N} w_{ij}u_i - \sum_{i=0}^{N} w^f_{ij}u_i. \tag{4}$$

Let us consider that only the connection between a neuron i_0 and the neuron j to be lost (i.e. only the weight w_{i_0j} is stuck-at-0). The error ε_j in the neuron j at the input summation is given by

$$\varepsilon_j = w_{i_0j}u_{i_0} \tag{5}$$

which is proportional to the weight magnitude. From Eq. (5) we realize that small synaptic weights are not likely to influence the output much, and large weights are likely to produce a wrong output when they are stuck-at-0. Here we are not saying that a large weight do necessarily produce a wrong output when it is stuck-at-0.

The pruning techniques described in [4], [11] estimate the sensitivity of the error function to removal of a neuron; then remove the neuron having least effect in order to enhance the generalization ability. In this paper it is the weight having the maximum relevance which is subjected to a modification (decrease).

In this paper the *relevance* $R(w_{ij})$ of a given weight w_{ij} is defined as the maximum error caused at the primary output by the stuck-at-fault of this weight. It is given by

$$R(w_{ij}) = \max_{p\in P, k\in K} \mid o^p{}_{kw_{ij}} - o^p{}_{kw^f_{ij}} \mid, \tag{6}$$

where $o^p{}_{kw_{ij}}$ is the practical output of the kth neuron in the output layer, $o^p{}_{kw^f_{ij}}$ is the output when the synaptic weight w_{ij} is stuck at a faulty value w^f_{ij}, and $\mid x \mid$ denotes the absolute value of x. The maximum is over the set of all primary output neurons K and the set of all training patterns P. The relevance $R(w_{ij})$ is an indication of the importance of the weight w_{ij} to the network. Note that the output $o^p{}_{kw^f_{ij}}$ is compared to the practical output rather than the theoretical output. using the practical output rather than the theoretical one makes possible using Taylor expansion for easy calculation of the weights relevance as shown later.

The relevance of any weights $R(w_{ij})$ can be evaluated exhaustively by setting w_{ij} to 0 and applying all the training patterns to the NN and evaluating the maximum error at the primary output. However the time for exhaustive evaluation of the maximum relevance can be very long and impractical since it requires a forward propagation of all the training patterns for each and every synaptic weight.

In this paper to avoid the long evaluation time, the relevances are estimated in the training phase using the Taylor expansion of the output around fault-free weights. A.F. Murray et al. perform the Taylor expansion of the output error function in term of the injected synaptic noise in [6]. This allows the authors to make specific predictions regarding fault tolerance of NNs, the nature of the internal representations developed during training and the process of training itself.

When the pth pattern is applied to the primary input, the output $o^p{}_k$ of the kth neuron in the output layer can be expressed as a function of the weight vectors W. Using the Taylor expansion of the output $o^p{}_k(W)$ to the second order, the maximum error caused at the primary output neurons, when w_{ij} is stuck-at-0, is expressed as follows

$$R(w_{ij}) = \max_{p\in P, k\in K} \left| -w_{ij}\frac{\partial o^p{}_k}{\partial w_{ij}}(W) + \frac{{w_{ij}}^2}{2}\frac{\partial^2 o^p{}_k}{\partial {w_{ij}}^2}(W) \right|. \tag{7}$$

The detail of the derivation of the above equation is given in [15].

3.2 Learning Algorithm

The algorithm estimates The relevances of the synaptic weights to the output error in each training cycle of the standard backpropagation using the Taylor expansion of the output around fault-free weights. Then the weight giving the maximum relevance is decreased.

The weight that has the maximum influence (relevance) on the output error in training phase is considered to encode much information. In this paper we propose to decrease the weight producing the highest relevance as follows

$$w_{ij} \rightarrow \frac{w_{ij}}{1 + \lambda R(w_{ij})}, \tag{8}$$

where λ is a positive real constant called *penalty factor*. The factor $1 + \lambda R(w_{ij})$ penalizes the weight that gives the highest relevance (i.e. that has large influence on the output error). Too small λ will have no influence, and large λ will cause too strong penalty and thus will make the training process long. Since decreasing the weight that has a maximum influence on the output may disturb the convergence of the learning process, the penalty factor λ is set to 0 after a fixed time T_λ is over.

The final algorithm is described as follows:

Step 1.
- initialize the weight matrix
- fix the training parameters
- fix the penalty factor λ and T_λ

Step 2.

for each training cycle do
- perform the standard backpropagation weights updating
- calculate the relevances of all weights
- find the weight $w_{i_0 j_0}$ producing the maximum relevance

Step 3.
- if the algorithm converges STOP.
- if the time T_λ is over, set λ to 0 and go to Step 2..
- otherwise decrease $w_{i_0 j_0}$ as indicated in Eq. (8) and go to Step 2..

Since it is the weight which has the highest relevance (effect) is decreased, the proposed algorithm is usually slower than the standard backpropagation algorithm (SBPA). To overcome this shortcoming a maximum time T_λ is fixed after which the penalty factor is set to 0 and the algorithm becomes equivalent to the SBPA.

4. Evaluation of the Proposed Algorithm

In this section we evaluate the proposed learning algorithm and compare it with the SBPA. We show that the proposed algorithm do enhance the fault tolerance and the generalization abilities of NNs. We also present the time overhead required by the proposed algorithm.

In the simulation, a character recognition problem is considered. The characters from A to P presented on 7×7 binary image plane are used to evaluate the fault tolerance and the generalization abilities. Four networks are investigated. The networks have 8, 10, 12, and 14 neurons in the hidden layer respectively. All the networks have 49 neurons in the input layer, and 16 neurons in the output layer. The initial weights are randomly set to values that are uniformly distributed inside the interval $[-1,1]$.

In the experiments, the 16 normal characters and 16 patterns generated by changing 3 randomly selected pixels in each normal character are used to train the networks.

The fault tolerance property, as the ability to function in the presence of faults, is different from the ability to classify non-trained data (generalization ability). The training set is used to assess the fault tolerance and a test set is generated to assess the generalization ability.

4.1 Fault Tolerance

After the training has been finished, the network's tolerance to damage is assessed. As the results depend on the weights from which the training process is initiated [8], [10], 20 experiments were made for each network with different initial weights. Two fault tolerance metric are adopted, the first is the number of essential links (which measures the ability of the NN to tolerate a single fault), the second is the percentage of recognized patterns as function of the percentage of broken links in the network (which measures how bad the network's performance degrades). Table 1 presents the number of essential links (N_{el}) in the networks for different value of λ ($\lambda = 0$ means the SBPA). The results show that N_{el} in networks trained with the proposed algorithm is significantly reduced as compared to N_{el} in the network trained with the SBPA. This means that the proposed algorithm leads to a network that exhibits better graceful degradation. The networks 49-12-12 , and 49-14-16 when trained with the proposed algorithm, becomes, in many cases, complete fault tolerant (i.e $N_{el}=0$, it can tolerate any single open fault). It was found interesting in experiments that all the essential links are hidden-to-output connections.

The presence of more faults is in the network is assessed by cutting a number of randomly selected links, then the patterns of the training set are applied, and the percentage of recognized patterns is assessed. As some links are more significant than others, the process was repeated 200 times for each number of broken links and the results are averaged. This evaluation is done for 20 random start positions (initial weights), and the results are averaged.

The simulation results are presented in Fig. 2 and Fig. 3 for the networks 49-10-16 and 49-14-16 respectively. The results show that the NNs trained with the proposed algorithm (FTNs) do exhibit better graceful degradation than those trained with the SBPA (SBNs). The recognition rate of SBNs degrades faster than the FTNS. The recognition rate is almost the same for $\lambda = 5$ and $\lambda = 10$, and it is better to some extent than the recognition rate for $\lambda = 1$ and $\lambda = 0.5$.

Comparing the recognition rate by 49-10-16 and 49-14-16, it can be realized that the fault tolerance do increase as the number of hidden neurons increases. This result proves that the finding of M.D. Emmerson and R.I. Damper in [8], who stated that " the fault tolerance do not increase as the number of hidden neurons increases," is not a general for all the pattern recognition problems.

Table 1 The average number of essential links N_{el} for different value of λ.

Network	49-8-16	49-10-16	49-12-16	49-14-16
$\lambda = 0.0$	47.8	28.4	18.44	10.2
$\lambda = 0.5$	38.6	14.4	2.6	0.3
$\lambda = 1.0$	37.9	11.4	0.3	0.1
$\lambda = 5.0$	35.8	9.5	0.4	0.0
$\lambda = 10$	34.2	7.1	1.1	0.0

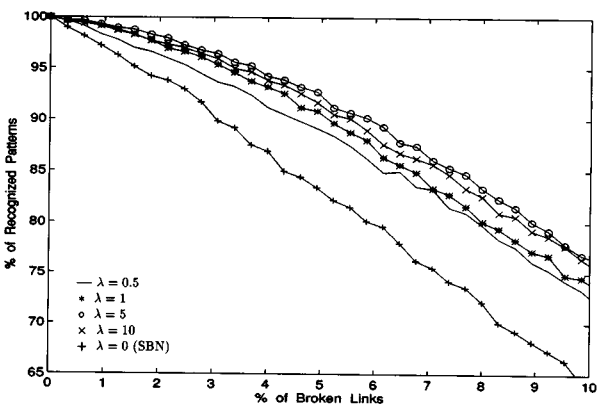


Fig. 2 The percentage of recognized patterns by 49-10-16 networks as function of the % of broken links.

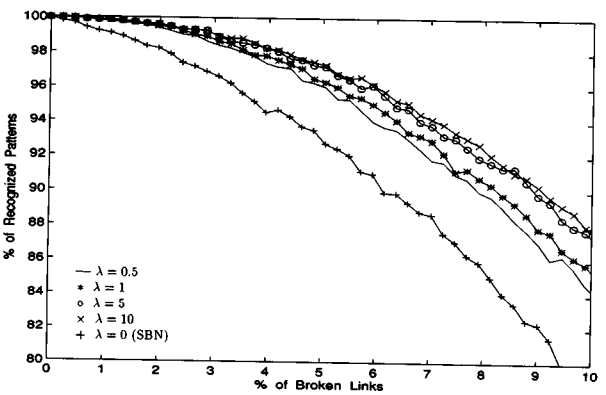


Fig. 3 The percentage of recognized patterns by 49-14-16 networks as function of the % of broken links.

Table 2 The average recognition rate in the networks for different values of λ.

Network	49-8-16	49-10-16	49-12-16	49-14-16
$\lambda = 0.0$	86.65	89.25	90.13	91.21
$\lambda = 0.5$	89.19	91.95	94.37	94.81
$\lambda = 1.0$	89.93	91.68	93.71	94.83
$\lambda = 5.0$	91.23	93.44	93.47	95.69
$\lambda = 10$	92.37	92.18	94.53	95.07

Table 3 The average of time overhead (O_t) required by the proposed learning algorithm.

penalty λ	49-8-16	49-10-16	49-12-16	49-14-16
0.5	41.9	39.2	36.7	31.9
1	59.6	40.3	41.3	34.5
5	62.3	44.8	39.9	38.9
10	64.2	45.6	46.2	39.1

4.2 Generalization Ability

In addition to the fault tolerance property, the ability to perform the generalization is one of the most important properties of NNs[16]. To evaluate the generalization ability, a test set of 16,000 patterns was generated by changing 1, 2, and 3 randomly selected pixels in each normal pattern, (1000 test patterns from each character). The patterns of the test set are applied to the network, and the percentage of recognized patterns is assessed. Examining Table 2, which present the average recognition rates on the test set, it can be seen that the proposed algorithm usually do improve the recognition rate of NNs. The amelioration depends on the value of λ and the network size.

4.3 The Time Overhead

The time overhead O_T required by the proposed learning algorithm is defined as follows:

$$O_T = \frac{t_p - t_o}{t_o}, \tag{9}$$

where t_p is the learning time of the proposed algorithm, and t_o that of SBPA. Table 3 shows the time overhead O_T required by the proposed learning algorithm. It depends on the size of the network, the value of λ and the maximum time T_λ after which the penalty factor λ is set at 0.

During the simulation, the maximum time T_λ was fixed to the average necessary time for the convergence of the networks trained with the SBPA.

5. Analysis

In this section, the influence of the proposed learning algorithm on the maximum relevance and the weights distribution is analyzed.

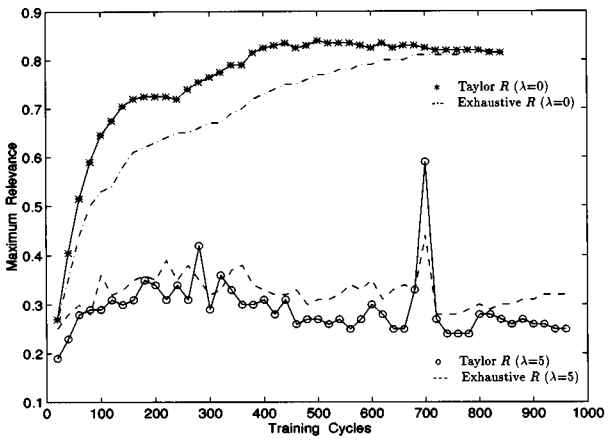


Fig. 4 The maximum relevance in a 49-12-16 network.

5.1 Maximum Relevance

The maximum relevance presents the maximum error caused at the primary output by an open fault. Figure 4 presents the maximum relevance during training phase of 49-12-16 network for both the proposed algorithm $\lambda = 5$ and SBPA ($\lambda = 0$). This graph presents a case where a complete fault tolerant network was obtained ($N_{el} = 0$). The results of exhaustive evaluation are also plotted for reference. Taylor R is the relevance obtained using Eq. (7) and Exhaustive R is the relevance evaluated exhaustively. The figure shows that the maximum relevance (i.e the maximum error at the output) of the proposed method ($\lambda = 5$) is almost constant about 0.3 whereas that of the SBPA ($\lambda = 0$) becomes larger when the number of training cycle increases. This means that the effect of an open fault on the output error in SBN is larger than that in FTN, that is FTN is more fault tolerant. The figure also shows that the estimates of the maximum relevance obtained by Taylor expansion are close the maximum relevance evaluated exhaustively. During the simulation it was found that the weight which has the maximum relevance is always a hidden-to-output weight. This finding allowed us to minimize the time overhead by estimating only the relevance of hidden-to-output weights.

5.2 Distribution of Weight's Magnitude

We assumed that the large synaptic weights are probably on the essential links. Then it is interesting to compare the distribution of weights after training has been finished in both the FTNs and SBNs. Through the experiments it was found that the essential links are always hidden-to-output connections. Figure 5 shows the distribution of the hidden-to-output weights (w_{jk}) for fifty 49-8-12 networks. We obtain a bimodal distribution for the FTNs and a simple normal distribution for the SBNs. The distribution shows that FTNs have less number of small $|w_{jk}|$ (non informative weights) and

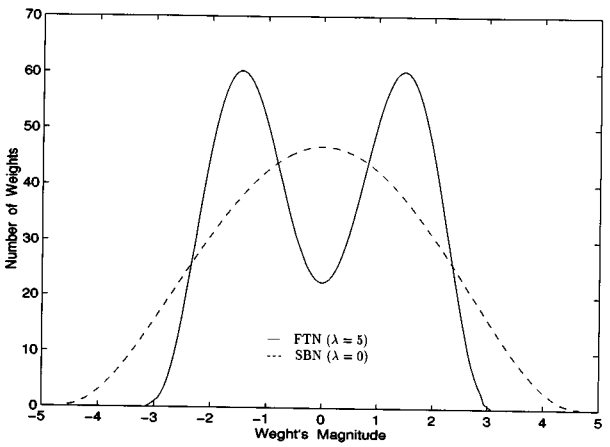


Fig. 5 The distribution of the hidden-to-output weights.

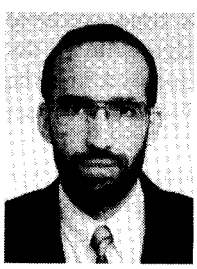
less number of large $|w_{jk}|$ (probably critical weights) than SBN. It shows also that the proposed algorithm generates an hidden-to-output weight distribution that has smaller variance than that generated by SBPA.

6. Conclusion

A new learning algorithm for fault tolerant neural networks was presented. This algorithm uses the Taylor expansion of the output around fault-free weights to estimate the relevances of weights to the output error, then decrease the weight producing the maximum relevance. This technique provides an effective design option to build fault tolerant networks. The simulation results indicate that the network trained with the proposed algorithm do exhibit better fault tolerance than the standard backpropagation network and that it is possible to obtain a complete fault tolerant network (i.e the number of essential links is zero). The results also indicated that the networks generalization ability was improved using the proposed algorithm. Further, using the Taylor expansion, the relevance was closely estimated. The analysis of weights distributions shows that the proposed algorithm generates a hidden-to-output weight distribution that has small variance and less number of weights having small absolute values than that generated by the standard backpropagation algorithm.

References

[1] S. Hayken, "Neural Networks, A Comprehensive Foundation," IEEE press, 1994.
[2] R.C. Frye, E.A. Rietman, and C.C. Wong, "Back-propagation learning and nonidealities in analog neural network hardware," IEEE Trans. Neural Networks, vol.2, no.1, pp.110–117, Jan. 1991.
[3] J. Nijhuis, B. Hofflinger, A. Schaik, and L. Spaanenburg, "Limits to fault-tolerance of a feedforward neural network with learning," Digest FTCS, pp.228–235, June 1990.
[4] B.E. Segee and M.J. Carter, "Fault tolerance of pruned multilayer networks," Digest IJCNN, pp.II-447–452, 1991.
[5] J.I. Minnix, "Fault tolerance of backpropagation neural network trained on noisy inputs," Digest IJCNN, pp.I-847–852, 1992.
[6] A.F. Murray and P.J. Edwards, "Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training," IEEE Trans. Neural Networks, vol.5, no.5, pp.792–802, 1994.
[7] H. Ito and T. Yagi, "Fault tolerant design using error correcting code for multilayer neural networks," IEEE Int. Workshop on Defect and Fault Tolerance in VLSI systems, pp.177–184, 1994.
[8] M.D. Emmerson and R.I. Damper, "Determining and improving the fault tolerance of multilayer perceptions in a pattern-recognition application," IEEE Trans. Neural Networks, vol.4, no.5, pp.788–793, 1993.
[9] Y. Tan and T. Nanya, "Fault-tolerant back-propagation model and its generalization ability," Digest IJCNN, pp.2516–2519, 1993.
[10] G.P. Drago and S. Ridella, "Statistically controlled activation weight initialization (SCAWI)," IEEE Trans. Neural Networks, vol.3, no.4, pp.627–631, July 1992.
[11] R. Reed, "Pruning algorithms—A survey," IEEE Trans. Neural Networks, vol.4, no.5, pp.740–747, Sept. 1993.
[12] D.S. Phatak and I. Koren, "Complete and partial fault tolerance of feedforward neural nets," IEEE Trans. Neural Networks, vol.6, no.2, pp.446–456, March 1995.
[13] C.T. Chiu, K. Mehrotra, C.K. Mohan, and S. Ranka, "Training techniques to obtain fault-tolerant neural networks," Digest FTCS, pp.360–369, June 1994.
[14] G. Bolt, "Fault models for artificial neural networks," Digest IJCNN, pp.1373–1378, 1991.
[15] N.C. Hammadi and H. Ito, "A learning algorithm for fault tolerant feedforward neural networks," IEICE Technical Report, FTS95-78, Feb. 1996.
[16] S.B. Holden and P.J.W. Rayner, "Generalization and PAC learning: Some new results for the class of generalized single-layer networks," IEEE Trans. Neural Networks, vol.6, no.2, pp.468–480, March 1995.
[17] L. Holmstrom and P. Koistinen, "Using additive noise in back-propagation training," IEEE Trans. Neural Networks, vol.3, no.1, pp.24–38, Jan. 1992.
[18] J.S.N. Jean and J. Wang, "Weight smoothing to improve network generalization," IEEE Trans. Neural Networks, vol.5, no.5, pp.468–480, Sept. 1995.



Nait Charif Hammadi was born in Tinghir, Warzazat, Morocco, on December 25, 1965. He received the Ingenieur d'Etat diploma in electronics and control, from Ecole Hassania des Traveaux Publics, Casablanca, Morocco, in 1990. In 1991, he joined the Ecole Superieure de Technologie, Mohamed I University, as assistant lecturer. Currently, he is working toward his Ph.D. in Information and Computer Sciences, Graduate School of Science and Technology, Chiba university, Japan. His research interests include neural networks and fault tolerant systems. He is member of IEEE.

Hideo Ito was born in Chiba, Japan, on June 1, 1946. He received the B.E. degree from Chiba University in 1969, and the D.E. degree from Tokyo Institute of Technology in 1984. He joined Nippon Electric Co. Ltd. in 1969, and Kisarazu Technical College in 1971. Since 1973 he has been a member of the Chiba University. He is currently a Professor of Department of Information and Computer Sciences. His research interests include easily testable design, test generation, VLSI architecture, fault-tolerant design of parallel & distributed systems, and defect-tolerant VLSI design. He is a member of the Information Processing Society of Japan and the IEEE Computer Society.