A software Tool for Teaching Fault Tolerance in Artificial Neural Networks

Pedro Fontes[1], Rui Borralho[1], Ana Antunes[1], Fernando Morgado Dias[2,3]
[1]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
[2]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 Universitário da Penteada
9000-390 Funchal, Madeira, Portugal
Tel: + 351 291 705181, Fax: + 351 291 705189
Email: pedro.fontes@netvisao.pt ; rborralho@esce.ips.pt ; aantunes@est.ips.pt ; morgado@uma.pt

Abstract
This paper presents a software tool for teaching, evaluating and improving the fault tolerance of an Artificial Neural Network.
Fault tolerance is a characteristic of parallel distributed systems such as neural networks. This property, though usually pointed out as one of the main advantages of the Neural Networks, is difficult to evaluate and can be a residual property allowing to maintain only a small part of the Neural Network characteristics.
The fault tolerance of the Artificial Neural Networks (ANN) can be very important in critic applications such as the ones of nanotechnologies in interaction with the human body. In these situations, retaining a part of the characteristics of the Neural Network by fault tolerance or graceful degradation (GD) is crucial.
To evaluate and teach the concepts of fault tolerance associated with neural networks (NNs) a software tool was developed. The tool is called Fault Tolerance Simulation and Evaluation Tool for Artificial Neural Networks (FTSET) and is composed of two main sub-tools (the Evaluator, for evaluation of the tolerance and the Improver, for improving the built-in tolerance).
The FTSET is able to evaluate the fault tolerance in feedforward ANN of arbitrary size, though the bigger the network, the longer it takes to evaluate its characteristics.
The tool can be used to evaluate fault tolerance from the set of weights of the network and the ranges of the inputs. The expression fault tolerance is in fact an abuse for most of the situations, since for real valued outputs what can be achieved is GD. Each fault does in fact affect the output, but for most of the situations the effect is not catastrophic and a concept of GD applies.
The graceful degradation evaluation follows an exhaustive approach, testing all possible faults in the network (the faults mentioned here are physical faults that may happen in the hardware). It supplies a numerical analysis of the results obtained to the user and this information is then used by the Improver to build a NN with a higher GD degree.
Changing the structure of an ANN to improve GD is a complex task since it may involve dealing with nonlinear functions such as the common activation functions hardlimit, hyperbolic tangent or other sigmoidal functions. (Dias & Antunes, 2007b) proposed a solution that involves duplicating part of the network (inputs, neurons, bias) to improve this characteristic, after the evaluation of the critical connections in terms of output impact. The Improver implements the ideas proposed in this paper.
The proposed tool is very useful to teach the concepts of fault tolerance and GD and to illustrate these concepts associated with neural networks.
The software tool presented here is the result of the final project of the Engineering degree (5 years) in Electronics and Computers and was implemented by a group of 2 students during a full year.

The Evaluator sub-tool
The faults are considered according to the models proposed in (Dias & Antunes, 2007a). 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 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 considerer 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.
As can be easily observed this procedure has an exponential 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: $n\_out\_calc = X^I$ 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:

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

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:

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

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

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

Or

$$n\_calculations = ((I+1)*K_1 + \sum_{j=1}^{n-1}(K_j+1)*K_{j+1} + (K_n+1)*O+1)*X^I \quad \text{Eq. 6}$$

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.

The Improver sub-tool

The GD improvement tool, the Improver, is based on the algorithm proposed in (Dias & Antunes, 2007b).
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.
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 configuration of the network after the GD has been improved.

Conclusion

The FTSET tool implements a software tool to evaluate fault tolerance/graceful degradation in NN.
The tests made by the authors show that the methodologies implemented allow to access the GD accurately and allow improving this characteristic without the need to recalculate the network.
The FTSET is also very versatile when choosing how to improve the NN'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 implemented.

References

Dias, F. M. and Antunes, A., "Fault tolerance of feedforward neural networks: an open discussion for a global Model", submitted to Engineering Applications of Artificial Intelligence, 2007.
Dias, F. M. and Antunes, A., "Fault Tolerance improvement through architecture change", submitted to Engineering Applications of Artificial Intelligence, 2007.
Lippmann, R. P., "An introduction to computing with neural nets", IEEE ASSP Magazine, 4-22, 1987.