# BACKPROPAGATION ARCHITECTURE OPTIMIZATION AND AN APPLICATION IN NUCLEAR POWER PLANT DIAGNOSTICS

Anujit Basu
Eric B. Bartlett

Department of Mechanical Engineering
Iowa State University
Ames, Iowa 50011.

This paper presents a Dynamic Node Architecture (DNA) scheme to optimize the architecture of backpropagation Artificial Neural Networks (ANNs). This network scheme is used to develop an ANN based diagnostic adviser capable of identifying the operating status of a nuclear power plant. Specifically, a "root" network is trained to diagnose if the plant is in a normal operating condition or not. In the event of an abnormal condition, another "classifier" network is trained to recognize the particular transient taking place. These networks are trained using plant instrumentation data gathered during simulations of the various transients and normal operating conditions at the Iowa Electric Light and Power Company's Duane Arnold Energy Center (DAEC) operator training simulator.

## Introduction

The safe operation of a nuclear reactor in a power plant is of utmost importance to the nuclear engineering community and quite vital to creating a positive attitude towards nuclear energy among the rest of the society. This paper is an attempt to demonstrate how ANNs can increase the operational safety of nuclear reactors by being the basis of a fault diagnostic system in a power plant. It is hoped that neurocomputing, as the science of neural networks is sometimes called, will provide a fast and reliable approach to recognizing and classifying operational transients at a nuclear power plant.

Most power generating stations currently employ automatic safety systems that allow the plant to operate within a predefined normal operating parameter space. These systems verify that the operating status conforms to the preassigned safety limits of the various plant variables. As the plant enters into an abnormal condition, indications of plant variables exceeding the normal range causes the safety systems to either trigger a scram that automatically shuts down the reactor or notify the operators through some alarms or indicators. The sequence of events leading to the plant shutdown are analyzed later by technical support teams located both on and off site. Use of the proposed adviser would be helpful in better understanding these events in real time.

Diagnostic systems being developed at this time almost always rely on elaborate expert systems to evaluate the current plant status. Sometimes, these expert systems are interactive with the operator. Also, they go through a long, computation intensive, fault-tree type diagnosis routine. This may make them slow to respond in an emergency situation. The proposed adviser is expected to have quicker response, thus providing the operators with more time to rectify the problem, mitigate any possible damage, and save the plant from an unnecessary shutdown.

This paper is part of an ongoing project at Iowa State University to develop ANN based fault diagnostic systems to detect and classify operational transients at nuclear power plants. The project envisages the deployment of such an adviser at Iowa Electric Light and Power Company's Duane Arnold Energy Center nuclear power plant located at Palo, IA. This adviser is expected to make status diagnosis in real time, thus providing the operators with more time for corrective measures.

## Neural Networks

Neural networks are a novel and fast-emerging branch of the science of artificial intelligence. Robert Hecht-Nielsen [10] defined a neural network as a "parallel, distributed information processing structure consisting of processing elements interconnected together with unidirectional signal channels called connections". Each processing element has a single output connection which "fans out" into as many collateral connections as desired. The processing element output signal can be of any mathematical type desired.

Neural networks draw interest because of the absence of a knowledge base which is the core of any expert system. Expert systems require that knowledge be specifically inserted into them about every aspect of a system that needs to be analyzed. This knowledge is stored as numerous 'if-then' logic statements that assist the system in performing a fault-tree type analysis. In the case of a nuclear power plant, this requires that every possible scenario be investigated in as much detail as possible. It also requires that the personnel developing the system understand all the processes and systems in the plant and know the significance of each sensor reading in each of the scenarios being investigated. On the other hand, ANNs do not require knowledge to be explicitly inserted into them. In fact, the designer need not have a very intimate understanding of the importance of each sensor reading. All he need to know is that certain sensors are important indicators of the health of the plant. ANNs learn the correct response from the training set during the training process, and generalize this information. The training set is the collection of input-output patterns that is used by the network to infer the functional relationship between the inputs and the outputs. Generalization is the ability to "quantitatively estimate certain characteristics or features of a phenomenon never before encountered based on similarities with things previously known" [4]. Neural networks, because of their parallel analog nature, are more noise tolerant than the conventional expert systems, and so manage to do a fairly credible job of classification under deteriorating sensor conditions. Faced with the fact that a majority of the anticipated transients at nuclear power plants have never actually occurred and data for such scenarios are obtained through computer simulation, the generalization capabilities of ANNs are especially useful for determining a solution for accident recognition [11]. If some physical aspects had been overlooked while constructing the models on which the simulations and expert systems are based, an expert system would still function on the rules built into them. However, an ANN would be able to disregard this particular

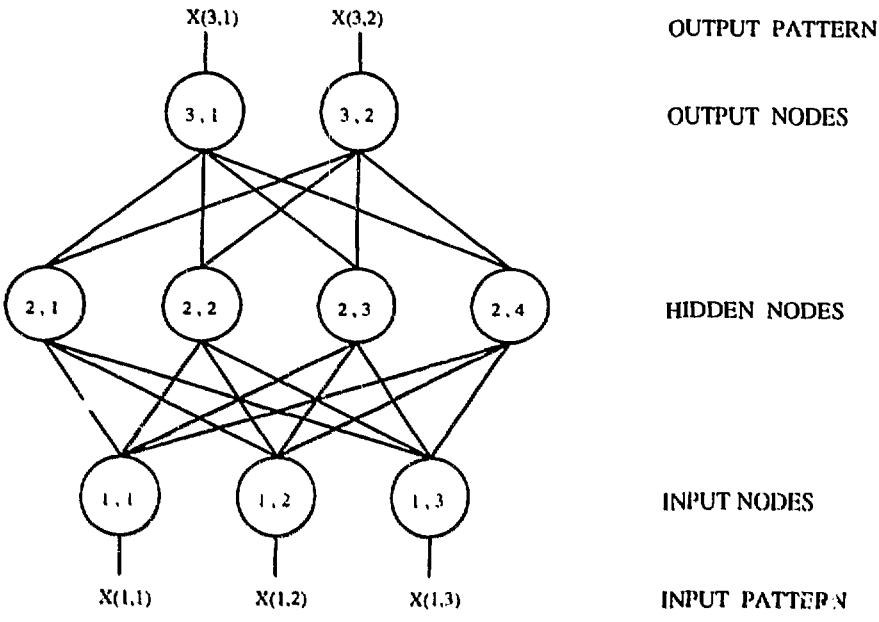


**Figure 1 : A three-layered feed-forward Backpropagation neural network**

mistake and work on the basis of other information garnered through the training process.

Artificial neural network models attempt to achieve good performance via dense interconnection of simple computational elements or nodes. Instead of performing a program of instructions sequentially as in a von Neumann computer, neural networks "explore many competing hypotheses simultaneously using massively parallel nets composed of many computational elements connected by links with variable weights" [12]. The layered feed-forward ANN consists of nodes arranged in layers, with the nodes of any layer being connected to the nodes in an adjacent layer through variable weights (see Figure 1). The nodes of a layer are connected to every node of the layers immediately above and below them but not to any node in the same layer. In such feed-forward layered networks, the first layer is the input layer where the nodes are inactive, their outputs being equal to their inputs. The last layer is the output layer. The layers in between consist of "hidden" nodes, so called because they are isolated from the outside environment. The design of a network architecture is rather arbitrary, only the number of nodes in the input and output layers being fixed by the problem at hand. The nodes used in ANNs are nonlinear and typically analog. The simplest node sums weighted inputs and passes the result through a nonlinearity function, sometimes also called the transfer function (see Figure 2).

The publication of the backpropagation technique by Rumelhart *et al.* [13] has unquestionably been the most influential development in the field of neural networks in the past decade. The learning procedure they suggested involved the presentation of a set of pairs of input and output patterns. The ANN uses the input vector to produce its own

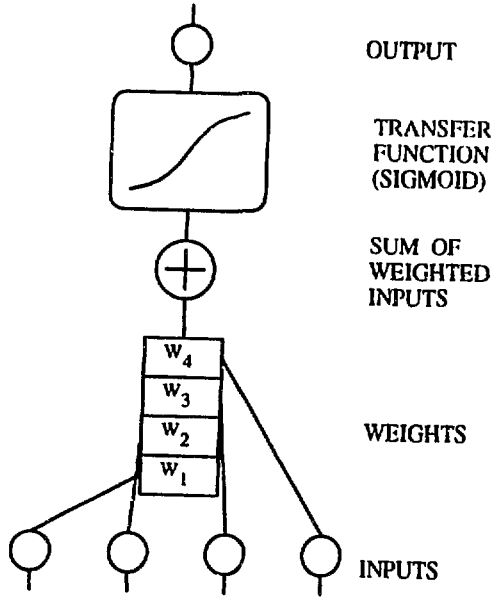


**Figure 2 : A simple node detailed**

output vector and then compares this with the expected or desired output vector. (This makes backpropagation a supervised learning method [6].) If there is no difference, no

learning takes place. Otherwise the weights are changed to reduce the difference. This process utilizes the Delta and the Generalized Delta rules. For the output nodes the error is easily calculated as the difference between the actual output and the desired output. But for the nodes in the hidden layers, it is not possible to calculate the error in this way. The correct output of the hidden nodes are not known. To assign an error to the hidden nodes we backpropagate the error of the output nodes to the hidden nodes using the very same weights that were used to propagate the error to the output nodes in the first place[7]. The Delta rule modified for the hidden nodes is called the Generalized Delta rule [7, 13].

### The Network Architecture Problem

Backpropagation networks are layered and feed-forward; they always consist of at least three layers of nodes. The middle layers, in between the input and the output layers, are isolated from the outside and are thus called hidden layers. The number of inputs and outputs are fixed by the problem at hand. The only choice of network architecture is the number of hidden nodes. This choice needs to be carefully exercised. If the hidden layer is too large it will encourage the network to memorize the input patterns rather than generalize the input into features [7, 10]. This is because the large number of nodes and weights give the network more ways to distinguish features [1], resulting in the specifics being learned better than the generalities. This reduces the network's ability to correctly classify unfamiliar patterns after training is complete. On the other hand, a hidden layer too small will drastically increase the number of iterations, and thus the computer time, required to train the network and will most likely reduce the accuracy of recall [7]. The problem of network architecture can be compared to finding the interpolation polynomial between various points [4]. An appropriate order polynomial gives a smooth curve. But as the order of the interpolating polynomial is increased, the interpolating curve tends to get chaotic, oscillating between the interpolating points.

There are no hard and fast rules for determining the optimum architecture; most rules in use at this time are empirical in nature, derived by heuristic methods. The usual approach is to start with various guesses, train all of them, and then retain the one with the best post-training characteristics [4]. This drastically increases the training time required before a network can be used to solve a problem. Problems of this kind have been instrumental in preventing the widespread use of neural networks as effective tools in solving many intractable problems [4].

The problem addressed in this paper involves status diagnostics in a nuclear power plant. The input data consisted of ninety-seven variables and the output was a combination of five booleans. The network needed to be trained a number of times with different training sets. If an appropriate architecture was needed for each trial, the problem would have assumed mammoth proportions. A lot of guesswork would have been involved, the physical problem being not so well understood. So it was imperative to come up with some systematic method that would derive the optimum, or near optimum, architecture for any given problem.

The derivative Dynamic Node Architecture (DNA) scheme was developed to skirt all the above problems and arrive systematically at a near-optimum architecture for any problem. The scheme is detailed in the next section.

### Dynamic Node Architecture Theory

The derivative Dynamic Node Architecture (DNA) scheme progresses in a systematic method to come up with the appropriate architecture for the problem. Training is commenced with a small network which won't be able to solve the problem. Typically, training was started with just one hidden node. As training progresses, the network soon reaches a plateau and cannot reduce the RMS error beyond a certain point. Now a node is added to the hidden layer. The weights connecting this node are assigned a very small value so that the addition of this node does not disrupt the network much. Training is now resumed until the network reaches another plateau. Now another node is added. This process is continued until the network reaches an RMS error value below a preselected value. This indicates that the network has learned the problem to the desired level of accuracy. Now, not all of these nodes may be necessary to recall the problem. So the hidden node with the least importance is removed from the network. The resultant network would require further training. If the deleted node had a very low level of importance, then the node had little contribution to the performance of the network. Upon continued training, the smaller network might be able to learn the problem. Then, another node (with the least importance) is deleted. This process is continued until the network is too small to learn the problem. Now nodes are added until the problem is relearnt. The process of deleting and adding nodes is continued until the algorithm starts oscillating about the optimum architecture. It is to be noted that the architecture given by this scheme may not be *the* optimum architecture but very close to it.

### Importance of a Node

While deleting nodes, we get rid of the node with the least importance. The importance of a node is a function of the network outputs. If changes in the output of a particular hidden node is instrumental in deciding the output of the network more than a similar change in the output of another hidden node, it stands to reason that the former node is more important to the "dynamic functioning of the network" [4] than the later node. The importance of the $j$th hidden node with respect to the $k$th output node is defined as [4]

$$I_{(x_j|z_k)} = E[|\delta x_{k,n}/\delta x_{j,n}|] * dx_j^{max} \qquad (1)$$

where $E[...]$ is the expectation over the entire training set and $dx_j^{max}$ is the maximum change in the output of the $j$th hidden layer node also over the entire training set. This importance function is called the derivative importance function. The derivative in the above equation, which is the change in the output of the $k$th output node due to a change in the output of the $j$th hidden node, can be evaluated by partial differentiation of the transfer function.

The above equation gives the partial importance of the $j$th hidden node with respect to the $k$th output node. If the network were to consist of more than one hidden layer, the partial importance of any of these hidden nodes with respect to any output node can be found out using the chain rule.

The total importance of the $j$th hidden node is the sum of the partial importances of that node with respect to all the output nodes. Mathematically,

$$I_{(x_j)} = \sum_{k=1}^{kmax} I_{(x_j|z_k)} \qquad (2)$$

where $kmax$ in the number of output layer nodes. In the same way, the importance of a layer can be defined as the sum of the importances of the nodes in that layer.

## Designing the Adviser

The specific problem investigated in this paper is the design of an ANN based nuclear power plant status diagnostic adviser. This adviser is expected to correctly identify and classify twenty-four distinct transients and the normal operating conditions. For a meaningful conclusion, the transients investigated needed to be much varied in nature. ANNs which can recognize such a diverse range of transients need to be able to draw information from a lot of plant variables. The choice of accidents and variables were thus very important for the project. Two major documents were consulted for the purpose. These were the *Updated Final Safety Analysis Report*,Chapter 15 : Accident Analysis [14] and the *Malfunction Cause and Effects Report* [9], both by the Duane Arnold Energy Center (DAEC). These documents describe most of the power plant transients of interest. Intensive discussions between personnel at DAEC and fellow researchers at ISU [8] resulted in a preliminary list of transients to be simulated and plant variables to be monitored [11]. From these transients, twenty-four distinct ones were selected for this work. Some of these transients had severities associated with them. In order to have the adviser detect a transient irrespective of the severity of the problem, some of these transients were simulated at different severities. This resulted in the adviser being trained to recognize twenty-four distinct transients using data from the simulation of thirty-seven scenarios. The data were obtained from the DAEC operators training simulator.

## Data Collection and Processing

Data were collected for ninety-seven plant variables at intervals of one second as the simulated transients progressed. These variables were selected from the complete list of computer points available on the simulator. They were decided to be sufficient to diagnose the transients currently being investigated. These variables covered a wide variety of plant instrumentation like pressure and temperature in the various working systems and building areas of the plant, radiation monitors, in-core flux monitors, valve monitors etc. The data consisted of the numerical values of these ninety-seven variables and the time corresponding to each set of values. The data also consisted of a boolean that indicated the onset of the transient. The data before the start of the transient related to normal plant operating conditions. The raw data, off the simulator, was in a very unusable form. Codes written at ISU were used to reformat the data in a form that could be used by neural networks. Neural networks require normalized data as input. It was decided to normalize the values of each variable based on the maximum and minimum possible values of that variable rather than on the maximum and minimum values of each variable during the simulations of the transients of interest. The former approach would make it unnecessary to renormalize data if the transients of interest increased at a future date.

## The Structure of the Adviser

The earlier approach taken in this kind of work [2, 3, 5] involved training one network to output a particular boolean if the plant were in a normal operating condition, and a different boolean for each of the transients being investigated. But this approach burdens one network with too much work. The approach taken here is to have a "root" network recognize if the plant is in a normal condition or not. If it is not, then a second network, called the "classifier" network, is trained to look at the off-normal patterns and identify the particular transient in progress.

The adviser is thus a collection of two networks. Both networks have ninety-seven input nodes. The root network has only one output node which gives an output of '0' if the plant is in a normal condition and '1' if it is not. The classifier network on the other hand needs to be able to distinguish twenty-four different transients, and so needs to output as many different booleans. This can be done with five binaries, and so, this network had five output nodes.

## Training

The values of the ninety-seven variables at any given time was expected to contain enough information to make it possible to look at them at any instance of time and diagnose the plant status [2, 3]. The training data for both the networks were chosen in an iterative manner. For the root network, in the first trial, one pattern at the beginning and one at the end of each simulation were taken to form the training set. Training was initiated with one hidden node. As the training progressed, the DNA scheme added more nodes and finally gave an optimum architecture with five hidden nodes for the first trial. This first trial had seventy-four (two from each scenario) training patterns and was trained to an RMS error of 0.10. This trained network was now used to recall on the whole length of the thirty-seven simulations, and the RMS errors of the outputs for each of the patterns was plotted out. Obviously, the network did not do a very good job of classifying all the patterns. The patterns with the worst recall errors were added to the training data set and the network from the previous trial was trained further. This process was continued until the network could detect the onset of a transient within a reasonable amount of time.

A similar approach is used to train the classifier network. Here, the normal operating conditions are not used as this network will be called upon to classify only those patterns that the root network had decided was abnormal. So the first training attempt had only thirty-seven (one from each scenario) patterns in it.

It is to be noted here that this problem is slightly different from conventional problems that are solved using neural networks. In most cases, a training set is given, and a network is trained based on that. The recall set is not known beforehand, and the trained network is then used to recall on unseen patterns. But in this case, the recall set is known from the simulations. We then try to define the training

**Table 1: The twenty-four transients used to design the adviser, the total simulation time for each transient, and the time required by the root and classifier networks to detect and diagnose the transient. The diagnosis and classification times are since the initiation of the transient.**

| No. | Scenario | Description | Simulation Time (in sec) | Root Diagnosis Time (in sec) | Classification Time (in sec) |
|---|---|---|---|---|---|
| 1 | cu10 | Reactor water clean-up coolant leakage | 149 | 8 | 107 |
| 2 | fw04a | Condensate Filter demineralizer resin injection | 200 | 25 | 40 |
| 3 | fw09a | Reactor feedwater pump trip | 181 | 17 | 53 |
| 4 | fw12c0 | Feedwater regulator valve controller failure | 182 | 7 | 81 |
| 5 | fw17a | Main feedwater line break inside primary containment | 182 | 0 | 101 |
| 6 | fw18a | Main feedwater line break outside primary containment | 192 | 5 | 78 |
| | fw18a_2 | 60% severity | 184 | 8 | 80 |
| | fw18a_3 | 30% severity | 214 | 26 | 64 |
| 7 | hp05_2 | High-pressure core injection steam supply line break (High pressure core injection room 60% severity) | 194 | 22 | 41 |
| | hp05_3 | 30% severity | 193 | 30 | 35 |
| 8 | hp08_2 | High-pressure core injection steam supply line break (Torus room) 60% severity | 321 | 35 | 72 |
| | hp08_3 | 30% severity | 365 | 39 | 111 |
| 9 | ic20scr2 | Spurious scram with effective operator action to avoid feedwater pump trip. Initial condition IC20 : 100% power, End of Cycle | 243 | 2 | 51 |
| 10 | ic20scrm | Spurious scram with no operator action Initial condition IC20 : 100% power, End of Cycle | 125 | 2 | 77 |
| 11 | ic23scrm | Spurious scram with no operator action. Initial condition IC23 : 75% power, Beginning of Cycle | 126 | 2 | 81 |
| 12 | ic24scrm | Spurious scram with no operator action. Initial condition IC24 : 100% power, Middle of Cycle | 125 | 10 | 95 |
| 13 | ms02 | Steam leak inside primary containment 100% severity | 188 | 4 | 84 |
| | ms02_2 | 60% severity | 181 | 4 | 77 |
| | ms02_3 | 30% severity | 172 | 5 | 81 |
| 14 | ms03a | Main steam line rupture inside primary containment 100% severity | 166 | 0 | 88 |
| | ms03a_2 | 60% severity | 182 | 0 | 112 |
| | ms03a_3 | 30% severity | 188 | 2 | 97 |
| 15 | ms04a | Main steam line rupture outside primary containment 100% severity | 154 | 0 | 45 |
| | ms04a_2 | 60% severity | 147 | 0 | 47 |
| | ms04a_3 | 30% severity | 137 | 0 | 47 |
| 16 | ms19ab | Spurious group 1 isolation | 182 | 20 | 83 |
| 17 | ms32 | Spurious group 7 isolation | 193 | 10 | 104 |
| 18 | rp05tc01 | Reactor protection system SCRAM circuit failure (ATWS) with alternate rod injection | 424 | 5 | 90 |
| 19 | rp5actel | Reactor protection system SCRAM circuit failure (ATWS) with failure of alternate rod injection | 520 | 5 | 115 |
| 20 | rr10 | Recirculation pump speed feedback signal failure | 262 | 17 | 104 |
| 21 | rr15a | Recirculation loop rupture (design basis Loss of Coolant Accident) 100% severity | 307 | 0 | 113 |
| | rr15a_2 | 60% severity | 301 | 0 | 113 |
| | rr15a_3 | 30% severity | 372 | 2 | 24 |
| 22 | rr30 | Coolant leakage inside primary containment 100% severity | 300 | 7 | 105 |
| | rr30_2 | 60% severity | 201 | 21 | 110 |
| 23 | rx01 | Fuel cladding (5%) failure | 185 | 11 | 18 |
| 24 | tc02 | EHC system hydraulic pump failure | 162 | 6 | 71 |

set so that it is a fairly accurate representation of the recall set. In fact, through the various trials, we force it to be so. Consequently, good generalization is extremely important in order to keep the number of patterns in the training set to a minimum. In the polynomial interpolation analogy given earlier, this is comparable to finding the smoothest interpolation curve for the given points, and then adding the points that are off the curve and finding the interpolation curve all over again.

The final architecture of the root network was 97 x 9 x 1 and for the classifier network was 97 x 26 x 5. The advantages of the DNA algorithm can be appreciated here. Suppose a conventional fixed architecture scheme were used to solve this problem. Then, for the first trial, a few architectures need to be guessed at, training done on all of them, and the architecture with the best performance used for later trials. But the training set changes, in fact increases, with each trial. So there is no guarantee that the best architecture for a given trial will also be good enough for the following trials. A fixed node architecture scheme would have been extremely inconvenient in developing the adviser.

Table 1 gives a summary of the diagnostic performance of the adviser. As can be seen, all the transients are detected within two minutes which was considered a reasonable period of time. It is possible to enhance the performance of the adviser by further training the networks.

## Conclusions

The first major conclusion from this work is the viability of a derivative importance function based Dynamic Node Architecture scheme to derive the optimum network architecture for any given problem. This work also demonstrates the viability of using neural networks to detect operational transients of a very wide variety in boiling water reactor (BWR) nuclear power plants. An ANN based adviser was successfully trained to detect and classify twenty-four distinct transients and the normal conditions.

The DNA scheme evolved a systematic method to arrive at the optimum architecture for a problem. It eliminated the guesswork associated with preset architectures used in conventional neural network training schemes. This helped in using neural networks in solving an intractable problem of the kind involving nuclear power plant diagnostics. The large amount of guesswork that would have been otherwise required to come up with the optimum architecture was eliminated.

## Acknowledgements

This work was made possible by the gracious support of the United States Department of Energy under Special Research Grant No. DE-FG02-92ER75700, entitled "Neural Network Recognition of Nuclear Power Plant Transients," and Iowa Electric Light and Power Company. Their support is greatly appreciated, however, it does not constitute an endorsement of the views expressed in this article.

## References

[1] T. Ash. "Dynamic Node Creation in Backpropagation Networks," IJCNN International Conference on Neural Networks, vol. 2. Washington D.C., June 1989. 623.

[2] E.B. Bartlett and R.E. Uhrig. "Nuclear Power Plant Status Diagnostics Using Artificial Neural Networks," Proceedings of the American Nuclear Society Meeting on Frontiers in Innovative Computing for the Nuclear Industry. September, 1991. 644-653.

[3] E.B. Bartlett and R.E. Uhrig. "Nuclear Power Plant Status Diagnostics Using Artificial Neural Networks." Nuclear Technology 97 (March, 1992): 272-281.

[4] E.B. Bartlett and Anujit Basu. "A Dynamic Node Architecture Scheme for Backpropagation Neural Networks." Intelligent Engineering Systems Through Artificial Neural Networks, Eds. C.H. Dagli, S.R.T. Kumara, and Y.C. Shin. New York: ASME Press, 1991. 101-106.

[5] Anujit Basu. "Nuclear power plant status diagnostics using a neural network with dynamic node architecture." M.S. Thesis, Iowa State University. 1992.

[6] M. Caudill. "Neural Networks Primer, Part 1." AI Expert (Dec, 1987): 46-52.

[7] M. Caudill. "Neural Networks Primer, Part 3." AI Expert (June, 1988): 53-59.

[8] Duane Arnold Energy Center simulator complex employees Don Vest, Cring Hunt and Dan Berchenbriter. Personal discussions and correspondence. Iowa Electric Power and Light Company, Palo, IA. 1991-1992.

[9] J. Gould. Malfunction Cause and Effects Report, Task no. 06000004. Cedar Rapids, Iowa: Duane Arnold Energy Center, 1991.

[10] R. Hecht-Nielsen. "Theory of the Backpropagation Neural Network." Proceedings of the International Joint Conference on Neural Networks, vol. 1, 1989. 593-605.

[11] T.L. Lane. "The Importance of Input Variables to a Neural Network Fault-Diagnostic System for Nuclear Power Plants." M.S. Thesis, Iowa State University, 1991.

[12] R.P. Lippmann. "An Introduction to Computing with Neural Nets." IEEE Acoustics Speech and Signal Processing Magazine 4 (April, 1987): 4-22.

[13] E.D. Rumelhart, G.E. Hinton, and R.J. Williams. "Learning Internal Representations by Error Propagation." Parallel Distributed Processing: Explorations in the Microstructure of Cognition Vol. 1, Cambridge, Mass.: The MIT Press, 1986. 318-362.

[14] "Accident Analysis" In Updated Final Safety Analysis Report. Cedar Rapids, Iowa: Duane Arnold Energy Center, 1984.