Empirical Study Of FFANNs Tolerance To Weight Stuck At Zero Fault

Amit Prakash Singh, Pravin Chandra, Chandra Sekhar Rai

University School of Information Technology
Guru Gobind Singh Indraprastha University, Delhi, India

***Abstract*— Fault tolerance property of artificial neural networks has been investigated with reference to the hardware model of artificial neural networks. Weight fault is an important link, which causes breakup between two nodes. In this paper weight fault has been explained. Experiments have been performed for Weight-*stuck-0* fault. Effect of weight-stuck-0 fault on trained network has been analyzed in this paper. The obtained results suggest that networks are not fault tolerant to this type of fault.**

Keywords- artificial neural network,weight Fault, fault tolerance

I. INTRODUCTION

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the biological nervous system, i.e., the human brain [2]. The key element of this paradigm is the novel structure of the information processing system. It is composed of large number of highly interconnected parallel processing elements (neurons) working together to solve specific problems. An ANN is configured for specific applications, such as pattern recognition or data classification, through learning process.

Artificial neural network have the potential for parallel processing due to the implementation on Application Specific Integrated Circuit (ASIC) [4] or Field Programmable Gate Array (FPGA) [5][6][7]. The input-output function realized by neural network is determined by the value of its weights.

In the case of biological neural network, tolerance to loss of neurons has high priority, since a graceful degradation of performance is very important for survival of the organism. Fault tolerance measures the capacity of neural network to perform the desired task under given fault condition. It also maintains their computing ability when a part of the network is damaged or removed. In [9], the study of fault tolerant properties of the neurons has been reported for partial fault tolerance by replication and training and the assertion is that Triple Modular Replication (TMR) leads to a fault tolerant network. This is a one of the popular technique in digital system.

Fault tolerances of ANN have been studied in [1][18]. Fault tolerance of ANN may be characterized/categorized on the following aspects:

(i) Weight error: Weight stuck at zero
(ii) Neuron error: Node stuck at zero
(iii) Input pattern errors: injecting noise during the training phase.

The focus of this paper is on effect of weight fault. Experiments have been performed on weight stuck at zero faults on the trained network. Detailed experimentation has been explained in section VI. An Architecture of feedforward artificial neural network (FFANN) has been chosen for the experimentation purpose because they are among the most popular types.

Hardware model(s) for artificial neural network(s) has (have) been widely implemented by various researchers for applications like image processing and controller-based applications. In recent years, many approaches have been proposed for the implementation of different types of neural networks, such as multilayer perceptron [8], Boltzmann machine [13] and other hardware devices [11].

The analysis of fault tolerance of a network normally requires study of weight fault, node fault and external faults [33]. *Stuck-at* model is a popular technique to study effect of fault on a given network [32], where a faulty gate delivers a constant logic one or logic zero at its output, or acts as if one of its inputs is stuck at a fixed logic value. Neural networks process analog function values, and thus the range of possible faults may be even larger. *Weight stuck at zero* faults has been chosen for the experimentation purpose, in order to make fault analysis manageable.

The emphasis of the fault tolerance investigation of ANNs has been focused on the demonstration of non-fault tolerant behavior of these networks and/or the design of paradigm for making a network fault tolerant to specific faults. A complete modeling of neural faults is still lacking. This paper aims to present an effect of weight fault specifically stuck at zero faults on a trained network.

In this paper, Section II discusses the architecture of FFANN. Section III discusses related work; Section IV discusses fault model and metrics for measurement of faults. Section V describes weight fault. Section VI discusses the experiments and the obtained results for the weight-*stuck-0* fault while conclusion is presented in Section VII.

II. ARCHITECTURE OF FFANN

The architecture of the proposed fault diagnosis neural network is illustrated in Figure 1. A feed forward artificial neural network has been chosen for the experiment purpose. 30 nos. of network have been trained for the purpose of experiment. A network has single hidden layer with two input nodes and one output nodes in the experiments conducted. Output of a neuron is:

$$y = f(\sum_{i=1}^{N} x_i w_i) \quad (1)$$

Where, x_i: input vector applied to network
Wi: weight applied to each input
f(a): activation function for a hidden node

And activation function is defined as

$$\tan sig = f(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \quad (2)$$

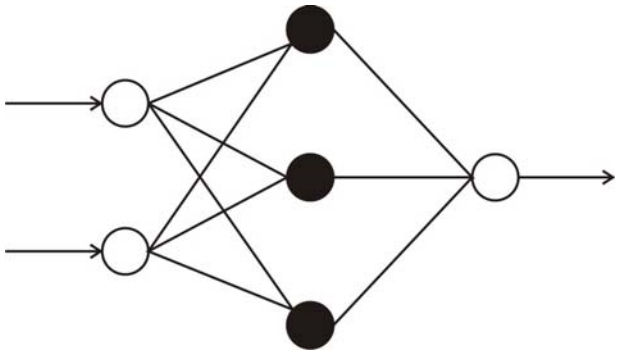


Figure 1: Schematic Architecture of FFANN with one hidden layer of nonlinear nodes.

A hyperbolic tangent sigmoid transfer function has been chosen as an activation function for hidden node and a linear transfer function has been chosen as an activation function for output node.

III. RELATED WORK

With the widespread usage of the chip-based device of the ANN as controller [11], it has become imperative to study the behavior of these circuits under various faults, i.e., the study of their fault tolerance behavior must be undertaken. The available literature on the fault-tolerance behavior of feedforward ANNs may be summarized as:

1. Demonstration of non-fault-tolerance to specific faults [9] [14].
2. Regularization during training [15].
3. Enhancement of fault tolerance by design of algorithms for embedding fault-tolerance into the network, during training [16] [19].
4. Redesigning the network architecture (after training) by replication of nodes and their associated weights and usage of majority voting [3] [17].

Piuri [14] asserts that the network can not be considered to be intrinsically fault tolerant. Edwards and Murray [15], use the regularization effect of weight noise to design a fault tolerant network. Chin et. al. [19] demonstrate a training algorithm that uses weight value restriction (and addition of additional nodes), fault injection during training and network pruning to achieve a fault tolerant network, while [3] and [12] redesign the trained network to achieve a fault tolerant network. Phatak and Koren [17] devised measures to quantify the fault tolerance as a function of redundancy. Bolt *et. al*. [20] indicated that the network trained by backpropagation algorithm seldom distribute information to connection uniformly. Due to this information few connection are key components, whose failure will cause great loss to the networks. A method to improve the fault tolerance of backpropagation networks is presented in [30], which restrained the magnitudes of the connections during training process. Hammadi and Ito [16] demonstrate a training algorithm that reduces the relevance of weight. In [16], relevance of weight in each training epoch was estimated, and then decreases the magnitude of weight

IV. FAULT MODEL AND METRICS

Three types of fault model exist in neural networks system [12]. Fault models are categorized as follows:

1) Weight faults
2) Input faults
3) Node faults

Missing link of interconnection between two nodes is called weight fault. The weight and node faults are often modeled as stuck-at-0 and most often occur during a memory disappearance or a link disconnection in VLSI. Categorization of weight faults is explained in section 5.

Any incorrectness in the input to the adaptive machine is defined as an input fault. These faults occur due to external disturbance or noise. Mainly these types of fault affect input vector of the machine.

Node fault is a similar type of fault as weight fault. Node faults are categorized in two types of node faults, namely hidden node faults and output node faults. Three types of node faults happen in node faults. Node fault categorized as follows:

1. Node stuck at zero
2. Node stuck at one
3. white noise in node

In this paper we consider only weight stuck at zero. This fault corresponds to linkage breaking in the hardware, and thus is one of the most important hardware faults.

To measure the effect of faults/errors on the network output enumerated above, the following types of error/fault/parameter sensitivity measures may be defined:

MSE, MAPE and Other Global Measures:

The mean squared error (MSE) and the mean absolute percentage error (MAPE) should be used to measure the effect of all types of faults if the output of the FFANN is real. The percentage of misclassification is suggested as a measure of fault/error, for classification problem

V. WEIGHT FAULTS

The removal of the interconnected weights in a network and the occurrence of stuck-at faults in neurons are of two types that can serve as a test bed for the robustness of neural networks. The robustness of a backpropagation trained multilayer network to remove weights to/from the hidden layer and the influence of redundancy in the form of excess hidden neurons has been investigated in [14]. The effect of "Stuck-at-0" and "stuck-at-1" neurons on the solutions found in recurrent optimization networks is investigated in [26].

These are the faults/errors that affect the weights of the network. Following types of faults / errors are defined:

(a) Weight stuck at zero (WSZ): This fault corresponds to an open fault or connection breakage between two nodes.

(b) Weight stuck at maximum/minimum (WSMa/ WSMi): Weight stuck at a value of $\pm|W|_{max}$, where $|W|_{max}$ is the maximum magnitude weight in the system. A -ve weight will be pushed to $-|W|_{max}$ while a +ve weight will be pushed towards $+|W|_{max}$ and vice-versa.

This allows us to model weight faults at substantially large values, which may or may not lead to node hard faults of the type NSZ or NSO depending on the weight interaction with the other weights leading to the same node as the faulted weight.

(c) White noise in weights (WNW): The presence of white noise (zero mean gaussian with finite variance) may be taken as a reflection of thermal noise or circuit degradation. This noise is different from node output noise as it is not correlated in weights leading from the same node.

Due to paucity of space, we restrict our attention to stuck at zero fault only.

VI. EXPERIMENTS AND RESULTS

A small experiment was conducted to demonstrate the applicability of WSZ fault on a trained neural network.

30 nos. of networks were trained for the following function approximation tasks [21]

Fn1: $y = \exp(x_1 * \sin(\pi * x_2))$; x1,x2 uniform in [-1,1]

Fn2:

$$y = 1.3356[1.5(1-x_1) + \exp(2x_1 - 1)\sin(3\pi(x_1 - 0.6)^2) + \exp(3(x_2 - 0.5))\sin(4\pi(x_2 - 0.9)^2)]$$

x_1,x_2 uniform in [0,1]

The data set for ANN are generated by uniform sampling of the domain of definition of the functions.

The network consists of two input, one hidden layer and one output node (Figure 1). The detail of the architecture used is summarized in Table 1. The architecture was identified by exploratory experiments where the size of the hidden layer was varied from 5 to 30 (that is, the number of nodes in the hidden layer were varied from 5 to 30 in steps of 5) and the architecture that give the minimum error on training was used. All the hidden nodes use tangent hyperbolic activation function while the output nodes are linear.

Table 1: Architecture of network used

Sr. No.	Function	Inputs	Hidden nodes	Output nodes	No. of weight
1.	Fn1	2	15	1	61
2.	Fn2	2	10	1	41

The resilient propagation (RPROP) [22] algorithm as implemented in MATLAB 7.2 Neural Network toolbox is used with the default learning rate and momentum constant. For training the network 200 random samples were generated from the input domain of the functions for training purposes. 5000 epochs of training was conducted for each problem. 30 nos. of networks has been trained with the above procedure.

Table 2 provides the summary statistics for the network chosen. From the value obtained we may infer that these networks do not show very good fault tolerance behavior for the WSZ. Though, some of the weights do not affect the network computation, as under these fault, both MSE and MAPE values for some of the weights is zero. This corresponds to weights that are not utilized in computation and can be easily pruned.

Table 2: Weight Stuck at zero summary data

	MINMAX		MINMEAN	
Fault (Metric)	Fn1 (18)	Fn2 (11)	Fn1 (30)	Fn2 (21)
MSE				
MIN	0	0	0.000335007	0
MAX	0.461826	4.17175	0.598987	5.59626
MEAN	0.0984594	0.843973	0.096724	0.653498
MEDIAN	0.049704	0.350913	0.0693296	0.321483
STD	0.11113	1.15733	0.115438	1.02684
(MAPE)				
MIN	0	0	0.176943	0
MAX	62.0969	162.796	66.3062	198.611
MEAN	20.3705	45.7728	19.9859	40.9889

MEDIAN	16.688	38.4144	14.7482	28.8604
STD	17.0994	48.7023	16.5305	45.0479

Figures 2 and 3 represents the behavior of all 30 networks for the average MSE and maximum MSE for (any) single WSZ fault for Fn1 and Fn2 respectively. From the figures, we may infer that, the value of the error metric is not the same for each of the 30 networks. That is, since the networks initially differ only in the choice of initial weights, thus we may conclude that initial weight choice has an important role to play in the fault tolerance behavior of FFANNs and needs to be further investigated.

We choose the network (out to 30), with the minimum maximum MSE (MINMAX) and minimum mean MSE (MINMEAN), for further analysis for each of the two tasks; that is, see network with the best fault tolerance behavior is chosen for further analysis.

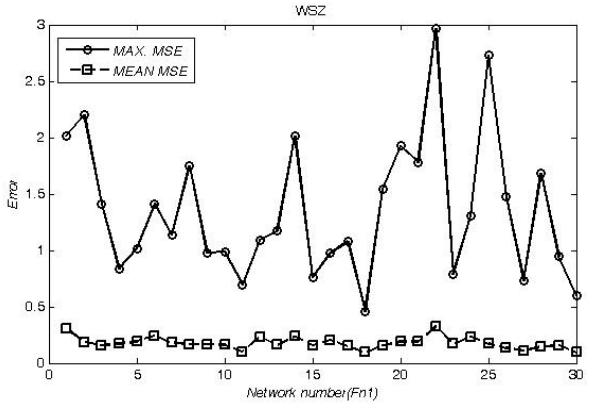


Figure 2: Behavior of 30 networks for Fn1.

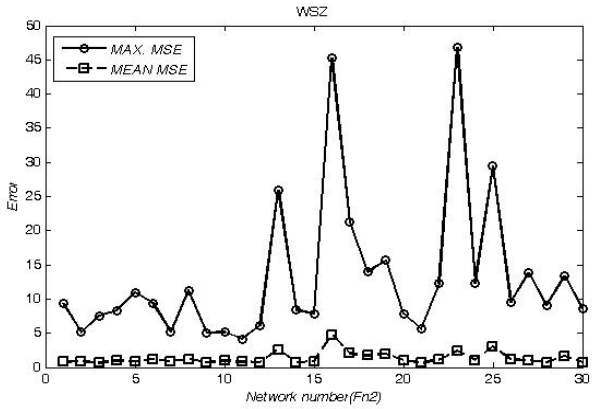


Figure 3: Behavior of 30 networks for Fn2.

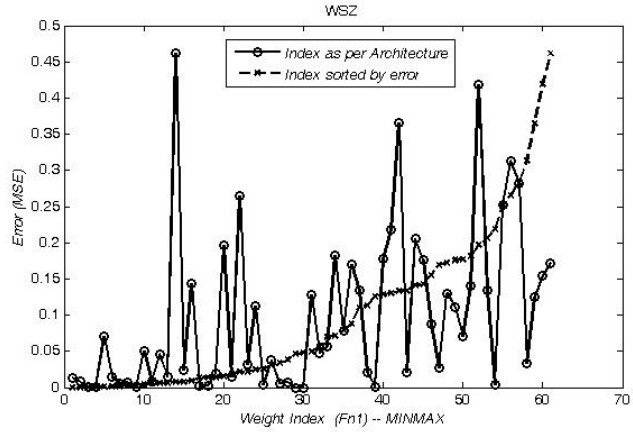


Figure 4: Weight distribution (MINMAX) for network 18 corresponding to Fn1.

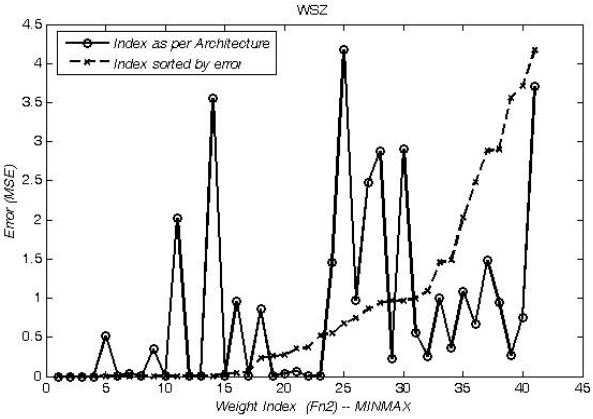
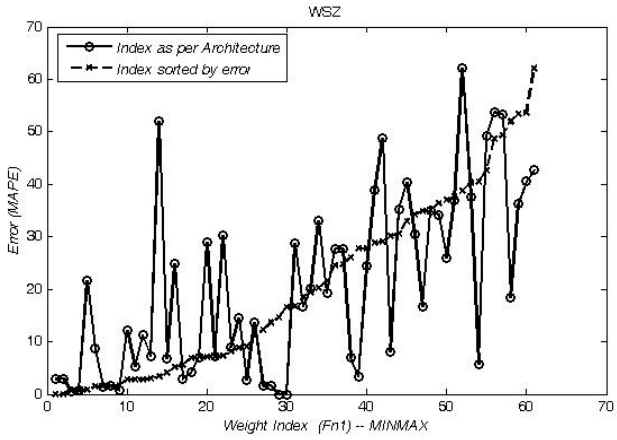


Figure 5: Weight distribution (MINMAX) for network 11 corresponding to Fn2.



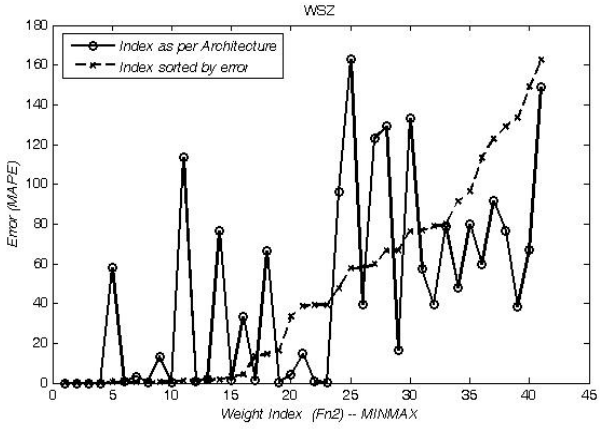


Figure 6: Weight distribution (MINMAX) for network 30 corresponding to Fn1.

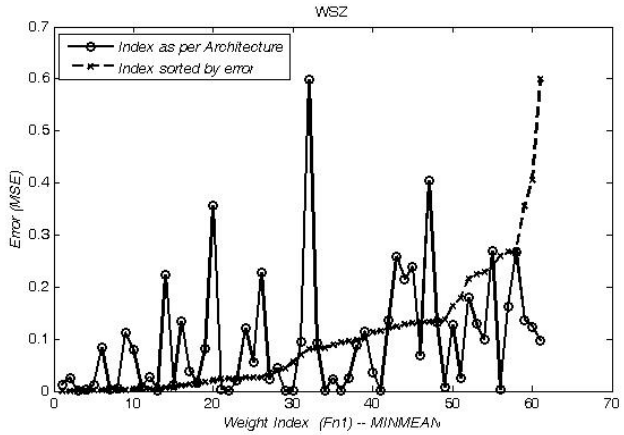


Figure 7: Weight distribution (MINMAX) for network 21 corresponding to Fn2.

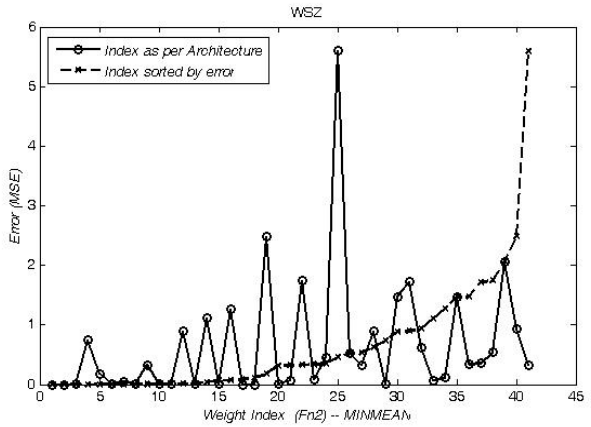


Figure 8: Weight distribution (MINMEAN) for network 18 corresponding to Fn1.

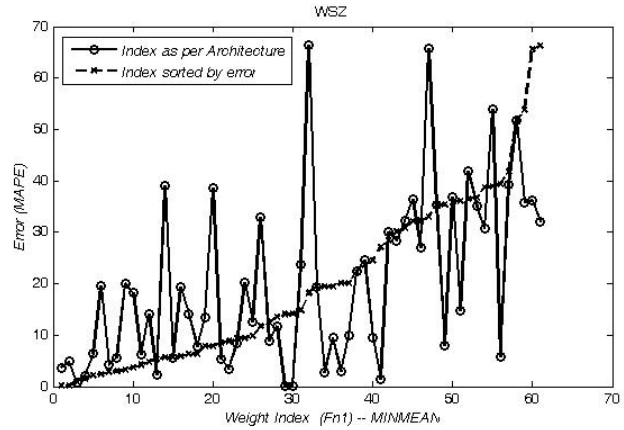


Figure 9: Weight distribution (MINMEAN) for network 11 corresponding to Fn2.

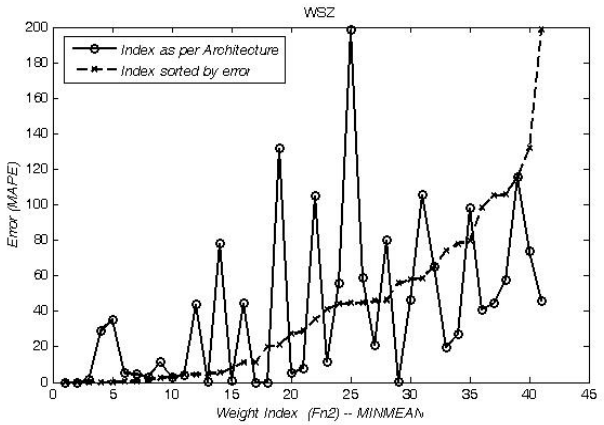


Figure 10: : Weight distribution (MINMEAN) for network 30 corresponding to Fn1.

Figure 11: Weight distribution (MINMEAN) for network 21 corresponding to Fn2.

From figure 4-7, we see that faults in some weights is tolerated (that is, the induced error under fault is small), but the figures also show that faults in some weights are critical (that is, the faults in these weights lead to large computational error in the network output), for the MINMAX metric, a similar behavior is seen in figure 8-11 for the MINMEAN metric.

From the results obtained in Table 2, it is apparent that these networks trained using the RPROP[22] algorithm can not be called fault tolerant to the faults reported.

VII. CONCLUSIONS

This paper has presented empirical results on weight stuck-at-zero faults for sigmoidal FFANNs. From the obtained results we may conclude:

1. Some weights are redundant and can be pruned (fault in these lead to no change in network output).
2. Partial fault tolerance is present in these networks (as faults in some weights lead to small change in output of these networks).
3. Some weights are critical for network computation and faults in these are not well tolerated.

In our opinion, the next step in the analysis of these networks is to device a mechanism that distributes the computational importance of the critical weights through realignment of weight or by addition of new nodes (hidden) and corresponding weights. Moreover, the effect of initial weights on the fault tolerance behavior of FFANN to weight stuck at zero faults needs to be further investigated.

REFERENCES

[1]. F. M. Dias and A. Antunes, "Fault Tolerance of Artificial Neural Networks: an Open Discussion for a Global Model," *International Journal of Circuits, Systems and Signal Processing*, Naun, July, 2008.

[2]. R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[3]. F. M. Dias and A. Antunes, "Fault Tolerance Improvement through architecture change in Artificial Neural Networks," *Engineering Applications of Artificial Intelligence*, 2007.

[4]. S. Satyanarayana, Y. P. Tsividis, and H. P. Graf, "A Reconfigurable VLSI Neural Network," *IEEE Journal of Solid State Circuits*, vol. 27, no. 1, January 1992.

[5]. M. A. Cavuslu, C. Karakuzu, and S. Sahin, "Neural Network Hardware Implementation using FPGA," *Neural Information Processing*, Lecture Notes in Computer Science, Berlin Germany: Springer, 2006, vol. 4234.

[6]. R. Raeisi and A. Kabir, "Implementation of Artificial Neural Network on FPGA," *American Society for Engineering Education*, Illinois-Indiana and North Central Joint Section Conference , April, 2006

[7]. S. Sahin, Y. Becerikli, and S. Yazici, "Neural Network Implementation in Hardware Using FPGAs," *Neural Information Processing*, Lecture notes in Computer Science, Berlin Germany: Springer, 2006, vol. 4234.

[8]. E. M. Ortigosa, A.Canas, E. Ros, P. M. Ortigosa, S. Mota, and J. Diaz, "Hardware description of multi-layer perceptrons with different abstraction levels," *Microprocessors and Microsystems,* vol. 30, pp. 435-444, 2006.

[9]. E.B. Tchernev, R. G. Mulvaney, and D.S. Phatak, "Investigating the Fault Tolerance of Neural Networks," *Neural Computation*, vol. 17, no. 7, pp. 1646-1664, July 2005.

[10]. P. Chandra and Y. Singh, "Feedforward Sigmoidal Networks-Equicontinuity and Fault-Tolerance Properties," IEEE *Transaction on Neural Networks*, vol. 15, no. 6, November 2004.

[11]. F. M. Dias, A. Antunes, and A. Mota, "Artificial Neural Networks: a Review of Commercial Hardware," *Engineering Applications of Artificial Intelligence*, IFAC, vol. 17(8), pp. 945-952, 2004.

[12]. P. Chandra and Y. Singh, "Fault Tolerance of Feedforward Artificial Neural Networks - A Framework of Study," *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 489-494, July 2003.

[13]. M. Skubiszewski, "An Exact Hardware Implementation of the Boltzmann Machine," *Proceedings of the fourth IEEE Symposium on Parallel and Distributed Processing,* pp. 107-110, December 1992**.**

[14]. V. Piuri, "Analysis of Fault Tolerance in Artificial Neural Networks," Journal *of Parallel and Distributed Computing*, pp. 18-48, 2001.

[15]. P. J. Edwards and A. F. Murray, "Fault Tolerance via Weight Noise in Analog VLSI Implementations of MLP's – A Case study with EPSILON," *IEEE Transaction on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 45, no. 9, September 1998.

[16]. N. C. Hammadi and H. Ito, "A Learning Algorithm for Fault Tolerant Feedforward Neural Networks," *IEICE Trans. Information and Systems*, vol. E80-D, no.1, pp.21-27, 1997.

[17]. D.S. Phatak and I. Koren, "Complete and Partial Fault Tolerance of Feedforward Neural Nets," *IEEE Transaction on Neural Networks*, vol. 6, no. 2, pp. 446-456, March, 1995.

[18]. C. Alippi, V. Piuri, and M. Sami, "Sensitivity to Errors in Artificial Neural Networks: A Behavioral Approach," *IEEE Transaction on Circuits and Systems-I: Fundamental Theory and Applications*, vol. 42, no. 6, June 1995.

[19]. C. T. Chiu, K. Mehrotra, C.K. Mohan, and S. Ranka,, "Training Techniques to obtain fault-tolerant neural network," 24th International Symposium on Fault-Tolerant Computing, pp360-369, June 1994.

[20]. G. Bolt, "Investigating Fault Tolerance in Artificial Neural Networks," University of York, Department of Computer Science, Technical Report YCS 154, Heslington, York, England, 1991.

[21]. V. Cherkassky, "Comparison of Adaptive methods for function estimation from samples," *IEEE Transaction on Neural Networks,* vol. 7, no. 4, July 1996.

[22]. M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, pp. 586-591, San Francisco, 1993

[23]. P. Ferreira, P. Ribeiro, A. Antunes, and F. M. Dias, "A high bit resolution FPGA implementation of a ANN with a new algorithm for the activation function," *Neurocomputing*, vol. 71, issues 1-3, pp. 71-73, December 2007.

[24]. F. Piazza, A. Uncini, and M. Zenobi, "Neural networks with digital LUT activation functions," Proceeding of IEEE Joint Conference on Neural Networks, vol. 2, pp. 1401-1404, 1993.

[25]. D. Kincaid and W. Cheney, "*Numerical Analysis: Mathematics of Scientific Computing*," Thomson Learning, 2001.

[26]. Y. F. Wang, X. Q. Zeng, and L. X. Han,"Sensitivity of Madalines to input and weight perturbations," *IEEE Proceeding of the Second International Conference on Machine Learning and Cybernetics*, Xian 2-5 November 2003.

[27]. H. Takenaga, S. Abe, Masao Takatoo, M. Kayama, T. Kitamura, and Y. Okuyama, "Optimal Input Selection of Neural Networks by Sensitivity Analysis and its application to Image Recognition*", IAPR workshop on Machine Vision Applications*, Tokyo, pp. 117-120, Nov 1990.

[28]. J. J. Montano and A. Palmer, "Numeric Sensitivity analysis applied to feedforward neural networks", *Neural computation & Application*, pp. 119-125, Vol. 12, 2003

[29]. J.M. Zurada, A. Malinowski, and I. Cloete, "Sensitivity analysis for minimization of input data dimension for feedforward neural network", ISCAS'94, vol. 6, pp. 447-450,1994

[30]. C.T. Chiru, K. Mehortra, C. K. Mohan, and S. Ranka, "Training techniques to obtain fault-tolerant neural networks", *International Symposium on Fault-Tolerant Computing*, pp. 360-369, 1994.

[31]. N. Wei, S. Yang, and S. Tong , "A modified learning algorithm for improving the fault tolerance of BP networks", IEEE International joint conference on Neural Networks, Washington, DC, vol. 1, pp. 247-252, 1996.

[32]. A. D. Friedman and P. R. Memon, "Fault Detection in Digital Circuits", Prentice-Hall, Englewood Cliffs, NJ 1971.

[33]. Amit Prakash Singh, Pravin Chandra, and Chandra Shekhar Rai, "Fault Models for Neural Hardware", IEEE First International Conference on Advances in System Testing and Validation Lifecycle (VALID 2009), held during September 20-25, 2009 in Porto, Portugal.