Fault Tolerance of Feedforward Artificial Neural Networks – A Framework of Study

Pravin Chandra
School of Information Technology
G.G.S. Indraprastha University
Kashmere Gate, Delhi 110006 (INDIA)
Email: pc_ipu@yahoo.com, pchandra@ipu.edu

Yogesh Singh
School of Information Technology
G.G.S. Indraprastha University
Kashmere Gate, Delhi 110006 (INDIA)
Email: ys66@rediffmail.com, ys@ipu.edu

Abstract— **Feedforward Artificial Neural Networks (FFANN's) are a realization of the supervised learning paradigm. With the availability of hardware implementation of these networks, it has become desirable to measure their fault–tolerance to structural and environmental faults as well as tolerance to noise in the system variables. In this paper, the learning system model is used to describe a framework in which these studies can be conducted. Fault models are describes and error measures suggested. The relation between fault–tolerance and the generalization capabilities of the network is conjectured and the relevance of regularization scheme to fault tolerance property discussed. The available literature on fault–tolerance of neural networks is briefly summarized in the proposed framework. Areas for further exploration are identified.**

I. Introduction

Artificial Neural Networks (ANN's) have been established as an alternative paradigm of computation *vis-à-vis* the concept of programmed computation [1] in which (usually procedural) algorithms are designed and sequentially implemented. ANN's have been inspired by the biological neural networks (BNN's), though the current models of BNN's cannot be called a realistic model of the BNN's by any stretch of imagination. BNN's are characterized by a computational structure that is fault–tolerant to a high degree. Fault–tolerance of BNN's is characterized by:

1) Tolerance to imprecise input (noisy input or incomplete input),
2) Redundancy in the structure that allows the BNN's to cope with the loss of a few neurons and/or neural connections without impairing the computational process,
3) Capability to relearn by growth of new neuron(s) and/or neural connections and/or retraining of the existing neural architecture.

Using the analogy to the BNN's, it is generally assumed that ANN's also possess an inherent fault–tolerant architecture [2], though evidence to the contrary has been presented. It is generally expected that ANN's have the potential to be inherently fault–tolerant, when implemented in the hardware form. [1]–[5] With the availability of ANN hardware, the study of fault–tolerance behaviour of these ANN's has become pertinent. Moreover, the fault–tolerance of ANN's may have a bearing on the generalization capability of these networks, indicating that a more fault–tolerant network will have better generalization capability.

Feedforward ANN's (FFANN's) have been one of the most common architectures for the solution of learning tasks. FFANN's are a realization of the supervised learning paradigm, wherein the parameters of the network are adjusted to minimize the deviations from the expected values of the output (given a specified input).

In this work, we focus on feedforward networks only, recurrent networks are not investigated. In this paper, we model the FFANN's as a learning system. This allows the construction of a framework within which the investigations of fault–tolerance properties of FFANN's can be conducted. We propose fault models and fault/error measures for usage in the study of fault–tolerance property. Existing results on the fault–tolerance features of FFANN's are briefly summarized within the proposed framework. The interplay between fault–tolerance and generalization property is discussed. The framework allows us to propose areas of further investigation. In the current work, the phrase "network", "net", "feedforward nets" etc. are used as a synonym of feedforward artificial neural networks.

II. A Learning System Model of FFANN's

Learning process, for the supervised learning paradigm, may be defined as the process of establishing an unknown input(s) to output(s) functional dependency using a finite number of observation samples or exemplars of pairs of input(s) and associated output(s) [6]. A system that has the capability to learn from examples has three components (See fig. 1).

The input generator produces random vectors $\mathbf{x} \in R^d$, where d is the dimension of the input space, drawn independently from a fixed probability density $p(\mathbf{x})$. In the experimental setting the vectors $\mathbf{x}$ may be contaminated by generation as well as measurement errors, while in the observational setting (where only the inputs from the system sensors are measured) may be contaminated with measurement errors.

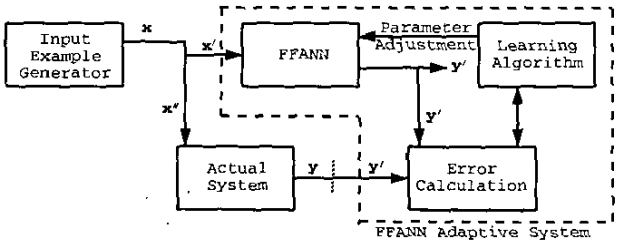

Fig. 1. Learning system model for FFANN's

0-7803-7898-9/03/$17.00 ©2003 IEEE

The actual system produces an output value $\mathbf{y}$ for every input $\mathbf{x}$, according to the fixed conditional density $p(\mathbf{y}|\mathbf{x})$, which is unknown. In the observational setting the actual input to the system will have all the information required from the inputs or $\mathbf{x}'' = \mathbf{x}$. But in the experimental setting, where we may not have the real system but a (prototype) model of the same, the input to this "real" system may be missing some component in the input vector ($\mathbf{x}'' \neq \mathbf{x}$ and $dimension(x'') < dimension(x)$) as well as being contaminated by measurement error. This may also be interpreted as – *real systems generally do not have random outputs; however, they often have unmeasured inputs.* Statistically, the effect of these changing unobserved inputs on the outputs of the system can be characterized as random and represented as a probability distribution. This allows the modeling of all types of regressions including the specific case of deterministic systems of the type $\mathbf{y} = \mathbf{f}(\mathbf{x})$. [7]

The FFANN adaptive machine can be interpreted as a parameterized functional set, where the weights, W, of the FFANN acts as the (index) parameters of the set, and the training algorithm acts a as search mechanism that identifies the function that gives the "best fit" to the training data. That is, the FFANN adaptive machine implements a set of functions $\mathcal{S} = \{\mathbf{f}(\mathbf{x}', w)\}$, where $w \in W$ and W acts as the set of parameter values (weights of the FFANN). The vector $\mathbf{x}'$ is the input vector to the system [6]. In general, this vector does not reflect the full component(s) of the vector $\mathbf{x}$ or $\mathbf{x}''$, (that is $dimension(\mathbf{x}') \leq dimension(\mathbf{x}'') \leq dimension(\mathbf{x})$), and may also be contaminated by noise.

The adaptive machine (in context to FFANN's) is characterized by the presence of an architecture for the FFANN, a procedure to find the error between the output generated by the FFANN and the output of the actual system (corresponding to a given input), and an algorithm or methodology for the adjustment of the FFANN's parameters based on the error thus estimated.

In general, the dimension of output vector generated by FFANN, $\mathbf{y}'$, may differ from the dimension of the output vector, $\mathbf{y}$, from the actual system ($dimension(\mathbf{y}') \leq dimension(\mathbf{y})$). This may be due to modeling constraints in the observational setting, in which some components of $\mathbf{y}$ may not be available because they were not measured. Or, in the experimental setting, some of components may not be available at all as the model, of the actual system used, does not generate these components. In both these cases, only the common components of the vectors ($\mathbf{y}'$ and $\mathbf{y}$) are used in the error calculation.

The question of fault tolerance arises only when the network has been trained over the given training data. Thus, we may ignore the absence of missing vector components from the input and the output vector data for training. But, it must be remembered that if the some components are missing from the training data, it will be reflected in the intrinsic error of the model (function) identified by the learning mechanism.

III. Fault Models

The formulation of the FFANN system as a learning system allows us to segregate the study of fault–tolerance behaviour of the FFANN into three separate sets, namely:

- Tolerance to faults/errors in the learning rule of the adaptive system (this includes performace / error function choice, weight initialization rule, training algorithm etc.) (algorithmic fault / error),
- Tolerance to faults/errors outside the adaptive machine (external / environmental faults / errors), and
- Tolerance to faults/errors in the structure of the adaptive machine (internal / architectural / structural faults/errors).

We define the algorithmic, external and internal faults below:

Definition 1: Any fault/error that is introduced because of value representation, error function calculation rule as well as its derivative's analytic behaviour, analytic behaviour of the transfer function used and their derivatives, parameter initialization techniques, data / input / output preprocessing / postprocessing method used, and quirks of the training algorithm is broadly classified as an *algorithmic fault.*

Definition 2: Any incorrectness in the input to the adaptive machine is defined as an *external fault.*

Definition 3: Any non-optimum working / value of any component or parameter of the FFANN that leads to incorrectness in the output of the system is defined as an *internal / architectural / structural fault.*

The algorithmic faults have a meaning for the answer to the question about which is the best training methodology (algorithm). This question is not persued in this paper. Our interest in this section is limited to developing a framework for the study fault–tolerance properties of already trained networks (but see section V). Though the question of which algorithm may lead to (trained) fault–tolerant networks for solving a task is important in its own right. The relation between generalization and fault–tolerance has been recognized [8], [9], [15], techniques that lead to better generalization capacity of the FFANN, will also lead to more fault–tolerant networks. The other two fault types – External and Internal – are detailed below.

A. External Faults

The external faults / errors are non-presence of some components of the input vector or contamination of inputs by noise.

1) External Fault Type I (EFTI): This is the error induced because some components of the input vectors are missing from the input data or, equivalently we may treat the missing component as a stuck at zero fault.

We do not know of a work that has investigated this type of fault. We feel that for the case of non-trivial dependency of the output on the input component that is missing, the output error will be very large (so as to make the FFANN unusable) for the case where the number of inputs to the FFANN is relatively small (dimensionality of the input vector

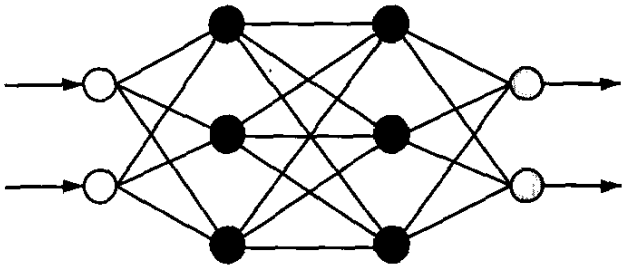

Fig. 2. Type - I Architecture, a 2-input, 2 hidden layer (3 nodes each) and 2 output network.

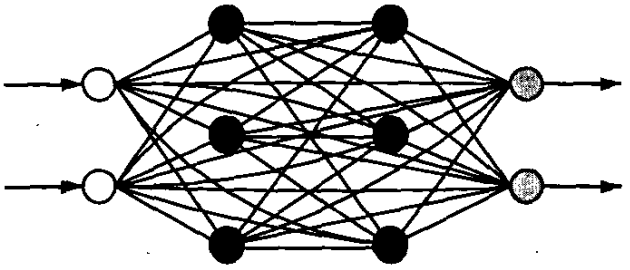

Fig. 3. Type - II Architecture, a 2-input, 2 hidden layer (3 nodes each) and 2 output network.

is small). But, for the case where the dimensionality of the input vector is relatively large,the error induced because of missing components should be investigated and methods found to mitigate the error induced.

2) External Fault Type II (EFTII): Input values stuck at a constant value (the different values that may be considered are – ● maximum or minimum of the corresponding input components. The case of the input component stuck at zero is not considered because it is equivalent to case (EFTI) considered in the section III-A.1. For this case also, we recommend an investigation in the case of large input networks only, as for them only a non-catastrophic output may be expected from the biological analogy.

3) External Fault Type III (EFTIII): White noise in the inputs, with the noise variance for the different components being different, in general. The noise presence may be due to measurement errors, modeling errors or data representation granularity. This fault/error should be investigated irrespective of whether the number of inputs is small or large. This error investigation is also an investigation of the error propagation through the network, the investigation of this error will allow us to comment on the stability of the FFANN as a computational mechanism. This fault has been investigated by [11]–[14], [17] Input noise introduction during training has been recognised to provide a regularization effect (See [15] for further references), this effect should be measured to see whether this leads to a more robust network or not.

B. Internal or Architectural Faults

To understand what may comprise an internal fault, we take a look at the two types of FFANN architure that may be thought of as the extreme end of an architectural spectrum (fig. 2 and 3). Fig. 2 represents a network in which the layer nodes are connected to adjacent layer nodes only (Type–I architecture) while fig. 3 represents a network in which layer nodes are connected to all subsequent layer nodes (Type–II architecture). There can be many variants in between these two extremes (like FFANN's in which nodes of a layer are connected to only a few nodes of preceding or succeeding layer nodes, etc.).

To enumerate the types of architectural faults / errors that can occur, we may define:

Definition 4: A FFANN is a 7-tuple $\{I, L, M, S, N, O, W\}$, where I denotes the indexed set of input nodes, L is an indexed set of layers (ordered left to right, and the the first layer is taken as the first hidden layer while the the last layer corresponds to the hidden layer just before the output layer), M denotes the cardinality of the set L, S is an indexed set of elements $S_i, i \in \{1, \ldots, M\}$ which denotes the size of the ith layer of the network, N is an doubly indexed set whose elements N_j^i (which denotes the jth node of the ith layer, ($i \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, S_i\}$) are a tuple of the form $\{A, \sigma\}$ specifying the input aggregation rule A and the activation rule σ, O is the set of output nodes, while W is also an indexed set representing the weights of the network.

In the Definition 4, the threshold is also modeled as a weight by the usage of a pseudo-input, this allows the treatment of thresholds as connection weights. This definition is general enough to encompass the full architectural spectrum including the Type–I and Type–II architecture and the case of more than one type of activation function and weight aggregation rule being used in the same network. We have treated the input faults separately in section III-A[1].

With this characterization of a FFANN, the study of architectural fault–tolerance reduces to the investigation of the tolerance to faults of L, M, S, N, O, and W.

The faults/errors may be classified as:

1) Layer Faults: These are the errors that affect hidden layer(s) at the same time. Following types of errors are defined:

1) Layer Stuck at Zero (LSZ):
   All nodes in a particular layer produce an output equal to zero.
2) Layer Stuck at One (LSO):
   All nodes in a particular produce an output equal to ± 1 (assuming that the tangent hyperbolic function, $f = tanh(x)$ is used as the activation function), if any other activation is used, then these two values should be modified to reflect the two limiting values of the activation function used and moreover these faults are relevant only if at least in each layer the activation functions are the same).
3) White Layer Noise (WLN):
   This is the presence of additive white noise that is correlated across the layer outputs. This may happen

[1]The system learning model puts the input nodes in the environment of the adaptive machine, and therefore defines a boundary for the the environmental (external) variables and the system (internal) variables.

because of voltage and/or current fluctuation affecting a complete layer.

Due to the architectural peculiarities of the FFANN architecture, some of these faults are catastrophic for the Type–I architecture (LSZ and LSO) therefore investigations should be limited to Type–II architectures only. The third fault (LN) should be investigated for all types of networks.

2) Hidden Node Faults: These are the errors that affect hidden nodes. Following types of errors are defined:

1) Node Stuck at Zero (NSZ):
   The node output is stuck at zero. [16], [17], [19], [20]
2) Node Stuck at One (NSO):
   Nodes produce an output equal to ± 1 (assuming that the tangent hyperbolic function, $f = tanh(x)$ is used as the activation function), if any other activation is used, then these two values should be modified to reflect the two limiting values of the activation function used. [21]
3) White Noise in Node (WNN):
   The presence of additive gaussian noise in the nodes' output with zero mean and fixed standard deviation. Thermal noise (then it can be parameterized by the temprature) and degradation of circuitry in time can lead to this type of noise in a FFANN hardware implementation.

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

1) Weight Stuck at Zero (WSZ):
   This fault corresponds to an open fault or connection breakage between two nodes [21], [22].
2) Weight Stuck at Maximum / Minimum (WSMa/WSMi):
   Weight stuck at a value of $\pm |W|_{\max}$ [21], 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 a 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.
3) 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. [8]

4) Output Node Faults: The output node faults can be described in an analogous manner to the input node faults (section III-A) but only the noise contamination effects are non-catastrophic and may be considered. [15]

IV. Error / Fault Measures

To measure the effect of faults / errors enumerated in the section III, we define the following types of error / fault / parameter sensitivity measures.

A. MSE and MAPE

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, while for classification problems, the percentage of misclassification, is suggested as a measure of fault / errors.

B. Parameter Sensitivity Measure

For measuring the sensitivity of the FFANN to the variation in a single parameter (both inputs and weights) for problems that have real output we may use the condition number [23], [24] (even for classification problem, we have access to real outout values at the output nodes as quantization / discretization for class definition is done thereafter, thus this measure is applicable in this case also). Let y_j be the output of the jth output node, and let ξ_i denote a parameter against which the variation of y_j is to be evaluated, (ξ's belong to the set composed of the inputs, and the weights, $\xi_i \in \{\mathbf{x}, W\}$). Then treating y_j as a function of ξ_i, we have:

$$E_i^{(1)} = \frac{1}{N} \sum_j \left| \left(\frac{\xi_i}{y_j} \right) \left(\frac{\partial y_j}{\partial \xi_i} \right) \right| \tag{1}$$

and the measure may be defined as

$$\left\langle E_i^{(1)} \right\rangle = \frac{1}{NP} \sum_{p=1,j}^{P} \left| \left(\frac{\xi_i^p}{y_j^p} \right) \left(\frac{\partial y_j^p}{\partial \xi_i^p} \right) \right| \tag{2}$$

where p is the pattern index. In the following the averageing over the patterns will be implicitly assumed and without any confusion we may use E to denote $\langle E \rangle$.

The (1) and (2) measures the sensitivity of the output to the parameter variation, similar to above we may define a measure of sensitivity as a variation of the MSE with the parameter change, as:

$$E_i^{(2)} = \left| \left(\frac{\xi_i}{\epsilon_i} \right) \left(\frac{\partial \epsilon_i}{\partial \xi_i} \right) \right| \tag{3}$$

where

$$\partial \epsilon_i = MSE\ variation\ for\ ith\ parameter\ variation$$
$$= \frac{1}{N} \left(\sum_j (y_j - t_j)^2 - \sum_j (y_j' - t_j)^2 \right) \tag{4}$$

where t's are the desired targets (if known, if the desired targets are not known, then we may modify (4) as (5)), y's are the outputs from the FFANN without any error while y''s are the corresponding outputs with the error induced in the ith parameter.

$$\partial \epsilon_i = \frac{1}{N} \left(\sum_j (y_j - y_j')^2 \right) \tag{5}$$

Equation (5) implies that we assume the network without the fault to be "perfect", that is, it produces the *desired* output for *any* given input.

The equation (4) and (5) allow us to use the FFANN itself to evaluate the change in MSE for small perturbations in the parameters.

For single faults, another set of sensitivity measures can be defined using the Taylor's expansion of the outputs (which measures the rate of variation of the output(s) (6) and the MSE (7), respectively, w.r.t. the parameters.

$$E_i^{(3)} = \frac{1}{N} \sum_j^N \left| \frac{\partial y_j}{\partial \xi_i} \right| \tag{6}$$

$$E_i^{(4)} = \left| \frac{\partial \epsilon_i}{\partial \xi_i} \right| \tag{7}$$

where variation in MSE is given by (4) or (5).

Fault measures for multiple faults is derived as follows

$$^{Multiple}E_i^{(m)} = \sum_{r \in \Phi} E_r^{(m)}; \quad m \in \{1, \ldots, 4\} \tag{8}$$

where Φ is the set of all parameters that has a fault.

V. Techniques for Fault-Tolerance Enhancement

The faults and errors enumerated in section III may be classified as:

- Faults that imply a removal of some structural element (weight or node), or some input component (hard faults), and
- Errors induced due to contamination by noise of the structural element value (weight value or node output) or the input value (noise contamination). Physical arguments can be given for the plausibility of the zero mean additive gaussian noise assumption.

The two classes of faults / errors differ intrinsically, as one is of absence or failure (large value error), the other corresponds to (small) value error only. Therefore, the methods for making the FFANN's tolerant to these two types of faults also differ.

A. Hard Fault Tolerance

One of the methods for increasing fault tolerance / reliability of electrical / electronic systems has been the incorporation of explicit redundancy. Phatak and Koren [21] and Emmerson and Damper [25] established that once a network has been found to solve some problem, then the replication of its hidden units leads to a network that can lead to complete tolerance of single faults. Moreover, they established that this explicit redundancy incorporation need not be complete for the FFANN to have a high degree of fault-tolerance.

Phatak [26] established that using permuted examples and / or cross-validation does not yield fault tolerance enhancement. The method of weight update (epoch bases versus pattern based) does not affect the fault tolerance of ANN's.And, provide a penalty term that explicitly takes into account the error caused by single node faults (NSZ). A similar algorithm based on optimization using genetic algorithms (where the penalty term, incorporating the cost of weight faults, WSO and WSZ, is used together with the standard mean squared error term as a fitness parameter) has been proposed by Zhou and Chen [10].

Another method proposed has been the incorporation of the hard faults during training itself. Sequin and Clay [18] incorporated hard faults (randomly) in hidden nodes during training and demonstrated that the fault-tolerance of FFANN's to these faults increase. This technique of fault injection during training so that (apparently) the FFANN learns to perform its operation in the presence of the faults. This method has been extended to be used for the tolerance of noise contamination also.

Similar techniques based on weight fault's have been proposed. Hammadi and Ito [22] proposed an algorithm that estimates the relevance of a weight by faulting it and measuring the change in the output (hessian w.r.t. weights is used). The weight with the maximum relevance is decreased.

B. Noise Contamination Tolerance

The noise contamination models enumerated in section III are used during training itself to produce a network that is capable of more tolerance to these errors [13], [14], [17]. The incorporation of noise in the inputs during training has been shown to act as a Tikhonov's regulizer. [15], [27]. Similarly, Murray and Edwards have shown that the weight noise also acts as a positive penalty term [8] and leads to better fault-tolerance. In [15] we have proposed a general regulizer term derived from a general noise model for FFANN's that brings together all approaches to noise injection and establishes the regularization effect of noise injection and their interplay.

The penalty term proposed by [26] and the fitness criteria specified in [10] also act as a model complexity control, and tries to evenly distribute the outputs of the hidden nodes. Thus, effectively complexity control mechanisms, including regularization, try to distribute computation through out the network, this leads to better generalization as well as fault-tolerance.

The above quoted results establish that the regulization scheme, that is used for complexity control or to improve the generalization capability of the network, also leads to more fault-tolerant network. Thus, we conjecture that fault-tolerance has a beneficial effect on the generalization capability of the network and vice-versa.

C. Other Techniques

It has been found that the presence of large magnitude weights adversely affects the fault-tolerance of FFANN's. Thus, techniques have been designed that incorporate strategies for weight size control and thereby increasing the fault-tolerance [28]. Weight splitting has been another method of controlling weight sizes. This is exemplified by the class of methods developed by Chiu et al [29], wherein nodes with high sensitivity are split (that is associated weights are shared).

The role of batch size for calculating the averaged gradient has been investigated for its effect on the fault-tolerance property, and the result has been negative in the sense that there is no statistically significant dependence. [26]

Activation function's role in finding a fault–tolerant FFANN instance has been investigated and a "better" activation function proposed. [30]

VI. Discussions and Conclusions

The FFANN architecture that has been investigated have a commonality of having only one hidden layer and using a sigmoid activation function(s). This may be due to the popularity of sigmoidal networks as compared to other feedforward networks (like those based on radial basis functions). Which itself can be due to the usage of FFANN's for solution of classification tasks (which have been used, usually, to investigate the fault–tolerance property), as sigmoidal networks are supposed to be better suited for classification tasks [31]. The role that activation functions (or, to find a solution to the question – Does there exist an activation function that leads to a more fault–tolerant architecture? See [32], [33] for other sigmoidal and non–sigmoidal functions that can be used in a FFANN.), and other choices made (like weight initialization rule, training algorithm used, etc.) to fix the neural network architecture has not been investigated to the extent desired. Also, it is known that the FFANN training is sensitive to the initial conditions [34].

Another common feature is the use of small hidden layer (numbere of hidden layer nodes is "small"), that is the usage of hidden nodes ranging only in a few tens [8], [11], [12], [16]–[19], [21], [25], [29]. Though the number of adjustable parameters may run into a few thousands because of the multiplicity of inputs and / or outputs [21]. These networks cannot be called "massive" as compared to the biological networks. Therefore, there exists a need to investigate networks with multiple and large hidden layers for the possession or absence of fault–tolerance.

In this paper, we have used the learning system model of the FFANN's to propose a framework for the analysis of fault–tolerance properties of feedforward networks. The proposed frameworks suggests fault / error models and associated fault / error measures to be used for the study of fault–tolerance of FFANN's. Existing results are briefly analyzed within the proposed framework. The issues raised in this work needs to be investigated in detail before any conclusion can be drawn labeling FFANN's as fault–tolerant or intolerant.

References

[1] N. K. Bose and P. Liang, *Neural Network Fundamentals with Graphs, Algorithms and Applications*, New York: McGraw–Hill, 1996.

[2] F. M. Ham and P. Kostanic, *Principles of Neurocomputing for Science & Engineering*, New York: McGraw–Hill, 2000.

[3] R. J. Schalkoff, *Artificial Neural Networks*, Singapore: McGraw–Hill, 1997.

[4] R. M. Golden, *Mathematical Methods for Neural Network Analysis and Design*, Cambridge: MIT Press, 1996.

[5] S. Haykin, *Neural Networks – A Comprehensive Foundation*, Singapore: Prentice Hall, 1999.

[6] V. Cherkassky and F. Mulier, *Learning from data – Concepts, theory, and methods*, New York: John Wiley & Sons, 1998.

[7] C.-L. Cheng and J. W. V. Ness, *Statistical regression with measurement error*, London: Arnold, 1999.

[8] A. F. Murray and P. J. Edwards, Synaptic weight noise during MLP training: 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.

[9] Z.-H. Zhou, S.-F. Chen and Z.-Q. Chen, Improving tolerance of neural networks against mult-node open faults. In *IJCNN'01*, vol. III, pp. 1687–1692, 2001.

[10] Z.-H. Zhou and S.-F. Chen, Evolving fault–tolerant neural networks. *Neural Computing and Applications*, 2003 (In press).

[11] P. Kerlirzin and F. Vallet, Robustness in multilayer perceptrons. *Neural Computation*, vol. 5, pp. 473–482, 1993.

[12] M. J. Carter, The illusion of fault tolerance of neural nets for pattern recognition and signal processing. In *Proc. of the Technical Session on Fault Tolerant Integrated System*, Durham: University of New Hampshire, 1988.

[13] L. Holmstrom and P. Koistinen, Using additive noise in back–propagation training. *IEEE Trans. on Neural Networks*, vol. 3, pp. 24–38, 1992,

[14] K. Matasuoka, Noise injection into inputs in backpropagation learning. In *IEEE Trans. Systems, Man, and Cybernetics*, vol. 22, pp. 436–440, 1992.

[15] P. Chandra and Y. Singh, Regularization and feedforward artificial neural network training with noise. In *IJCNN'03*, 2003, (Accepted).

[16] B. E. Segee and M. J. Carter, Fault sensitivity and nodal relevance relationships in multilayer perceptrons. *Technical Report ECE.IS.90.0*, Intelligent System Group, Robotics Laboratory, Electrical and Computer Engineering Department, Durham: University of New Hampshire, March 1990.

[17] J. I. Minnix, Fault tolerance of backpropagation neural network trained with noisy data. In *IJCNN'91*, vol. 1, pp. 847–852, 1991.

[18] C. H. Sequin and R. D. Clay, Fault tolerance in artificial neural networks. In *IJCNN'90*, vol. 1, pp. 703–708, 1990.

[19] C. Neti, M. H. Schneider and E. D. Young, Maximally fault tolerant neural networks. *IEEE Trans. Neural Networks*, vol. 3, pp. 14–23, 1992.

[20] T. R. Damarla and P. K. Bhagat, Fault tolerance of neural networks. In *IEEE Proc. 1989 SOUTHEASTCON*, pp. 328–331, 1989.

[21] D. S. Phatak and I. Koren, Complete and partial fault tolerance of feedforward neural nets, *IEEE Trans. Neural Networks*, vol. 6, pp. 446–456, 1995.

[22] N. C. Hammadi and H. Ito, A learning algorithm for fault tolerant feedforward networks. *IEICE Trans. Information and Systems*, vol. E80-D, no.1, pp. 21–27, 1997.

[23] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, New York: McGraw Hill, 2002.

[24] D. Kincaid and W. Canale, *Numerical Analysis: Mathematics of Scientific Computing*, (3rd Eds.), Pacific Grove, CA: Brooks/Cole, 2002.

[25] M. D. Emmerson and R. I. Damper, Determining and improving the fault tolerance of multilayer perceptrons in a pattern recognistion application. *IEEE Trans. Neural Networks*, vol. 4, pp. 788–793, 1993.

[26] D. S. Phatak, Fault tolerant artificial neural networks. In *Proc. of the 5th Dual Use Technologies and Applications Conf.*, Utica/Rome, NY, pp. 193–198, 1995.

[27] C. Bishop, "Training with noise is equivalent to Tikhonov regulariation," *Neural Computation*, vol. 7, no. 1, pp. 108–116, 1995.

[28] D. Simon, Distributed fault tolerance in optimal interpolative nets. *IEEE Trans. Neural Networks*, vol. 12, pp. 1348–1357, 2001.

[29] C.-T. Chiu, K. Mehrotra, C. K. Mohan and S. Ranka, Training techniques to obtain fault tolerant neural networks. In *Digest FTCS*, pp. 360–369, 1994.

[30] N. C. Hammadi and H. Ito, On the activation function and fault tolerance in feedforward artificial networks. *IEICE Trans. Info. and Systems*, vol. E81-D, pp. 66–72, 1998.

[31] Y. LeCun, L. Bottou, G. B. Orr and K.-R. Muller, Efficient BackProp. In *Neural Networks: Tricks of the Trade*, G. Orr and K. Muller, Eds., pp. 9–50. Springer Verlag, Berlin, 1998.

[32] W. Duch and N. Jankowski, Survey of neural transfer functions. *Neural Computing Surveys*, vol. 2, pp. 163–212, 1999. *http://www.icsi.berkeley.edu/~jagota/NCS*.

[33] Y. Singh and P. Chandra. A class + 1 activation functions for FFANN's. *Journal of Economic Dynamics and Control*, 2003 (In Press).

[34] G. W. Kolen and J. B. Pollock, Backpropagation is sensitive to initial conditions. In *Technical Report TR 90-JK-BPSIC*, Lab. of AI Research, Computer and Information Sc. Deptt., Columbus: The Ohio State University, 1990.