A Matlab Tool for Analyzing and Improving Fault Tolerance of Artificial Neural Networks

Rui Borralho*. Pedro Fontes*. Ana Antunes*. Fernando Morgado Dias.**

**Escola Superior de Tecnologia de Setúbal do Instituto Politécnico de Setúbal,*
Departamento de Engenharia Electrotécnica, Campus do IPS, Estefanilha, 2914-508 Setúbal, Portugal
Tel: +351 265 790000, Fax: +351 265 721869
(e-mail:author: pedro.fontes@netvisao.pt ; rborralho@esce.ips.pt ; aantunes@est.ips.pt ;)
***Departamento de Matemática e Engenharias, Universidade da Madeira*
Campus da Penteada, 9000-390 Funchal, Madeira, Portugal
Tel: +351 291-705150/1, Fax: +351 291-705199
[3]Centro de Ciências Matemáticas - CCM
Universidade da Madeira, Campus da Penteada
9000-390 Funchal, Madeira, Portugal
Tel: + 351 291 705181, Fax: + 351 291 705189 (e-mail: morgado@uma.pt)

Abstract: FTSET is a software tool that deals with fault tolerance of Artificial Neural Networks. This tool is capable of evaluating the fault tolerance degree of a previously trained Artificial Neural Network given its inputs ranges, the weights and the architecture. The FTSET is also capable of improving the fault tolerance by applying a technique of splitting the connections of the network that are more important to form the output. This technique improves fault tolerance without changing the network's output. The paper is concluded by two examples that show the application of the FTSET to different Artificial Neural Networks and the improvement of the fault tolerance obtained.

Keywords: Artificial Neural Network, Fault tolerance, Simulator, Hardware.

1. INTRODUCTION

Fault tolerance in Artificial Neural Networks (ANNs) has been a topic almost forgotten in the last decade. Only a few papers concerning this topic have been published after 1994: (Eickhoff and Rückert, 2005), (Protzel et al. 1993), (Arad and El-Amawy, 1997), (Cavalieri and Mirabella, 1999), (Phatak, 1995). In spite of that, the intrinsic fault tolerance of Neural Networks is a very important characteristic. As stated in [9]: "The characteristic of a graceful performance degradation without additional redundancy is specially interesting for applications such as long-term, unmanned space missions, where component failures have to be expected but no repair or maintenance can be provided". Recently the utility of ANNs' built in fault tolerance was also pointed out within nano-electronic systems (Eickhoff and Rückert, 2005). Beyond these areas, situations where hardware acts inside the human body can make use of both fault tolerance and graceful degradation. It is suitable that in the presence of a malfunction such hardware still performs, at least, part of its functions instead of showing complete failure.

Fault tolerance (FT) is the capacity of a system to perform correctly under the presence of a fault. Graceful Degradation (GD) refers to the performance of a system that does not exhibit complete failure in the presence of a fault. This means that the performance might degrade due to a fault, but in a graceful way.

To study FT and GD in a general way a global model for the faults is needed. Several authors have addressed this question and proposed solutions but, as will be shown here, the solutions proposed have all left uncovered some important situations.

This paper proposes the fault model that the authors believe to be more accurate and general (regardless of the implementation type used in the hardware). This model is the first step that led to a proposal of the changes in the ANNs that can improve the built in fault tolerance.

Both the model and the solutions proposed for improving fault tolerance are implemented in the Fault Tolerance Simulation and Evaluation Tool for Artificial Neural Networks (FTSET).

The remainder of this paper is organized as follows: section 2 shoes the kind of ANNs that the FTSET deals with section 3 discusses fault tolerance in ANNs, section 4 introduces the FTSET tool, section 5 presents examples and results obtained with the tool and section 6 presents the conclusions of the paper.

2. ARTIFICIAL NEURAL NETWORKS

This section describes the types of ANNs that can be used with the FTSET tool. The tool deals with feedforward ANNs. In this type of networks, shown in figure 1, the signal flows in a single direction: from input to output. There are no lateral or feedback connections.

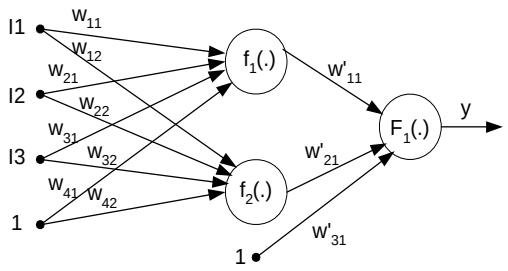


Fig. 1. Feedforward Neural Network structure.

The network of figure 1 is referred to as having an input layer, a hidden layer and an output layer. The intermediate layer is referred this way because its inputs and outputs are not directly accessible.
The activation functions can be chosen to be: Heaviside, Symmetrical Heaviside, Linear, Linear with saturation, Linear Symmetrical with saturation, Hyperbolic Tangent and Log-sigmoidal.
The FTSET does not impose a limit in the size of the network either in terms of inputs, outputs, hidden layers or number of weights. Nevertheless, the user should be aware that the larger the network is, the longer the calculations will take.

3. FAULT TOLERANCE IN ARTIFICIAL NEURAL NETWORKS

Fault tolerance has been an important topic among the Neural Network community. Although it has been mostly assumed that ANNs are inherently fault tolerant, there are several reports that study this subject: (Chiu et. al, 1993), (Phatak, 1995), (Piuri et. al, 1991) and (Tchernev and Phatak, 2005) deal with models to evaluate and represent fault tolerance; (Arad and El-Amawy, 1997) and (Cavalieri and Mirabella, 1999) consider different solutions for improving fault tolerance and (Bolt, 1991), (Eickhoff et. al, 2005), (Elsimary et. al, 1995) and (Protzel et. al, 1993) analyze ANNs in the fault tolerance perspective.
The model for representing fault tolerance in the present work was discussed in (Dias and Antunes, 2008a) and the solution that motivated the construction of the simulator was presented in (Dias and Antunes, 2008b).
This solution improves the fault tolerance of a fully trained network through architecture change. The details will not be discussed here but the principle is that, independently of the activation functions, it is possible to modify an ANN while maintaining the outputs if the sum at each neuron remains the same. Based in this axiom the ANN can be analyzed for the points where a failure can have a highest impact in the output and this impact can be reduced while maintaining the outputs unchanged. This results in some of the connections being split and their weights set to half of their value.

4. THE FTSET SIMULATOR

The FTSET simulator is composed of three main sub-tools. The first one is responsible for receiving the parameters of an ANN, the second does the evaluation of the fault tolerance capabilities and the last one improves the fault tolerance capability of the ANN.
Figure 2 shows a block diagram of the FTSET simulator where the interaction of the sub-tools can be seen.

4.1 The Insertion sub-tool

The Insertion sub-tool is responsible for receiving the parameters of the ANNs that will be evaluated and improved with respect to their fault tolerance. The simulator can receive networks in three different formats: an Excell format (defined in the user's manual), a Matlab file and it can also be inserted manually in the main window (see figure 3) by selecting the appropriate parameters.

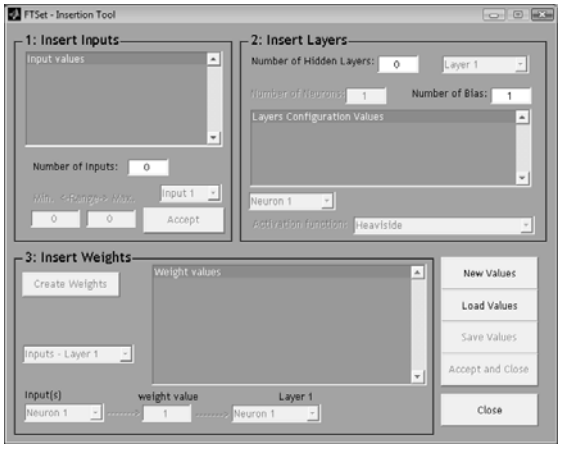


Fig. 3. Main window of the Insertion sub-tool.

4.2 The Evaluator sub-tool

The Evaluator sub-tool allows the simulation of the output of the network. The main window is shown in figure 4.
The faults are considered according to the models proposed in (Dias and Antunes, 2008a).

The evaluation of the GD is done in percentage, according to the following expression:

$$GD = \frac{output\_with\_fault - output\_without\_fault}{output\_without\_fault} \quad \text{Eq. 1}$$

To evaluate the GD, the output of the network must be calculated in several different situations. Since the input can take several values and it is not possible to consider a single value of the inputs to measure the effects of each fault, the limits in the range of values for each input will be used as a worst case value.
After calculating the outputs for every possible combination of the inputs ranges, a matrix of possible values for the output is obtained. For each possible single fault, the network must be evaluated in the very same situations taken before simulating the fault, resulting for each fault in a new matrix that must be compared with the one obtained in the absence of faults.

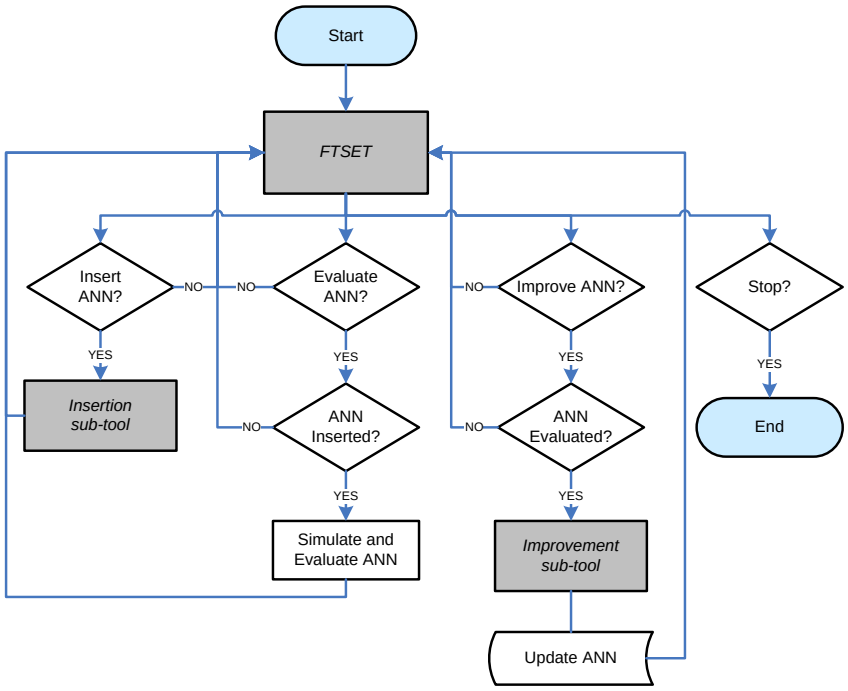


Fig. 2. Block diagram of the FTSET simulator.

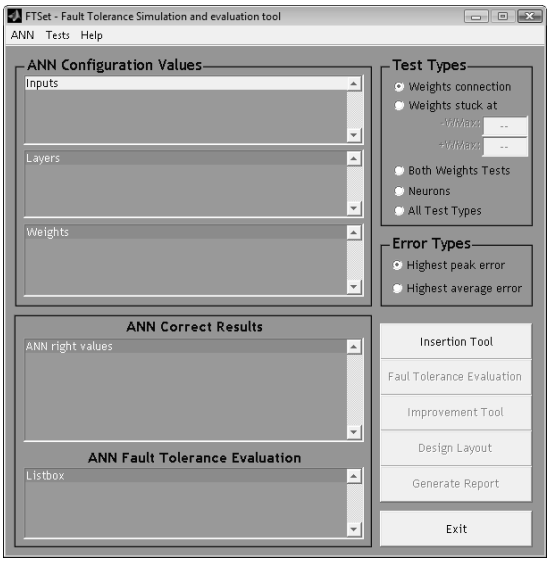


Fig. 4. Main window of the Evaluator sub-tool.

As can be easily observed, this procedure has a rapid growth of calculations with the size of the network.
In fact the number of calculations needed can be evaluated from the size of the network through the following expressions:

The number of outputs to be calculated initially is:

$$n\_out\_calc = X^I \qquad \text{Eq. 2}$$

Where X is the number of possible values that the input can assume and I is the number of inputs.
The total number of weights for a network of k layers is given by:

$$n\_weights = (I+1)*K_1 + (K_1+1)*K_2 + ... + (K_{n-1}+1)*K_n + (K_n+1)*O \qquad \text{Eq. 3}$$

Where K_j is the number of neurons in the layer j and O is the number of outputs.
Equation 3 can be written as:

$$n\_weights = (I+1)*K_1 + \sum_{j=1}^{n-1}(K_j+1)*K_{j+1} + (K_n+1)*O \qquad \text{Eq. 4}$$

The number of calculations that has to be performed is then:

$$n\_calculations = (n\_weights+1)*n\_out\_calc \qquad \text{Eq. 5}$$

Or

$$n\_calculations = ((I+1)*K_1 + \sum_{j=1}^{n-1}(K_j+1)*K_{j+1} + (K_n+1)*O+1)*X^I$$

Eq. 6

From this equation, with the size of the network to be evaluated, an estimate of the complexity of the calculations that need to be performed can be obtained.
Apart from estimating the complexity, the Evaluator has to calculate every output in the presence and absence of a fault and determine the effect of each fault according to equation1. This can be done in two different ways: searching the highest peak error and largest average error, according to the options chosen by the user in the Improver sub-tool.

4.3 The Improver sub-tool

The GD improvement tool, the Improver (whose main window is shown in figure 5), is based on the algorithm proposed in [12].
This algorithm proposes the augmentation of the GD capabilities through a two step methodology: detection of the most sensible connections or neurons to a fault and replacement of these connections or neurons by two connections or neurons which are less sensible to a fault. This results in an iterative procedure that can be applied in several different ways.

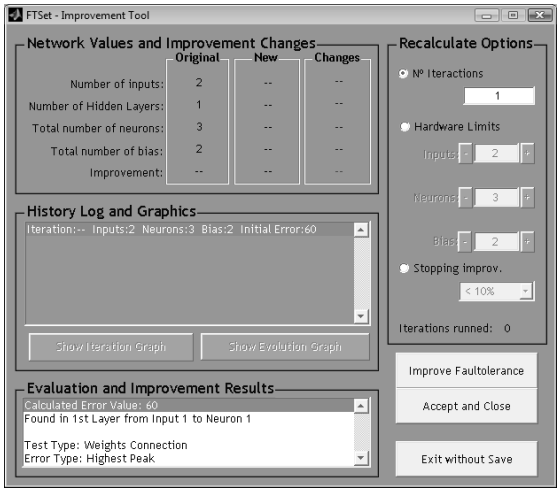


Fig. 5. Main window of the Improver sub-tool.

Because of this, the Improver is implemented here with several options:
At the implementation level:

- Reduce the importance of the connection that has the highest peak error introduced by a single fault.
- Replace the importance of the connection that shows the highest average error.

At the stopping condition level:

- Stopping when the limits (in terms of number of neurons and inputs) of the hardware are reached.
- Stopping after a certain number of iterations
- Stopping when the fault tolerance/GD exhibited reaches a predefined level

The options can be chosen by the user at the very beginning, resulting in different sets of weights being supplied to the user at the end.
The FTSET shows the improvements obtained at each step and supplies the user with the configuration and the weights of the network after the GD has been improved.

5. APPLICATION EXAMPLES

This section presents two examples of the simulator improving the GD of an ANN.

5.1 Example 1

The network for the example is a feedforward ANN with 2 inputs, 1 hidden layer with 2 neurons and 1 output neuron. The hidden layer's activation functions are hyperbolic tangents while the output neuron has a linear function.
For the evaluation of the fault tolerance, the range of the inputs is needed. The range of both inputs can be seen in the matrix 1.

$$\begin{bmatrix} -5 & 5 \end{bmatrix}; \begin{bmatrix} -3 & 7 \end{bmatrix} \qquad \text{mat. 1}$$

This range has to be supplied by the user. It can be determined by the knowledge of the inputs nature or by the hardware characteristics.
The weights associated with each layer of the ANN can be seen in the matrix 2.

$$\begin{bmatrix} 0.5 & -0.9 \\ -0.1 & 0.3 \\ 0.9 & -0.4 \end{bmatrix} \begin{bmatrix} 0.5 & -0.7 & 0.8 \end{bmatrix} \qquad \text{mat. 2}$$

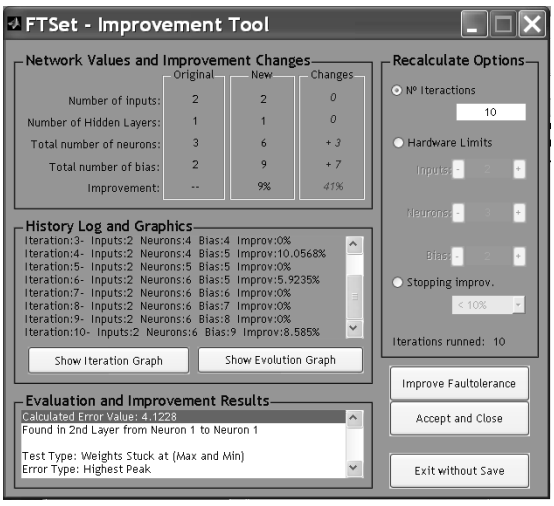


Fig. 6. Final result of the Improver.

Selecting the "weights stuck at" in the test type with the highest peak error option, the initial evaluation of the fault tolerance produces a value of 6.97. This option allows using

the range of the weights as obtained from the amount of values available or selecting a different range if that is convenient for the type of implementation (for instance selecting the limits that result from a certain hardware implementation or analog to digital converter).
This value obtained by the Evaluator is the base value that the FTSET will try to improve.
For this example the Improver will work with a request of 10 iterations.
After these iterations, as can be seen in figure 6, 3 additional neurons were introduced, 7 new bias inputs were added and the fault tolerance has improved 41%. This means that the worst failure evaluated in the highest peak error mode was of 6.97 and is now of only of 4.1228.
Naturally the improvement of tolerance came at the cost of using more hardware than it was used at the initial implementation.
In Figure 7 the evolution of the relative error is shown. Here it is possible to see that some iterations do not contribute at all to improving the fault tolerance level. This can be explained quite easily. If one connection is split, because of its high impact on the output, two new connections are created with half of the weight. If afterwards these connections are evaluated as having the highest importance then two iterations are needed to solve the problem. In this situation the first of such iterations does not result in any improvement since the second iteration finds a similar fault tolerance level due to the existence of two similar weights.

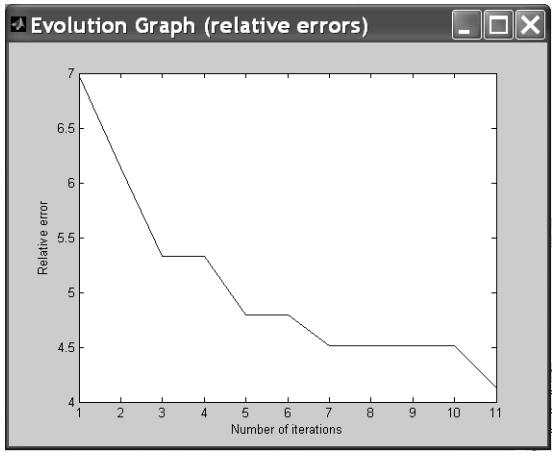


Fig. 7. Evolution of the fault tolerance level during the 10 iterations of the example.

The weights modified by the Improver sub-tool can now be saved and exported for the user to apply them where necessary.

5.1 Example 2

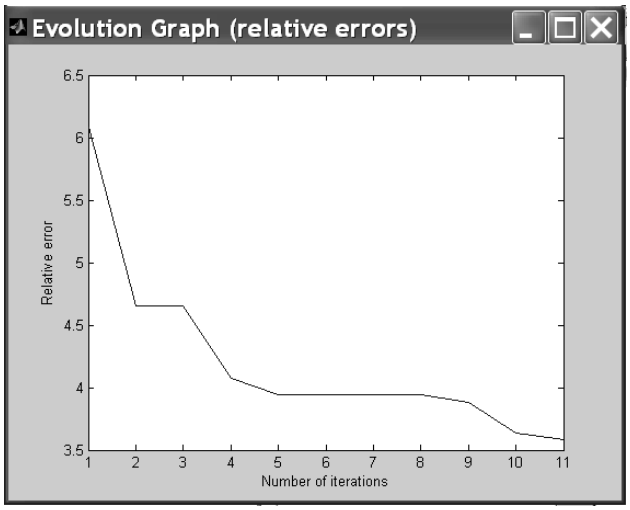
For the second example a network a little larger was selected. This ANN has 4 inputs, 3 neurons in the hidden layer, all with hyperbolic tangent has activation function and one output neuron with linear activation function. . The range of the inputs can be seen in the matrix 3and the matrices with the weights are represented in matrix 4.

$$[1 \quad 6] \; ; [-3 \quad 1] ; [0 \quad 4] ; [-2 \quad 2] \qquad \text{mat. 1}$$

$$\begin{bmatrix} 2.7 & -2.4 & 0.8 \\ -1.4 & 1.3 & 2.5 \\ -1.2 & -2.3 & -0.7 \\ 0.8 & -2.1 & -0.9 \\ 0.2 & -0.1 & 0.3 \end{bmatrix} [1.2 \quad -2.4 \quad 1.6 \quad -0.2]$$

mat. 4

The options used for this test are: “All test types”, “Highest average error” and an improvement with 10 iterations.
The initial fault tolerance value is 6.087 and after 10 iterations an improvement of 41% was achieved and 3.581 fault tolerance value has been reached.
For this improvement 9 additional neurons and 1 additional bias input was needed.
The evolution is shown in figure 8.



6. CONCLUSION

The FTSET tool implements a software tool to evaluate fault tolerance/graceful degradation in NN.
The tests made show that the methodologies implemented allow assessing the GD accurately and allow improving this characteristic without the need to recalculate the weights of the network.
The utility of the FTSET is also shown here through two examples where, with only 10 iterations, it was possible to enlarge the fault tolerance capability by 41% in both cases. This improvement is achieved at the expense of additional resources for the hardware implementation.
The FTSET is also very versatile when choosing how to improve the ANN’s graceful degradation, allowing the stopping condition to be determined by the number of iterations, the GD level or the limits of the hardware that can be physically implemented.

REFERENCES

B. S. Arad and A. El-Amawy, “On Fault Tolerance Training of Feedforward Neural Networks”, Neural Networks, Vol. 10/3, pp.539-553, 1997.

G. Bolt, “Investigating the Fault Tolerance in Artificial Neural Networks”, Technical Report YCS 154 from the University of York, available at the internet, 1991.

S. Cavalieri and O. Mirabella, “A novel learning algorithm which improves the partial fault tolerance of multiplayer neural networks”, Neural Networks, Vol.12, pp. 91-106, 1999.

C. Chiu, K. Mehrotra, C. Mohan and S. Ranka, “Robustness of Feedforward Neural Networks”, 2nd IEEE International Conf. on Neural Networks, Vol. 2, pp. 783-788, 1993.

R. Eickhoff and U. Rückert, “Tolerance of Radial Basis Functions Against Stuck-at-Faults”, International Conference of Artificial Neural Networks, pp. 1003-1008, Poland, 2005.

H. Elsimary, S. Mashali and S. Shaheen, “Generalization Ability of Fault Tolerant Feed Forward Neural Networks”, International Conference on Systems, Man and Cybernetics, Vol. 1, pp. 30-34, 1995.

D. S. Phatak, “Complete and Partial Fault Tolerance of Feedforward Neural Nets”, IEEE Transactions on Neural Networks, Vol. 6/2, pp. 446-456, 1995.

V. Piuri, M. Sami, R. Stefanelli, “Fault Tolerance in Neural Networks: Theoretical Analysis and Simulation Results”, 'Advanced Computer Technology, Reliable Systems and Applications', 5th Annual European Computer Conference, Bologna, Italy, 1991.

E. B. Tchernev and D. S. Phatak, “Investigating the fault tolerance of neural networks”, NCA, 2005.

P.W. Protzel, D. L. Palumbo and M. K, Arras, “Performance and fault tolerance of neural networks for optimization”, IEEE transactions on Neural Networks, vol.4, 1993.

F. Morgado Dias and A. Antunes, “A global model for fault tolerance of feedforward neural networks”, 9th International Conference on Automation and Information, Bucareste, Roménia, June 2008.

F. Morgado Dias and A. Antunes, “Fault Tolerance Improvement through Architecture Change in Artificial Neural Networks”, 3rd International Symposium on Intelligence Computation and Applications (ISICA 2008) Wuhan, China, vol. LNCS 5370, pp. 248–257, December, 2008.