A NEURAL NETWORK APPROACH TO THE DETECTION OF INCIPIENT FAULTS ON POWER DISTRIBUTION FEEDERS

Sonja Ebron
Student Member IEEE

David L. Lubkeman
Member IEEE

Mark White
Member IEEE

Electric Power Research Center
College of Engineering
North Carolina State University
Raleigh, NC 27695-7911

ABSTRACT

A high-impedance fault is an abnormal event on an electric power distribution feeder which can not be easily detected by conventional overcurrent protective devices. This paper describes a neural network strategy for the detection of this type of incipient fault. Neural networks are particularly well-suited for solving difficult signal processing and pattern recognition problems. An optimization technique allows a network to "learn" rules for solving a problem by processing a set of example cases. The data preprocessing required to set up the training cases and the implementation of the neural network itself are described in detail. The potential of the neural network approach is demonstrated by applying the detection scheme to high-impedance faults simulated on a model distribution system.

Keywords: High-Impedance Faults, Neural Networks

INTRODUCTION

Description of High Impedance Faults

A high-impedance fault (HIF) can be defined as an abnormal event on a primary distribution feeder which can not be easily detected by modern protective devices. Most HIFs involve current levels much less than those of bolted (short-circuit) faults, and can not be reliably detected by conventional overcurrent devices. These faults often exhibit arcing phenomena when no solid return path for current is available, resulting in fault currents with noticeable high-frequency components. Unfortunately, this same behavior can result from other events such as capacitor switching and transformer tap changing. In addition to service interruption, HIFs can result in fires, electric shock, and the ensuing utility liability. Even though the actions taken to detect these faults may be identical to those of bolted fault clearance, the motivation for HIF detection is improved public safety rather than system protection [1].

Previous Research Efforts

The most common detection scheme for HIFs involves lowering the settings on overcurrent protection devices. However, since most HIFs produce current levels indistinguishable from normal loads, this method can result in unnecessary service interruptions. Another simple technique calls for lowering the settings on ground relays in hopes of detecting abnormal ground currents. This is obviously implausible for multiply grounded systems and unreliable for feeders with a normally high degree of imbalance [2]. Researchers at Westinghouse Electric Corporation and Pennsylvania Power & Light have recommended the use of a Ratio Ground Relay[3]. Based on the concept that the amount of load imbalance on a given feeder is invariant over a small time interval, these researchers suggest computing the ratio of neutral current to positive-sequence current (called RGR) as a fault indicator. Devices for fault detection using this method have been in use since 1982 [4].

Since changes in feeder load occur slowly, another detection scheme seeks to measure the rate of change in current. Again, most HIFs do not change current levels at all, rendering this scheme unreliable [5]. A more sophisticated technique involves finding a *Chi*-squared test statistic, using 60-Hz sequence components and harmonics, to detect HIFs [6]. Like the RGR method, this technique seeks a change in feeder imbalance to indicate the presence of a fault. After implementation, the method failed in 60% of cases [7]. Microprocessor-based implementation in 1985 produced slightly better results [8].

Another scheme uses increases in frequency components near 60 Hz to detect faults. These 60-Hz parameters normally dominate the frequency spectrum, so it is difficult to detect meaningful changes in them [9]. A similar method relies on analysis of the third harmonic, which is slightly more sensitive to change, to signal the occurrence of an HIF [10]. None of these techniques has had sustained and reliable success in practice.

Yet another suggested scheme is to monitor several line segments and buses in an effort to identify low-current or low-voltage conditions downstream from a fault [11]. This technique, though undoubtedly reliable, is precluded by the high cost of metering. Possibly the most successful method involves detecting increases in 2–10 kHz frequency components [12]. Both laboratory and staged-fault tests indicate that arcing waveforms contain substantial high-frequency elements. However, high-frequency signal grounding by capacitors is a major stumbling block to HIF detection by this method [13]. A knowledge-based (expert) system has recently been developed which attempts to detect faults by monitoring frequency changes [14]; it remains untested as of this date. Clearly, no technique based on only one HIF characteristic is suitably reliable for detection of these types of faults.

89 TD 377-3 PWRD A paper recommended and approved by the IEEE Power System Relaying Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1989 Transmission and Distribution Conference, New Orleans, Louisiana, April 2 - 7, 1989. Manuscript submitted October 7, 1988; made available for printing January 19, 1989.

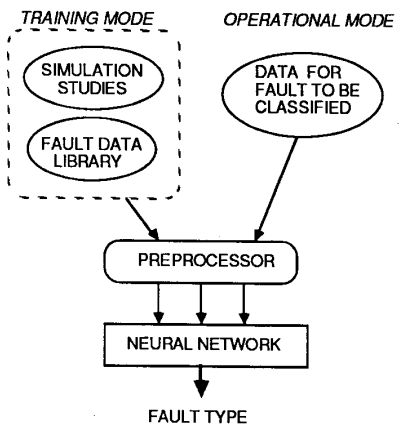

Figure 1: Fault Detection by a Neural Network

Neural Network Processing Strategy

The strategy presented here for detecting high-impedance faults is motivated by recent developments in parallel distributed processing, or *connectionist* theory. Though much promise has been shown by expert systems technology, a fundamental limitation is the *a priori* formulation of "rules" for the device or system application. In other words, an expert system solution to a given problem presumes that some human expert can solve the problem. This is not a serious limitation for many problems, since heuristic rules can be derived in most cases; however, expert systems cannot be applied to problems for which little human expertise exists, as is the case with high-impedance faults.

Connectionist theory is an emerging science which seeks to model the parallelism and interconnectedness of the human brain. A connectionist model has been developed which has the potential, through pattern recognition and analysis, of deriving rules for certain applications. This model, called an *artificial neural network*, has shown dramatic results in "learning" patterns of speech, handwriting, logic gate mappings, and other applications, given only examples of the patterns [15]–[19]. Recent advances in applications of the theory range from visual and adaptive pattern recognition to motion detection. In fact, VLSI implementation of a neural network is the focus of current research. The parallelism inherent in neural networks gives them more speed than traditional serial computers. Also, the distributed nature of a neural network makes it robust, in that loss of a small portion does not substantially decrease its level of performance.

Neural network processing is distinguished from signal processing by the capability for *nonlinear* interpolation. Modern signal processing techniques perform linear interpolation of input signals. That is, when unfamiliar input is presented to an adaptive filter, the corresponding output lies on some hyperplane (a line in two dimensions) between output for two known inputs. In contrast, when a neural network is fully trained, it is capable of mapping an unfamiliar input vector to an arbitrary surface. In other words, a neural net generalizes instead of performing "table-lookup".

High-impedance faults result in currents that exhibit certain characteristics. By careful observation of current waveforms and frequency spectra, some characteristics may be identified as peculiar to high-impedance faults, such as large transients or ground return currents. To implement a neural network scheme, a distribution feeder is simulated by computer to provide samples of substation phase currents, with which analysis of fault currents is performed. A library of actual current measurements associated with various faults and switching transients could also be used. A *preprocessor* is used to produce any current characteristics of interest by performing Fourier transforms, measuring transients, and computing other parameters. A neural network is appropriately configured and "trained" with these parameters from simulated or measured waveforms. The network is then capable of processing feeder current data in a real-time operating mode and based on the past examples presented to it, determine whether a high-impedance fault is present. A diagram of the neural network strategy is shown in Figure 1.

INTRODUCTION TO NEURAL NETWORKS

Description of a Neural Network

A neural network consists of a set of connected nodes and a propagation rule. A general node model is represented in Figure 2. The node, or *neuron*, receives its input through weighted links. This input may come from other nodes in the network or from outside stimuli. An activation function, usually a summation, acts on the input; the node's internal bias is then added to the summed and weighted input. The result is called *node activation*. The node's output is determined by an output function, which responds to the activation. An example is the S-shaped sigmoid shown in Figure 2. The propagation rule of the node consists of its activation and output functions. The node's output travels along the links, or *synapses*, either to other nodes or to the output of the system.

A neural network is simply a layered collection of these nodes. The nodes are connected by links of varying weight. An n-node network with a given propagation rule is fully described by an n-dimensional internal bias vector and an n-dimensional square weight matrix.

Many types of networks exist, but the research described here focuses on feedforward layered networks, with each node's activation and output determined by summation and sigmoid functions, respectively. In feedforward networks, all input is received on one layer, and the resulting signals propagate forward, one layer at a time, until the signals reach the last

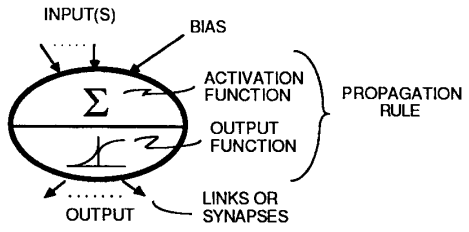

Figure 2: Neuron Model

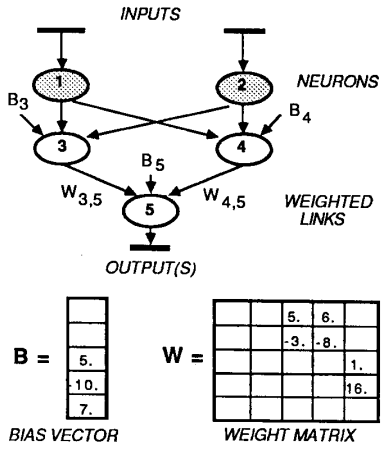

Figure 3: Feedforward Layered Network

layer. An example of such a neural network is shown in Figure 3, with a corresponding bias vector and weight matrix.

This particular network contains five nodes in three layers, and only six weighted synapses are required to fully connect the network. Each node on the first layer is connected to each node on the second, and each node on the second layer is connected to the node on the third. The last layer is referred to as the *output layer*, since the network's output is the response of neurons on this layer. The first layer is referred to as the *input layer*. Nodes on this layer differ from others in a feedforward network; they are strictly linear in that the output of a node is simply its one allowable input. Nodes on the input layer have no internal biases. All other layers in the network are referred to as *hidden layers*, since they are not accessible to the outer environment.

Operation of a Neural Network

The operation of a neural network consists of the presentation of a set of inputs and subsequent propagation of this input through the network. The 5-node neural network shown in Figure 3 can be used to illustrate the *forward propagation* of input signals. In this example, inputs x_1 and x_2 are presented to Node 1 and Node 2, respectively. Since nodes in the input layer have linear propagation rules,

$$y_1 = x_1 \quad \text{and} \quad y_2 = x_2 . \tag{1}$$

Activation for any node on the other layers is the sum of the node's internal bias and weighted outputs passed to it; that is,

$$x_3 = w_{13}y_1 + w_{23}y_2 + b_3$$
and
$$x_4 = w_{14}y_1 + w_{24}y_2 + b_4 . \tag{2}$$

Output for these nodes is determined by a sigmoid function, such that

$$y_3 = \frac{1}{1+e^{-x_3/T}} \quad \text{and} \quad y_4 = \frac{1}{1+e^{-x_4/T}}, \tag{3}$$

where T is the chosen "temperature", or degree of nonlinearity, of the sigmoid function. For sigmoid function output, note that

$$\lim_{x_n \to -\infty} y_n = 0 \quad \text{and} \quad \lim_{x_n \to \infty} y_n = 1 .$$

Finally,

$$x_5 = w_{35}y_3 + w_{45}y_4 + b_5 \tag{4}$$
and
$$y_5 = \frac{1}{1+e^{-x_5/T}} . \tag{5}$$

Hence, y_5 is the network's response to inputs x_1 and x_2. The method is called forward propagation because node responses on a given layer can only be calculated after those on the preceding layer are found.

In general, then, forward propagation consists of passing weighted and summed input signals through a chosen nonlinearity. It presumes knowledge of the network's bias vector and weight matrix. Again, once activation and output functions are chosen, a neural network is completely described by its weights and biases. Since a given neural network solves a specific problem, or function, finding weights and biases for the network is equivalent to finding the input/output relationship that describes the function. Thus, neural networks are especially appropriate and powerful when used to find relationships that are difficult to describe explicitly, because weights and biases can represent a given function.

Training of a Neural Network

In order for a neural net to learn the "rules" for solving a problem, data sets describing the problem must be given. These data sets consist of input vectors and desired, or target, output vectors for each input vector. A full *training set* for a neural network describes the full range of expected inputs and associated, desired outputs.

The neural net used in this research is trained by a learning rule called the *Back Propagation Learning Algorithm* [20], alternately known as the Generalized Delta Rule. Using the network of Figure 3 as an example, small random weights are first assigned to the network links. These weights are

$$w_{ij} \text{ and } b_j \text{ for } i = 1, \ldots, 4 \text{ and } j = 3, \ldots, 5 ,$$

with the weights of nonexistent links (w_{15}, for instance) set to zero. To activate the network, an input vector p, consisting of elements x_{1p} and x_{2p} is presented. Forward propagation is performed on this input, resulting in a response, y_{5p}. The output error, defined as

$$E_p \triangleq \frac{1}{2}(t_p - y_{5p})^2 , \tag{6}$$

is then calculated, where t_p is the target, or desired output. The objective of the learning algorithm is to minimize this output error for each training vector. The error must be allocated to all nodes in the network (except input nodes) so that weights connected to the nodes can be changed.

Let

$$\delta_{jp} \text{ for } j = 3, \ldots, 5$$

represent allocated node errors. The output error, E_p, is locally minimized when its gradient, $-\dot{y}_{5p}(t_p - y_{5p})$, is zero. Thus, assigned node error for the output node must be proportional to $\dot{y}_{5p}$. Any change in the objective function should be in the

direction of steepest descent (or negative gradient), so that assigned node errors, in general, are proportional to

$$\dot{y}_{5p} = \frac{1}{T}\frac{e^{-x_{5p}/T}}{\left(1+e^{-x_{5p}/T}\right)^2} = \frac{1}{T}y_{5p}(1-y_{5p}). \qquad (7)$$

This means that the assigned error for the output node is simply

$$\begin{aligned}\delta_{5p} &= \dot{y}_{5p}(t_p - y_{5p}) \\ &= \frac{1}{T}y_{5p}(1-y_{5p})(t_p - y_{5p}). \end{aligned} \qquad (8)$$

The assigned error for a node on the hidden layer depends on weights connecting the node to the output node(s) and on the assigned error of the output node(s). Hence,

$$\delta_{3p} = \frac{1}{T}y_{3p}(1-y_{3p})(\delta_{5p}w_{35})$$

and

$$\delta_{4p} = \frac{1}{T}y_{4p}(1-y_{4p})(\delta_{5p}w_{45}). \qquad (9)$$

No error is attributable to input nodes, whose propagation rules are linear. In this algorithm, node errors at the output layer must be calculated before those at the hidden layer can be determined; that is, node errors are propagated backwards.

Weights are changed to reduce the output error. The changes in weights are

$$\Delta w_{ijp} = \eta\delta_{jp}y_{ip} \quad \text{and} \quad \Delta b_{jp} = \eta\delta_{jp} \qquad (10)$$
$$\text{for } i = 1,\ldots,4,\ j = 3,\ldots,5,$$

where η is the *learning rate*, a constant between zero and one. Weights are *not* changed at this point, however. First, weight changes are calculated for each input/output vector, then new weights are computed by

$$w_{ij}^{k+1} = w_{ij}^{k} + \sum_p \Delta w_{ijp}^{k+1}$$

and

$$b_j^{k+1} = b_j^k + \sum_p \Delta b_{jp}^{k+1} \qquad (11)$$

$$\text{for } i = 1,\ldots,4,\ j = 3,\ldots,5.$$

The entire process is repeated with these new weights, starting with the first training vector. The Back Propagation Learning Algorithm consists of repeatedly passing the training set through the neural net until its weights minimize the output errors over the entire set. At this point the objective function,

$$E = \sum_p E_p, \qquad (12)$$

has reached a local minimum.

DATA PREPROCESSING

Need for Preprocessing

The neural network approach to the detection of high-impedance faults consists of three basic tasks — collecting sets of sampled, processed feeder line currents, using these sets to train a neural network by error propagation, and testing the network on separate sets of processed line currents. The preprocessor is an integral part of this strategy. Training cases were developed using a computer simulation of high-impedance faults on a distribution feeder. The simulation generates samples of current waveforms on three phases. Since these samples are taken at a high frequency, it is impractical to use them directly as input for a neural network. Hence, certain characteristics of the waveforms must be identified and reduced to quantitative form in order for the network to distinguish between normal and abnormal feeder operation.

Simulation of a Distribution Feeder

Training cases for a typical 12 kV distribution system are generated through the use of the Electromagnetic Transients Program (EMTP) [21]. EMTP readily handles switch closings or openings at specified or random times; thus, the program is ideal for fault simulation and transient analysis. Since a successful HIF detection scheme must be able to distinguish between normal current harmonics and the possible arcing from a fault event, harmonic sources are included in the simulation. Several induction motors fed from a power electronic circuit are set up to provide this effect. Capacitors are also included in the feeder model to simulate switching events that must be distinguished from faults. Capacitors may also trap and ground any fault signals that propagate back to the substation. Switches connected from distribution transformers to high-impedance elements (on the order of load impedances) are used to represent the incipient faults.

A computer program is used to randomly generate the input data in the EMTP format, execute EMTP, calculate chosen parameters of interest from the resulting current samples, and then repeat this process until the number of cases specified by the user has been collected [22]. All EMTP cases vary from a fixed operating point (base case) only in the frequency of fault arcing (0–10 kHz), size of passive loads (±20%), size of line and load fault impedances (±20%), size of induction motor loads (±20%), switching times (0–169 msec) and locations, and diode conduction voltages (±25%). In all other respects, generated cases are identical to the base case. Four types of cases are generated:

1. normal load switching cases,
2. normal load and capacitor switching cases,
3. high impedance fault cases with load switching,
4. high impedance fault cases with load and capacitor switching.

Selection of Neural Network Input

Certain features of high-impedance faults may be observable by inspecting the waveforms associated with such faults. Since a typical distribution feeder includes constantly changing loads, tap-changing transformers, switched capacitors, and power-electronic loads, then various transients associated with harmonic current propagation, capacitor switching, load switching

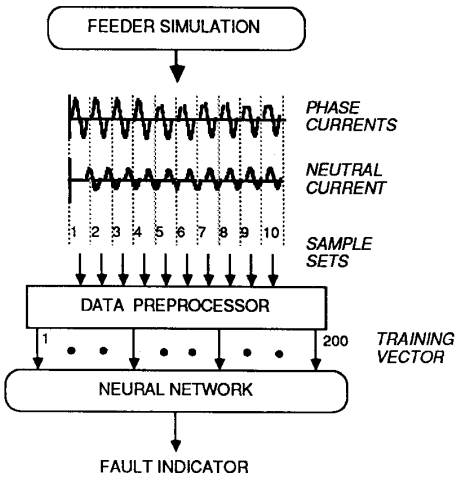

Figure 4: Preprocessing of Fault Simulation Data

and other time-varying phenomena will normally be observed. Hence, information obtained by inspection of current waveforms must allow differentiation between a high-impedance fault and a normal transient event.

The preprocessor extracts pertinent information on the state of the feeder over 10 cycles of operation. The current samples used to train the network are first separated into sets of one cycle each. For each set, 20 parameters are computed to represent the feeder over 1 cycle of operation. These parameters are

- the peak value of transient current over the three phases,
- the current (on the corresponding phase) before, and immediately after, the largest transient occurs,
- the number of transients falling within 75% of the maximum perturbation,
- the magnitude of positive sequence current,
- the amount of imbalance between phases (using the Ratio Ground Relay method [3]),
- first, third, and fifth harmonic components of the neutral current (as suggested by [10]),
- frequency activity in the neutral current over ten sections of its frequency spectrum (as suggested by [12]).

Several of these quantities, such as positive sequence current, are for reference purposes only; they allow the neural network to compare transient current levels and other parameters and find *relative* amounts of disturbance. Figure 4 shows the arrangement for including the preprocessor in the fault detection scheme.

Collection of a Training Set

By repeatedly analyzing EMTP test cases and processing the resulting data, the preprocessor collects a training set for the neural net. Each vector in the training set represents 10 cycles of feeder operation; it contains 200 inputs and one output indicating whether the feeder is undergoing normal operation or a fault. The target is 0.1 for normal feeder operation and 0.9 for a fault. These values are chosen because of the limits of neural network output (sigmoid) functions, which are suited only to logic gate mappings.

Though the output of each training case fits within the tolerable range of a sigmoid function, elements of the corresponding input may fall anywhere on the positive real line. If the range of values for a given input parameter includes numbers on the order of one or greater, nodes in the network quickly become saturated, and learning is prohibited. Large input values initially place each node output near one; node outputs are thereby predetermined. Since the learning algorithm normally pushes node responses to a saturation level, the network is "tricked" into believing the initial weights are accurate.

Thus, the input range for each vector in the training set must be contracted (or expanded) to the range $[0, 1]$. For example, if M is the number of training cases, the first element of every training vector constitutes a collection of signals, $\{\tilde{x}_{j1}, \ldots, \tilde{x}_{jp}, \ldots, \tilde{x}_{jM}\}$, for the j^{th} node in the neural net's input layer. The values of these signals range from $\min_j$ to $\max_j$. The input space $[\min_j, \max_j]$ is transformed to $[0, 1]$ by letting

$$x_{jp} = \frac{\tilde{x}_{jp} - \min_j}{\max_j - \min_j} \tag{13}$$

for the p^{th} training case. By this transformation, the training set for high-impedance fault detection becomes the representation for a logic gate function. Since the transformation is nonlinear, it changes the direct relationships between inputs in each vector. Relationships between transformed inputs still exist, however; they are simply redefined for presentation to the neural net. This additional preprocessing of data can be thought of as an external hidden layer of the network.

Thus, a training set is collected by repeatedly generating EMTP cases and preprocessing the resulting data. For this particular study, approximately 50% of the cases generated have load changes only, 20% have both load changes and capacitor switching, 20% have load changes and a fault, and 10% have load changes, capacitor switching, and a fault. Each vector in the training set represents one EMTP case. Again, it contains 200 inputs and one output indicating whether the case simulates normal operation or a fault. Since a preprocessing routine randomly decides when to generate a fault case, it is able to specify the training set output, or target, for each case.

FAULT DETECTOR IMPLEMENTATION

Overview of Detection Strategy

Simulation of an HIF fault detector involves training a neural network with processed current samples collected from the feeder, then testing the network with forward propagation of additional samples. A computer program that implements the Back Propagation algorithm receives a training set and produces an initial weight matrix of appropriate size. Initial elements of $[w_{ij}]$ and $[b_j]$ consist of small random values. In addition to an input/output vector for each case, the training set contains a configuration for the neural net. The number of layers and the total number of nodes are specified, as well as the number of nodes in each layer. The number of cases available for training is also given.

The training algorithm forms a neural net with 200 input nodes and random weights. For each input/output vector in the training set, the algorithm computes changes in the network's weights that minimize the difference between the

target output and that resulting from forward propagation of the vector's input signals. The entire training set is repeatedly passed through the neural net until the weights minimize output errors over all input/output vectors. Since these vectors represent randomly-collected line currents from the simulated feeder, the weights should be capable of producing an output, for new input signals, that is *similar* in value to targets specified for similar input vectors in the training set.

In order to test the effectiveness of the neural network strategy, the weights found during training are tested with unfamiliar input vectors; these vectors are collected by the preprocessor in the same manner as those in the training set. When a 200-element vector is applied to the network, the signals propagate forward until an output results. The preprocessor places the targets for collected input vectors in a separate file, so the network's output can be checked. If the output is incorrect, the input vector is used to retrain the network, thereby obtaining a revised set of weights. Testing and retraining are performed on-line, so the neural net is in operation at this stage. Hence, the neural network can learn from its mistakes.

Example of Preprocessing

The preprocessor routines are performed on 10 sets of data representing one EMTP case. The preprocessor thereby extracts pertinent information on the state of the feeder over 169 ms of operation. This time division of the data helps to identify the onset of switching. Each set serves as a 1-cycle "window" that may distinguish normal feeder operation from a fault event when compared to other sets from the same case. The 15,360 current samples produced by EMTP are first separated into 1-cycle "windows", each with 512 samples per phase. For each window, 20 parameters are computed to represent the feeder over 1 cycle of operation. Thus, the 15,360 samples produced by EMTP are reduced by the preprocessor to 200 inputs for the neural net. These parameters are

1. window number (1–10),
2. peak value of transient current over three phases,
3. current before the largest transient occurs,
4. current after the largest transient occurs,
5. number of transients greater than 25% of peak transient,
6. magnitude of positive sequence current,
7. amount of imbalance on the feeder (RGR),
8. fundamental component of neutral current,
9. third harmonic component of neutral current,
10. fifth harmonic component of neutral current,
11. energy over the 0.95–2.37 kHz range,
12. energy over the 2.37–3.79 kHz range,
13. energy over the 3.79–5.21 kHz range,
14. energy over the 5.21–6.63 kHz range,
15. energy over the 6.63–8.05 kHz range,
16. energy over the 8.05–9.47 kHz range,
17. energy over the 9.47–10.89 kHz range,
18. energy over the 10.89–12.31 kHz range,
19. energy over the 12.31–13.73 kHz range,
20. energy over the 13.73–15.15 kHz range.

Table 1: An Example of Preprocessing Results

	Set #1	Set #2	Set #3	Set #4	Set #5	Set #6	Set #7	Set #8	Set #9	Set #10
1	1	2	3	4	5	6	7	8	9	10
2	15.45	7.71	36.13	7.34	7.43	15.19	89.30	92.40	88.76	88.65
3	7.23	3.85	16.06	3.67	3.71	7.59	44.65	46.25	44.38	44.32
4	0.00	1.02	1.39	1.08	0.96	0.83	16.93	1.48	2.01	7.91
5	156	341	7	334	339	50	257	371	360	380
6	123.1	127.4	127.6	114.2	126.4	126.6	86.88	140.9	141.1	140.9
7	0.13	0.13	0.09	0.07	0.06	0.06	0.31	0.23	0.23	0.23
8	22.77	22.61	14.97	10.99	11.07	11.19	36.30	45.39	44.83	44.74
9	0.14	0.18	3.40	0.10	0.06	0.13	7.06	1.20	0.47	0.53
10	0.14	0.06	0.99	0.07	0.05	0.13	2.07	0.76	0.13	0.17
11	0.20	0.15	4.84	0.23	0.13	2.64	8.44	3.12	0.51	0.83
12	0.22	0.15	2.60	0.15	0.12	1.94	4.43	1.91	0.38	0.70
13	0.60	0.46	2.38	0.48	0.47	1.22	3.10	1.38	0.56	0.84
14	0.64	0.19	2.00	0.25	0.23	0.37	2.51	1.08	0.39	0.78
15	0.37	0.36	1.33	0.37	0.39	0.49	13.26	7.31	6.80	7.11
16	0.71	0.63	1.32	0.65	0.67	1.01	3.26	1.06	0.66	1.13
17	0.66	0.20	1.08	0.29	0.23	0.46	2.36	1.30	0.65	1.30
18	0.38	0.29	0.81	0.36	0.35	0.34	2.51	1.53	1.02	1.67
19	0.27	0.30	0.94	0.29	0.28	0.40	4.04	3.67	2.62	4.04
20	0.08	0.10	0.84	0.06	0.06	0.57	16.52	21.46	21.09	19.19

An EMTP simulation of one event, and the subsequent preprocessing, results in the 200 elements shown in Table 1. The 20 parameters listed above are calculated for each of the 10 1-cycle windows. In the table, values representative of normal feeder operation are shown in columns 1–6, which cover 101.4 ms of time. In window 7, however, marked increases in several frequency parameters clearly indicate the onset and presence of a fault. In addition, the transient sizes, current levels, and feeder imbalance all show changes when compared to earlier cycles. These changes persist throughout the case simulation.

The 200 elements in the table form one input vector for a neural network. The associated output vector consists of one element, which is a target value of 0.9. By repeatedly executing EMTP data sets and reducing the resulting samples to 200-element input vectors, the preprocessor collects a training set for a neural network.

Testing of Neural Network Approach

A set of 50 training cases generated by the EMTP program were used to test the neural network strategy. High-impedance faults were simulated in roughly 30% of these cases. The resulting input/output vectors formed a training set for high-impedance fault detection. The neural network was configured with 2 hidden layers and 801 nodes. There were a total of 200 input nodes and one output node. The number of nodes on the first hidden layer was 200, while 400 nodes lay on the second. The network required 120,400 weights and 601 biases for full interconnection. Using the Back Propagation algorithm, the network learned the patterns presented in 38 iterations of the 50-vector training set.

The weights generated during training were then tested with unfamiliar input vectors, 15% of which represented high-impedance faults. These vectors were collected by the preprocessor in the same manner as those in the training set. Unlike the training process, detection required no iteration and only one input vector was known to the network at any given time. Again, since the range of inputs did not conform to that of a logic gate, a transformation of each input vector was required. The transformation was accomplished by passing the

vectors

$$\begin{bmatrix} \min_1 \\ \vdots \\ \min_j \\ \vdots \\ \min_{200} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \max_1 \\ \vdots \\ \max_j \\ \vdots \\ \max_{200} \end{bmatrix},$$

where j is the index of an input node, to the high-impedance fault detector. (These vectors were collected while training the detector.) The range of input vector $\{\tilde{x}_1, \ldots, \tilde{x}_{200}\}$ was then reduced, or expanded, by letting

$$x_j = \frac{\tilde{x}_j - \min_j}{\max_j - \min_j}. \tag{14}$$

Since the output range was fixed at $[0,1]$ by sigmoid functions in the network, a fault was indicated during on-line operation when the network's response to input surpassed 0.5. In that event, the detection routine caused the user's terminal to sound an alarm and reported a "certainty factor", analogous to a probability, that a fault had occurred. This factor was simply the output of the neural network. The program temporarily suspended execution at this point to retrain (if necessary) and to allow the user to end feeder observation. When the detector was in error, the routine changed weights in a manner that would allow it to respond to the same input correctly. If continued operation was desired after a fault was detected, the routine resumed propagation of input signals until another fault occurred.

A detection set, consisting of 100 input vectors representing feeder operation, was collected by the preprocessor and then passed to the detector. Of the 15 HIF cases present in this set, the network responded correctly to 10, though on-line retraining undoubtedly helped to "fine-tune" its responses. Of the remaining 85 cases, the network mistakenly caused an alarm in 17. Interestingly, 19 of the 100 cases involved capacitor switching in the absence of a fault, so the network apparently had difficulty distinguishing between the two high-frequency switching activities.

A more effective evaluation of a neural network approach to the detection of high-impedance faults will require actual field data. The simulations generated during the course of this work were primarily designed to test whether a neural network approach was technically feasible. Hence the current waveforms generated were only representative of the types of waveforms that could be generated by a high-impedance fault. Using the EMTP program to more accurately simulate the high-impedance fault phenomenon would be a separate research project in itself, given that a suitable fault model could be incorporated into the program.

A fault data library obtained from measurements in the field would be used to train the network in a practical implementation of a high-impedance fault detector. This library would include cases corresponding to high-impedance faults and normal system transients which have characteristics similar to such faults. The neural network fault detector can then be trained to identify high-impedance faults, taking into account the special characteristics of each feeder. If fault recording capabilities existed, then the cases that the neural network fault detector failed to identify correctly could be used to retrain the network.

The parameters used to characterise the current waveforms and the number of data windows which were used to show time-varying effects need to be reexamined in an actual implementation. Many of the parameters used in this research may not be needed for an effective fault detection system. The parameters which are used to create the training set for the neural network are designed to help discriminate between a high-impedance fault and a normal system transient. An opportunity exists for creating more effective input parameters which will increase the detection efficiency of the neural network.

Other Practical Considerations

Theoretically, neural networks can find a representation for any observable phenomenon. Examples of some problem, taken in its environment, can be used to find appropriate network weights for its solution. However, several problems exist in practice. First, data describing the phenomenon may be difficult or expensive to obtain. In that case, simulation of the environment is necessary, but the resulting error may lead to a set of inaccurate weights. Also, the data must describe the state, or condition, of the system with respect to the phenomena being observed. For example, relevant characteristics of high-impedance faults are evident in successive line current samples, but they are *not* apparent in, say, energy consumption levels over time. Appropriate state variables must be found to describe a given system.

Second, samples of the environment's state over time may require an enormous number of input nodes which may, in turn, require very large hidden layers. Though no constraints on the number of network nodes (or weights) exists in the training algorithm, implementation by computer places a limit on the network's size. This problem can be avoided by the development of a data preprocessor, which transforms state samples into concise and meaningful parameters of the system. There is no guarantee, however, that representative features can be found. If these characteristics are found, it may then be impossible to quantify them.

CONCLUSIONS

High-impedance faults cannot be reliably detected and cleared by conventional protective devices. A neural network strategy for detecting these faults has been presented. This approach consists of collecting samples of substation current during normal and abnormal feeder operation, then using these samples to "teach" a neural network the rules for fault detection. The learning capability utilized in a neural network approach makes it possible to adapt partially-trained fault detectors to individual feeders. Although a great deal of research still needs to be done, the neural network approach shows great potential as a more effective strategy for detecting such incipient faults.

Neural networks have shown astounding pattern recognition properties. Thus far, however, the greatest successes of connectionist theory have been in communications and signal processing applications. In the next few years, VLSI implementations of neural networks will make it possible to deploy this technology on a massive scale. Undoubtedly, many more applications will be found for this new technology, including presently intractable problems in power systems research.

REFERENCES

[1] M. Aucoin, "Status of High Impedance Fault Detection", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-104, No. 3, March 1985, pp. 638–644.

[2] J. Carr, "Detection of High Impedance Faults on Multi-Grounded Primary Distribution Systems", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-100, No. 4, April 1981, pp. 2008–2016.

[3] H. Calhoun, M.T. Bishop, C.H. Eichler, R.E. Lee, "Development and Testing of an Electromechanical Relay to Detect Fallen Distribution Conductors", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-101, No. 6, June 1982, pp. 1643–1650.

[4] R.E. Lee and M.T. Bishop, "A Comparison of Measured High Impedance Fault Data to Digital Computer Modeling Results", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-104, No. 10, October 1985, pp. 2754–2758.

[5] J. Carr and G.L. Hood, "High Impedance Fault Detection on Primary Distribution Systems", Report for the Canadian Electrical Association, November 1979, p. 43–44.

[6] "Detection of High Impedance Faults", EPRI Report EL-2413, Prepared by Power Technologies, Inc., June 1982.

[7] "Implementation of a High-Impedance Fault Detection Algorithm", EPRI Report EL-4022, Prepared by Power Technologies, Inc., May 1985.

[8] S.J. Balser, K.A. Clements, D.J. Lawrence, "A Microprocessor-Based Technique for Detection of High Impedance Faults", *IEEE Transactions on Power Delivery*, Vol. PWRD-1, No. 3, July 1986, pp. 252–258.

[9] B.M. Aucoin and B.D. Russell, "Detection of Distribution High Impedance Faults Using Burst Noise Signals Near 60 Hz", *IEEE Transactions on Power Delivery*, Vol. PWRD-2, No. 2, April 1987, pp. 347–348.

[10] "High Impedance Fault Detection Using Third Harmonic Current", EPRI Report EL-2430, Prepared by Hughes Aircraft Co., June 1982.

[11] L.A. Kilar, M. Rosado, H.F. Farnsler, R.E. Lee, "Innovative Relay Methods for Detecting High Impedance Faults on Distribution Circuits", *Proceedings of the American Power Conference*, Vol. 41, April 1979, pp. 1180–1183.

[12] "Detection of Arcing Faults on Distribution Feeders", EPRI Report EL-2757, Prepared by Texas A & M University, December 1982.

[13] B.M. Aucoin and B.D. Russell, "Distribution High Impedance Fault Detection Utilizing High Frequency Current Components", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-101, No. 6, June 1982, pp. 1596–1606.

[14] B.D. Russell and R.P. Chinchali, "A Digital Signal Processing Algorithm for Detecting Arcing Faults on Power Distribution Feeders", Presented at the IEEE Power Engineering Society 1988 Winter Meeting, New York, NY, 88 WM 123-2.

[15] T. Kohonen, "The 'Neural' Phonetic Typewriter", *IEEE Computer* Magazine, Vol. 21, No. 3, March 1988, pp. 11–22.

[16] H.P. Graf, L.D. Jackel, W.E. Hubbard, "VLSI Implementation of a Neural Network Model", *IEEE Computer* Magazine, Vol. 21, No. 3, March 1988, pp. 41–49.

[17] S.E. Fahlman and G.E. Hinton, "Connectionist Architectures For Artificial Intelligence", *IEEE Computer* Magazine, Vol. 20, No. 1, January 1987, pp. 100–109.

[18] J.L. Elman and D. Zipser, "Learning the Hidden Structure of Speech", ICS Report 8701, Institute for Cognitive Science, University of California at San Diego, February 1987.

[19] T.J. Sejnowski and C.R. Rosenberg, "NETtalk: A Parallel Network that Learns to Read Aloud", Technical Report EECS–86101, John Hopkins University, 1986.

[20] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning Internal Representations by Error Propagation", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (eds. Rumelhart & McClelland), Vol. 1, MIT Press, 1986, pp. 318–362.

[21] "EMTP Rule Book - Version 1.0", EPRI Report El-4541, Vols. 1-2, Prepared by Systems Control, Inc., 1986.

[22] S. Ebron, *A Neural Network Processing Strategy for the Detection of High Impedance Faults*, Master's Thesis, Electrical and Computer Engineering Department, NCSU, 1988.

BIOGRAPHIES

Sonja Ebron was born in Durham, North Carolina in 1963. She received her B.S.E.E. and M.S.E.E. degrees from North Carolina State University in 1986 and 1988, respectively. Ms. Ebron is currently a Ph.D. student and McKnight Fellow at the University of Florida in Gainesville. She is also a member of the IEEE Power Engineering, Computer, and Control Societies.

David Lubkeman was born in St. Louis, Missouri in 1957. He received his B.S.E.E., M.S., and Ph.D. degrees from Purdue University in 1979, 1980, and 1983, respectively. Dr. Lubkeman has been an assistant professor in the Department of Electrical and Computer Engineering at North Carolina State University, as well as a principal investigator in the Electric Power Research Center, since 1983.

Mark White received his B.S.E.E. degree from the University of Nebraska in 1971 and his Ph.D. degree from the University of California, Berkeley in 1978. He served on the faculty at the University of California, San Francisco from 1978 to 1986. Dr. White has been an associate professor in the Department of Electrical and Computer Engineering at North Carolina State University since 1986. His areas of research include non-linear adaptive signal processing algorithms and speech processing for cochlear implants and hearing aides.

N. E. Nilsson (Ohio Edison Company, Akron, Ohio): The authors are to be commended on their fine work. However, it should be recognized that a neural network strategy for analyzing high impedance ground faults is highly theoretical and may require some refinements before it becomes truly practical.

The authors have utilized neural networking which is a branch of artificial intelligence. Accordingly, there will be some interaction required by way of consultation with a system operator[1][2]. It is not clear from the paper whether the authors intend for the neural network to alarm the operator, in which case an explanation will be required describing the condition which the neural network believes to be abnormal in order to justify action on the part of the operator; or if the neural network is to behave like a fault protective relay and initiate a trip. In either case, it will be necessary for the neural network to explain why it tripped or alarmed. Would the authors more specifically describe the ranges of the twenty sensing neurons which would indicate a high impedance ground fault with a high level of confidence?

A neural network learns by matching patterns. The authors appear to have experimented with one specific circuit. Are there a range of parameters which can be sensed which are universal to all circuits or will the neural network have to learn a different pattern for each and every different distribution circuit?

I am left with the understanding that the authors have simply taught the neural network to look for either a normal load case or a high impedance ground fault situation. A neural network can be taught to look for several different patterns. If the neural network in this can were taught to look for capacitor switching and motor starting, for example, would its rate of success be better at detecting a high impedance ground fault? The reason for asking this question is that if that neural network can learn to identify things like capacitor switching which may be similar to a ground fault, it is less likely that it will incorrectly identify capacitor switching as a ground fault.

As was pointed out in the paper, the neural network learns by applying weighting factors to the neural connections. There are a number of ways of doing this. How did the authors determine that their method was the optimum method? Could the weighting routine provide a more accurate conclusion if it did not have to provide an answer so fast?

The use of neural networks is somewhat like the application of fuzzy logic to a rule based artificial intelligence expert system. There is a range of probability that any particular conclusion is the correct conclusion. I am not totally convinced that a neural network of twenty sensor nodes can identify a high impedance ground fault correctly anywhere and under any circumstance on a distribution circuit with one hundred percent probability of success. Nevertheless, the authors have shown that this technique is at least partially successful and so I would encourage them to continue their work in this area.

[1] R. P. Schulte, S. L. Larsen, G. B. Sheble, J. N. Wrubel and B. F. Wollenberg, "Artificial Intelligence Solutions to Power System Operating Problems," IEEE Transactions on Power Systems, Volume PWRS-2, No. 4, November 1987, pp. 920-926.

[2] N. E. Nilsson, "Application of Computer Artificial Intelligence Techniques to Analyzing the Status of Typical Utility Electric Power Plant Systems," IEEE Transactions on Energy Conversion, Volume 4, No. 1, March 1989, pp. 1-8.

Manuscript received May 31, 1989.

S. Ebron, D. L. Lubkeman, M. White: We thank Mr. Nilsson for his interest in our work as well as for his thoughtful comments and questions. We will answer them in the order in which they appear.

The discusser is correct when he states that the practical use of neural networks for high-impedance fault (HIF) detection will require further investigation. We made several points to that effect in our paper. Since this approach involves a new technology, we concentrated on demonstrating the *procedure* required in a practical application of neural networks. We chose to use simulation studies so that we could have more control over the mix of cases that were presented to the neural network. Future work will involve training a neural network on real HIF data.

Mr. Nilsson asks whether the neural network would sound an alarm or initiate a trip. We envisioned a scenario in which an operator would be alarmed. The neural net need not explain specifically why it alarms, any more than a smoke detector does. An alarm is caused by recognition of a pattern of events indicative of an HIF. The activation of hidden neurons would indicate which features triggered the alarm, but this type of information might be difficult to interpret for use as an operational aid.

The discusser next asks for boundaries on the twenty input parameters that would suggest the presence of a fault. This is a good question. The answer is that we cannot readily describe the ranges of values for these twenty parameters (over 1 cycle), the rates of change in those values (over 10 cycles), nor the correlations between parameters measured in each cycle, that would indicate a fault. In a problem as complex as HIF detection, it is difficult to state a "rule," as in knowledge-based systems, that would explicitly describe an HIF. For example, the rule governing output for an XOR (exclusive-or) logic gate, shown in Figure 1, can be stated: "If input #1 is high and input #2 is low, or if input #1 is low and input #2 is high, then output is high—else, output is low." A neural network could be trained to mimic the XOR gate using data in the truth table and the resulting weights could be used to *infer* the rule being implemented. However, a neural net that performs the XOR function [20] requires only five nodes, three node biases and six weighted links (see Figure 1). Though the neural net described in this paper is trained in the same manner, it consisted of 801 nodes, 601 biases, and 120,400 weights. Any inference of an HIF rule using these weights would be mere guesswork, as would an inference based on training data. If a set of explicit rules could be found, no further use of the neural network would be required.

Mr. Nilsson asks whether any one set of parameters, such as the twenty variables used in our research, can be relied upon to detect HIFs on a wide range of distribution circuits. There are certain parameters that tend to indicate an HIF on any circuit. Some of these parameters have been identified in past work on HIF detection (reviewed in the paper); they include information relating to high-frequency current components and a high ratio of zero sequence to positive sequence current. However, since the configuration and operating characteristics of feeders vary from utility to utility, the parameter values indicating a fault will be relative to values indicating normal operation. Also, some parameters may be more clearly correlated to an HIF on certain types of feeders than on others. The feeder voltage levels, types of customers, level of background harmonics, etc., would have to be taken into account. In short, each feeder would require a customized neural network, but each network would accept the same types of input and have the same configuration (number of nodes); the neural nets would differ only in the values of link weights. This could be implemented by training a network on examples relating to a number of circuits, and then fine-tuning weights for a particular circuit.

The discusser asks whether we simply trained the neural network to monitor normal and abnormal cases. Actually, the network was trained to look for four distinct case types and to provide different output for each type. The case types were as follows:

(1) Normal load switching—output 0.1,
(2) Normal load and capacitor switching—output 0.2,

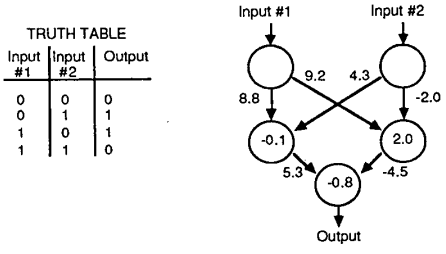

Fig. 1. Truth Table and Trained Neural Network for XOR Gate Implementation

(3) Normal load switching with an HIF event—output 0.8,
(4) Normal load and capacitor switching with an HIF event—output 0.9.

As reported in the paper, the network was not effective in distinguishing HIF events from capacitor switching. Performance could be improved by including a number of outputs relating to indirect goals, such as identifying capacitor switching and motor starting. It also may have helped to add additional parameters for identifying the insertion of a capacitor, possibly related to changes in reactive power flow.

Mr. Nilsson next asks whether our method for computing weights is optimal, and whether we sacrificed accuracy for speed. As yet, there is no optimal method of choosing weights in a feedforward layered network with hidden layers. However, the method (developed by Rumelhart, et. al. [20]) reviewed in the paper is based on a steepest descent algorithm that is *globally* optimal for networks with no hidden layers [A]. Experience with other applications has shown that this method is generally successful when hidden layers are added, though optimality is not guaranteed [15–19]. This training algorithm is not fast; it is an iterative procedure. For our case studies, decreasing the error tolerance increased the training time but produced a negligible change in weights and no improvement in accuracy. However, in other applications, a relation between training time and accuracy could exist. Note, however, that the time required to train the network is unrelated to the speed at which the network operates in the detection mode, since the weights used for fault detection are computed during the training phase.

We disagree with the discusser's assertion that neural networks are comparable to fuzzy logic in expert systems. Unlike the latter form of artificial intelligence, neural networks do not require explicitly-stated rules with probabilities of rule accuracy. Indeed, a neural network forms its own logic and stores that knowledge in its weights; instead of being given a rule, it "learns" one by example. In our neural network, there is no range of probability associated with the network's response; it is either right or wrong in each case.

Finally, Mr. Nilsson expresses skepticism at the notion that a network of twenty input nodes can flawlessly detect HIFs. Actually, there were 200, not 20, input nodes in our network. However, we join the discusser in his skepticism. We did not suggest that a network with even one million input nodes could detect an HIF "anywhere and under any circumstance on a distribution circuit with one hundred percent probability of success." Our intent was to show the procedure by which a neural network could be used in a practical application. Obviously, additional work is needed in identifying a more effective set of input parameters for this particular HIF detection technique. There are also a number of alternate neural network configurations that could be tested.

References

[A] R. Rosenblatt, *Principles of Neurodynamics,* Spartan Books, New York, NY, 1959.

Manuscript received June 12, 1989.