A Neural Network Methodology for Process Fault Diagnosis

The ability of knowledge-based expert systems to facilitate the automation of difficult problems in process engineering that require symbolic reasoning and an efficient manipulation of diverse knowledge has generated considerable interest recently. Rapid deployment of these systems, however, has been difficult because of the tedious nature of knowledge acquisition, the inability of the system to learn or dynamically improve its performance, and the unpredictability of the system outside its domain of expertise.

This paper proposes a neural-network-based methodology for providing a potential solution to the preceding problems in the area of process fault diagnosis. The potential of this approach is demonstrated with the aid of an oil refinery case study of the fluidized catalytic cracking process. The neural-network-based system successfully diagnoses the faults it is trained upon. It is able to generalize its knowledge to successfully diagnose novel fault combinations it is not explicitly trained upon. Furthermore, the network can also handle incomplete and uncertain data. In addition, this approach is compared with the knowledge-based approach.

Venkat Venkatasubramanian
King Chan
Laboratory for Intelligent Process Systems
School of Chemical Engineering
Purdue University
West Lafayette, IN 47907

Introduction

The ability of knowledge-based expert systems (KBES) to facilitate the automation of difficult problems in process engineering that require symbolic reasoning and an efficient manipulation of diverse knowledge has generated considerable interest over the recent past. Much attention has been paid recently to the problem of process fault detection and diagnosis by artificial-intelligence-based techniques (Kramer and Palowitch, 1985; Shum et al., 1988; Venkatasubramanian and Rich, 1988). However, rapid deployment of these systems has been difficult to achieve because of certain inherent limitations associated with current KBES. These limitations include the tedious nature of knowledge acquisition, the inability of the system to learn or dynamically improve its performance, and the unpredictability of the system outside its domain of expertise.

The present study proposes a neural-network-based approach for providing a potential solution to the preceding problems in the area of process fault diagnosis. The feasibility of this approach is explored through a neural-network-based fault diagnosis case study of a fluidized catalytic cracking unit (FCCU). Previously, the FCCU case study has been used to develop a knowledge-based diagnostic expert system implemented in OPS5 (Brownston et al., 1985). Thus, it serves as a useful test bed for evaluating the proposed neural network diagnostic methodology, and it also facilitates the comparison of both approaches.

In the current research, an analysis of the recall capability to trained faults and the generalization capability to symptoms resulting from novel fault combinations is performed. The generalization proficiency vs. network topology (i.e., number of hidden layers and hidden units) is examined. A comparison between the neural-network-based and the knowledge-based FCCU diagnostic versions are also investigated.

The remainder of the paper is organized in the following manner. First, a synopsis of neural networks along with motivational issues for their application is addressed. Second, the learning algorithm used in the present study is discussed. Following this, an overview of CATDEX, a rule-based diagnostic expert system for a fluidized catalytic cracking unit, is given. Next, the details of the neural-network-based fault diagnosis system are described. Subsequently, the fault diagnosis experiments and sim-

Correspondence concerning this paper should be addressed to V. Venkatasubramanian.

ulation results are presented. Lastly, from these results, the facility of a neural-network-based approach for overcoming the KBES bottlenecks is explicated.

Neural Networks: Synopsis and Motivational Issues

Rapid advances in neuroscience and in computer science are arousing renewed interests in neural networks as potentially new problem-solving architectures. Neural networks are dynamic systems composed of highly interconnected layers of simple neuron-like processing elements (Amari, 1977; Feldman and Ballard, 1982; Kohonen, 1984; Rumelhart et al., 1986a). These layers are categorized as input layers where patterns are presented to the network and as output layers which contain the response to a given input. Furthermore, they may also contain intermediate layers or hidden layers. Figure 1 presents an example of a typical neural network architecture.

Neural network operations consist of a learning or training phase, a recall phase, and a generalization phase. During the learning phase, the network is repeatedly presented with a set of input-output patterns. Learning is accomplished by a general rule which dynamically modifies the weights of all interconnections in an attempt to generate the desired output pattern for each presented input pattern. After the learning is complete, the network operates in a recall phase where it generates responses to input patterns used in training. It also operates in a generalization phase where it generates responses to similar or novel input patterns.

Neural network computations are collectively performed by the entire network with knowledge represented in the connection weights between processing elements. Consequently, the collective operations result in a high degree of parallel computation which enables the network to solve complex problems rapidly. In addition, the distributed representations lead to greater fault tolerance and to graceful degradation when problems are encountered beyond its range of experience. Furthermore, other beneficial attributes include the ability to adjust dynamically to environmental changes, to perform useful generalizations from specific examples, and to recognize invariances from complex high-dimensional data.

Thus, neural networks appear to be applicable to chemical process engineering problems requiring pattern recognition and pattern classification. These areas include fault diagnosis and detection, process design, process control, and process simulation. Recently, researchers have begun to look at applications such as interpreting biosensor data (McAvoy et al., 1989), fault diagnosis of a chemical process system (Venkatasubramanian, 1985), and for single fault diagnosis of three continuous-stirred-tank reactors connected in series (Hoskins and Himmelblau, 1988).

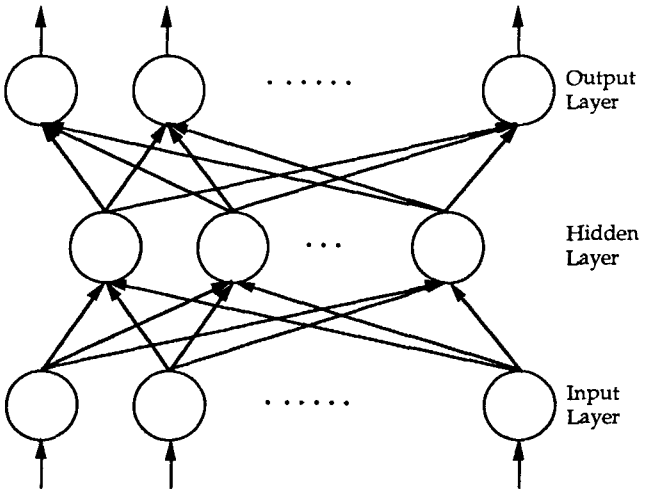

Figure 1. Typical neural network architecture.

Back-Propagation Neural Network

The back-propagation learning algorithm (Jones and Hoskins, 1987; Rumelhart et al., 1986b, Werbos, 1974) has been widely explored in recent years and has produced successful learning and generalization results in various task domains. For instance, some systems include those for text-to-speech conversion (Sejnowski and Rosenberg, 1986), handwritten character recognition (Burr, 1986), speech recognition (Ellman and Zipser, 1987), and sonar target identification (Gorman and Sejnowski, 1988). It is an error-correcting learning procedure which generalizes the Widrow-Hoff algorithm (or delta rule) to multilayer networks. Back-propagation is intended for networks with an input layer, any number of hidden layers, and an output layer. Each layer is fully connected to the succeeding layer, and connections within a layer or from higher to lower layers are prohibited.

The networks employed in the present study are feedforward and possess one or two hidden layers of processing elements (PE). Upon presentation of an input-output pattern, the total input, net_j, to the jth processing element is obtained by calculating the weighted sum of all PE i outputs that are connected to PE j,

$$\text{net}_j = \sum_i w_{ji} x_i + b_j \tag{1}$$

where w_{ji} is the connection weight from the ith PE to the jth PE, x_i is the output of PE i, and b_j is the bias of the jth PE. However, the bias, b_j, is eliminated by giving every PE an extra input connection with an activity level fixed at one (Figure 2a). Consequently, the weights on this special connection are just the negative of the threshold and must be learned in the same manner as other weights. A sigmoid transfer function (Figure 2b) is then applied to net_j to obtain the output of the jth PE,

$$x_j = f(\text{net}_j) = \frac{1}{(1 + e^{-\beta \text{net}_j})} \tag{2}$$

where β is a constant gain term which determines the slope of the sigmoid function at $\text{net}_j = 0$. The output serves as input to succeeding layers which is continued until the output layer is reached and is referred to as forward-activation flow. The subsequent weight adaptation or learning process is accomplished by the back-propagation learning algorithm.

The goal of the back-propagation learning algorithm is to iteratively minimize the average squared error between values of the output PE and the correct patterns provided by a teaching input (i.e., gradient descent in error space). This is accomplished by first computing the error gradient (δ_j) for each PE on the output layer,

$$\delta_j = x_j(1 - x_j)(t_{pj} - x_j) \tag{3}$$

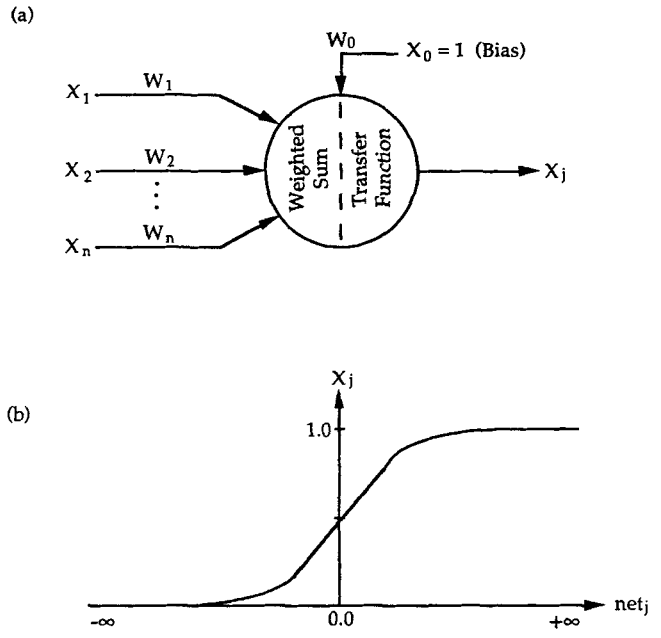

Figure 2. Fundamental unit of neural networks: a. processing element or artificial neuron; b. sigmoid transfer function.

where t_{pj} is the correct teaching value for output unit j and for input pattern p. The error gradient is then recursively determined for hidden layers (i.e., internal PE's) by computing the weighted sum of the errors at the previous layer,

$$\delta_j = x_j(1 - x_j) \sum_k \delta_k w_{kj} \tag{4}$$

where k is over all PE's in the layers above j. Thus, the errors are propagated backwards one layer, and the same procedure is recursively applied until the input layer is reached. This process of back-propagating the errors is referred to as backward-error flow. The error gradients are then used to update the network weights,

$$\Delta w_{ji}(n) = \eta\, \delta_j\, x_i \tag{5a}$$

$$w_{ji}(n + 1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{5b}$$

where n indexes the presentation number during training and η is the learning rate which provides the step size during gradient descent. Generally, to assure rapid convergence, large step sizes which do not lead to oscillations are used. Convergence, however, may be improved by including a momentum term, α, which determines the effect of previous weight changes on present changes in weight space. The weight change after the nth data presentation is,

$$\Delta w_{ji}(n) = \eta\, \delta_j\, x_i + \alpha \Delta w_{ji}(n - 1) \tag{6}$$

where $0 < \alpha < 1$.

Overview of CATDEX: An Expert System for Diagnosing a FCCU

The complexity of modern chemical plants is causing increased safety and reliability problems because of the greater possibility of equipment failure and operator error. Current efforts in solving this problem employ KBES for automating the detection and diagnosis of process abnormalities. This is accomplished by the application of artificial intelligence techniques for capturing the reasoning processes of a diagnostic specialist (Firebaugh, 1988; Harmon and King, 1985; Nilsson, 1980).

The catalytic cracking unit diagnostic expert (CATDEX) is one such prototypical expert system for diagnosing a fluidized catalytic cracking unit (Figure 3) (Venkatasubramanian, 1988) which is a major conversion apparatus of the petroleum refining industry (consult Shreve and Brink, 1977, for the details of catalytic cracking). CATDEX was developed as an off-line interactive decision support expert system for a novice petroleum engineer or a refinery operator. Implemented in OPS5, it utilizes a backward-chaining or goal-driven inference mechanism and a generalized depth-first search control strategy.

CATDEX focuses on three categories of operational problems which may eventually lead to plant shutdown and/or unit damage. These problems are excessive catalyst loss, poor catalyst circulation, and yield loss. Furthermore, each of these faults exhibits symptoms which in turn may be faults that exhibit other symptoms. This proceeds until one reaches rudimentary faults, called elemental symptoms, which cause no other process malfunctions. As a result, an exhaustive set of potential FCCU faults and their associated symptoms may be fully enumerated into a fault-tree-like structure (Himmelblau, 1978), called an inference network (Rich and Venkatasubramanian, 1987). This network is then encoded as rules in the knowledge base of the expert system. Consequently, fault diagnosis is performed by systematically searching for links and relationships within the inference network, which produce the causal source of the symptoms resulting from operational malfunctions.

Neural Network Methodology for Diagnosing a FCCU

CATDEX is thus a fault diagnostic expert system which utilizes an interactive guided inference on a knowledge base consisting of production rules that represent the inference network. The resulting system shows promise in efficient diagnosis. However, it lacks the flexibility to process changes, is incapable of diagnosing novel symptom combinations, and requires a considerable amount of implementation effort. In addition, as is typical of fault trees and inference networks, it requires considerable effort in getting them to perform multiple fault diagnosis or diagnosis based on incomplete or partial symptoms.

The proposed neural-network-based methodology (Neural CATDEX) treats fault diagnosis as one of classification. For instance, in the initial training process the neural network is presented with N observation vectors (the symptoms resulting from a fault) and its associated class (the fault or malfunction). Similar to classical decision theory, neural networks perform the classification by creating decision boundaries to separate the N different pattern classes. Unlike traditional classifiers, however, neural networks, using the back-propagation learning algorithm, develop complex input-output mappings which represent salient features and regularities of the task domain (Rumelhart et al., 1986c). Moreover, it is these internal representations which lead to novel generalizations in future classification activities involving new or slightly modified fault symptoms.

The representation of CATDEX's inference network consid-

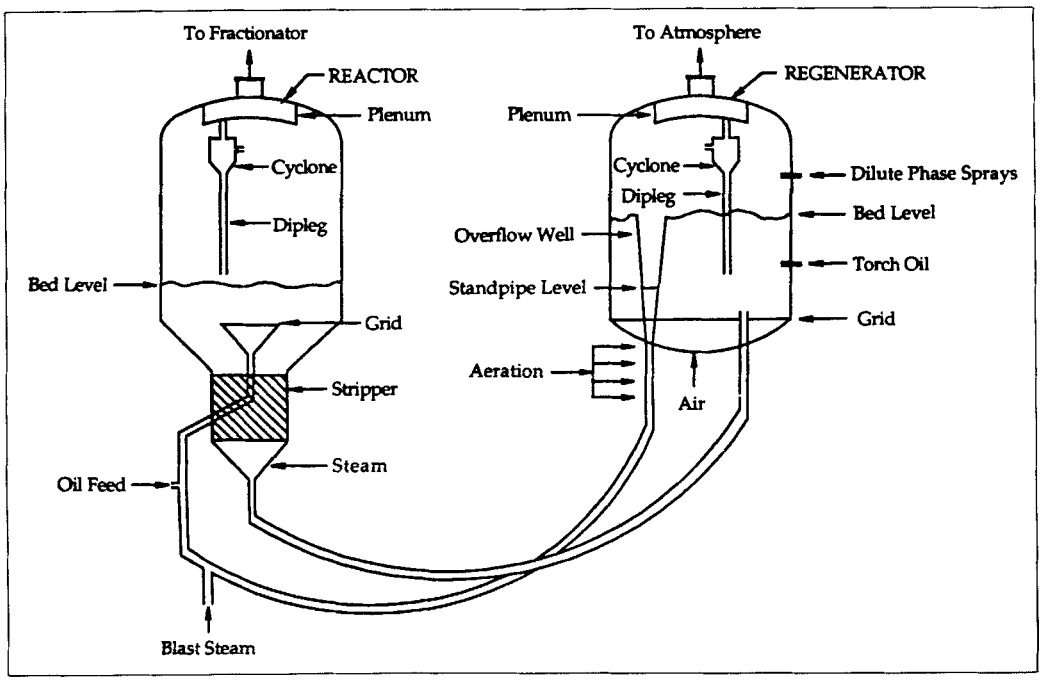

Figure 3. Typical fluidized catalytic cracking unit.

ers each fault and symptom node as a neural-network-processing element. Most importantly, hidden layers are added between symptoms and their immediate faults to enable the development of internal representations. The simulation issues for Neural CATDEX are presented in the following sections.

Network training

In this initial study, training was performed only on the bottom-most inference network layer consisting of the elemental symptoms and its immediate faults (Figure 4). In addition, to simplify the ensuing analysis, the excessive catalyst loss and yield loss malfunction categories were chosen because they contain the most intriguing Boolean combinations for learning and subsequent generalization. During network training, the input patterns consist of symptoms resulting from a fault, and the output patterns contain the specific fault. Thus, as numerically depicted in Figure 4, the neural network contains 18 input nodes and 13 output nodes which represent the symptoms and its immediate faults, respectively. In this study, the network was trained on all single fault occurrences in the partial FCCU inference network. Table 1 lists the training data for the 13 faults involved. The training data were presented a total of 500, 1,000, 2,000, 4,000, 6,000, and 8,000 iterations or time steps. Lastly, all network training was repeated three times with different initial random weight values to average over variations in performance.

Back-propagation learning parameters

In all simulation experiments, the learning rate parameter, η, was set at 0.9, the momentum term, α, was set at 0.6, and the gain term, β, was set at 1.0. Furthermore, the initial weights of the network were assigned to small uniformly-distributed random values between -0.1 and $+0.1$ to prevent the hidden units from acquiring identical weights during training. Finally, the results of this study were obtained by the NeuralWorks Professional II neural network simulation package.

Accuracy calculations

It is evident from Table 1 that the training data are binary values, in which 1 represents a fault and 0 a nonfault. The outputs, however, are real values since infinite weights are required to drive the results to 1 or 0.

In this study, the accuracy of a single output node is defined as one minus the absolute difference between the desired fault and the predicted fault determined during the recall or generalization phase. The network accuracy in fault detection for a single training pattern is computed as the average of the 13 output node accuracies. Consequently, the overall accuracy is the average network accuracy for all training patterns over all training repetitions.

Number of hidden layers and hidden units

As previously mentioned, neural networks partition the problem space into numerous decision regions when performing its classifications. The partitioning capability is a function of both the number of hidden layers and hidden units. For example, networks utilizing sigmoid nonlinearities form fuzzy hyperplanes with no hidden layers, convex polygonal boundaries for one hidden layer, and arbitrarily complex decision regions for two hidden layers (Lippmann, 1987; also see Kolmogorov, 1956, for a mathematical treatment on the capabilities of multilayer networks). Likewise, the number of hidden units must also be determined such that the complexity of the decision regions formed corresponds to the problem complexity needed for correct classification.

Presently, neural networks with one hidden layer were used primarily in simulations. Nevertheless, experiments on networks with two hidden layers were performed to examine fault diag-

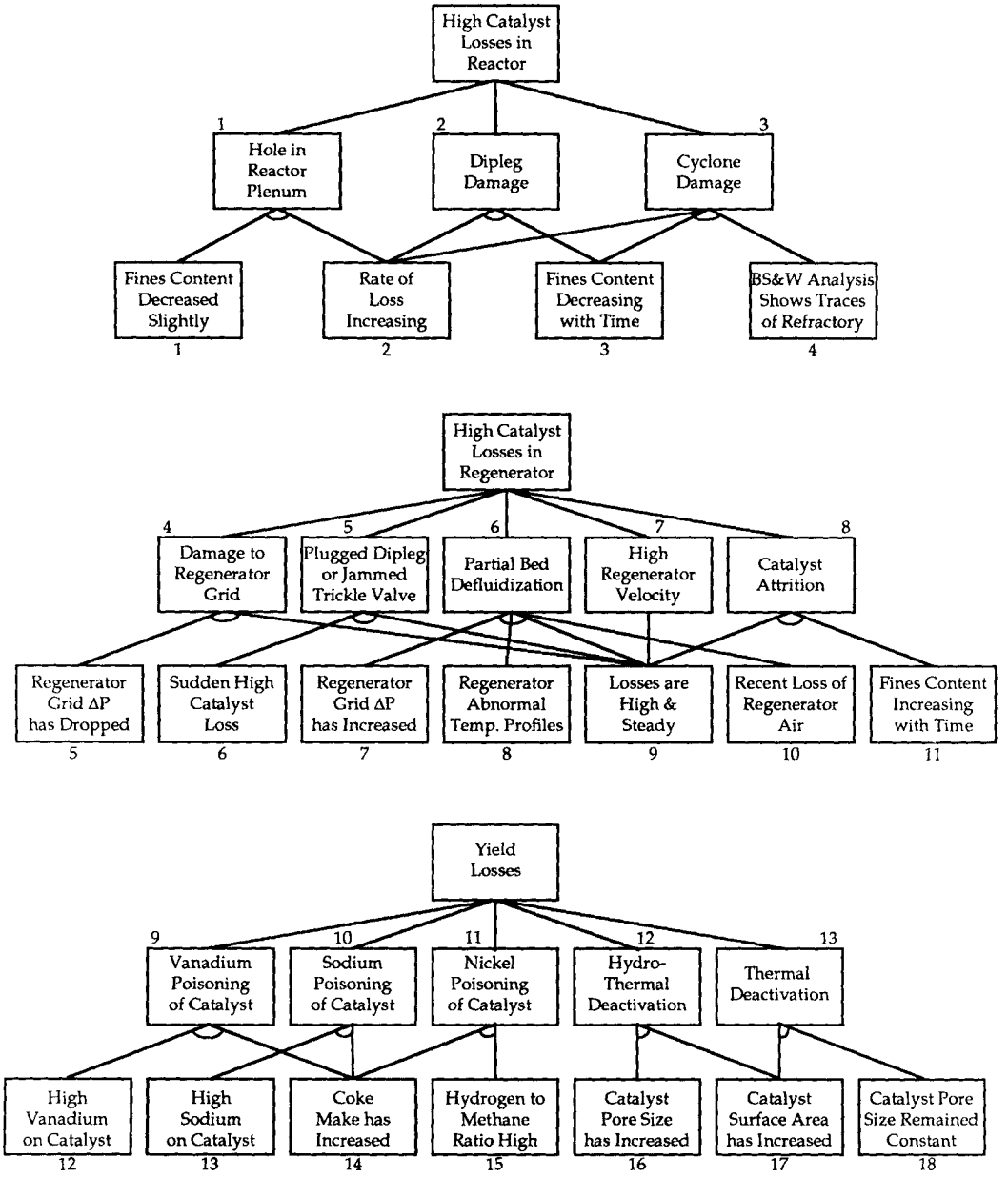

Figure 4. Presently employed FCCU inference network for Neural CATDEX.

The top layer represents the operational problem categories. The next layer (1–13) represent the faults of the elemental symptom layer (1–18).

Legend: △ AND Node ∧ OR Node.

nosis on the FCCU with more complex decision regions. The number of hidden units required to perform accurate diagnosis was determined empirically. All series of experiments were repeated using networks of 5, 7, 11, 15, 21 and 27 hidden units. Due to overcrowding of the plots, only the results of the networks that contain 5, 7, 15 and 27 hidden units are shown. The other results were similar to these.

A Performance Evaluation of Neural CATDEX

In the ensuing sections, the fault diagnosis simulation results are presented in their entirety. The facility of neural networks for recalling trained faults is explored. Next, the trained network generalization capacity to multiple faults is investigated by presenting symptoms resulting from two and three process malfunctions. Following this, the fault diagnosis behavior is analyzed for cases where faults share common symptoms and for cases where partial symptoms of a fault are presented to the network. Lastly, the recall and generalization results are examined for networks trained on a subset of single fault input-output patterns. The preceding experiments are duplicated for two hidden-layer neural network architectures.

Recall results of trained single fault patterns

To test the learning proficiency and thus the network convergence, the symptoms resulting from the 13 faults were presented to all of the trained networks. During this stage, all output val-

Table 1. Single-Fault Training Data

i = Input Pattern Symptoms
d = Output Pattern Single Fault Occurrence
(18 PE's)
(13 PE's)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
i	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d	1	0	0	0	0	0	0	0	0	0	0	0	0					
i	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d	0	1	0	0	0	0	0	0	0	0	0	0	0					
i	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d	0	0	1	0	0	0	0	0	0	0	0	0	0					
i	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
d	0	0	0	1	0	0	0	0	0	0	0	0	0					
i	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
d	0	0	0	0	1	0	0	0	0	0	0	0	0					
i	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
d	0	0	0	0	0	1	0	0	0	0	0	0	0					
i	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
d	0	0	0	0	0	0	1	0	0	0	0	0	0					
i	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
d	0	0	0	0	0	0	0	1	0	0	0	0	0					
i	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
d	0	0	0	0	0	0	0	0	1	0	0	0	0					
i	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
d	0	0	0	0	0	0	0	0	0	1	0	0	0					
i	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
d	0	0	0	0	0	0	0	0	0	0	1	0	0					
i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
d	0	0	0	0	0	0	0	0	0	0	0	1	0					
i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
d	0	0	0	0	0	0	0	0	0	0	0	0	1					

Consult Figure 4 for the input-output pattern legend: 1 = fault; 0 = nonfault

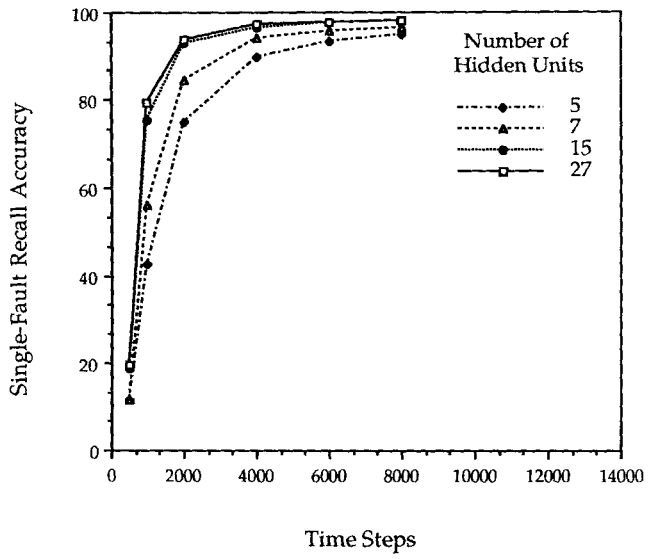

Figure 5. Network performance curves for recalling trained single faults.

ues were quickly (within seconds) and accurately classified to the correct fault. Figure 5 shows the overall average learning curves for networks trained on single faults.

The networks, therefore, were able to learn the correct association between the process symptoms and process malfunctions. The best average performance was achieved by a network with 27 hidden units which attained 98.18% accuracy in fault classification at 8,000 time steps (Table 2). Table 2 suggests an insignificant increase in performance with the number of hidden units since the observed recall deviation is less than 4% between networks of five and 27 hidden units. However, as nearly exact

Table 2. Network Performance During Recall of Single-Fault Patterns

No. of Hidden Units	Overall Accuracy (%) @ 8,000 Time Steps
5	94.71
7	96.42
11	97.39
15	97.88
21	98.02
27	98.18

recall is usually expected, the number of hidden units is more critical for developing desirable generalizations to novel symptom combinations.

Multiple fault generalization results

In the next series of simulations, a set of symptoms excluded from the initial training patterns was presented to the network to determine its ability to perform useful generalizations. This capacity was examined by introducing input symptoms resulting from two and three process malfunctions. This was deemed interesting because multiple fault diagnosis is difficult to accomplish by traditional fault tree analysis. Since there was an inordinate number of two- and three-fault combinations, a subset of symptom patterns which were considered difficult to generalize was chosen. Thus, 33 symptoms for two faults and 30 symptoms for three faults were presented to the network for generalization. Figure 6 shows the overall average two-fault generalization curves for networks trained on single faults. Similarly, Figure 7 shows the overall average three-fault generalization curves.

The networks are thus able to perform novel generalizations to multiple fault conditions. As in recall, the asymptotic limits exhibited by these curves suggest a sufficient number of hidden units to perform the task. However, unlike recall where learning performances are not significantly affected by the number of hidden units, useful generalizations seem to require a certain minimum number of hidden units which depend on the complexity of the particular task. If too few are present, all the essential attributes of the input-output patterns cannot be completely represented.

The best average performance was achieved by a network with 27 hidden units which attained 77.54% accuracy for two-fault generalization and 53.60% accuracy for three-fault generalization at 8,000 time steps. It appears from Figures 6 and 7 that the generalization accuracy can further be improved with more hidden units as saturation levels have definitely not been attained. Thus, for the task of generalizing to multiple faults,

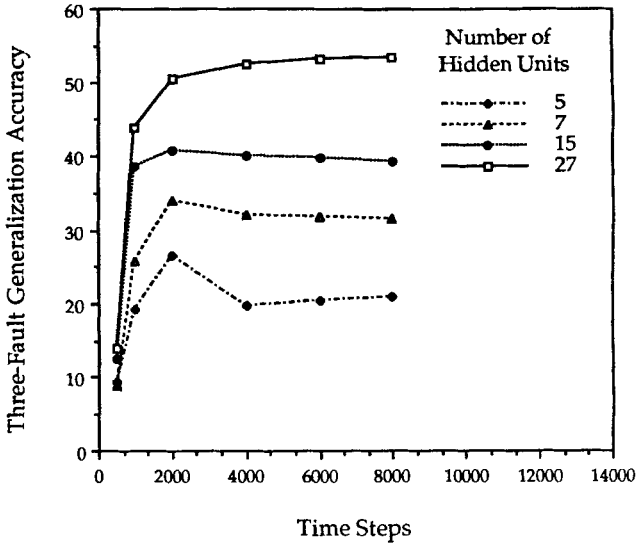

Figure 7. Network performance curves for three-fault generalization.

performance tends to increase with the number of hidden units. Lastly, Figures 8 and 9 display bar charts of selected two- and three-fault generalizations for networks trained on single faults, respectively. All the bar charts correspond to a network with 27 hidden units in a single hidden layer. Figure 8 displays the detection accuracy of the detection of two fault occurrence by the neural network. One of the two faults in Figure 8 is always fault 1, namely, "Hole in reactor plenum" as seen in Figure 4. The other member of the two fault pair varies from fault 2 through fault 13. So, the x-axis displays the fault combinations—(1, 2), (1, 3), and so on. The dark-colored bars correspond to the accuracy of the detection of fault 1, while the hatched bars correspond to the accuracy of the other faults in the set 2 through 13. Figure 9 has similar results, except that they now correspond to three fault combinations.

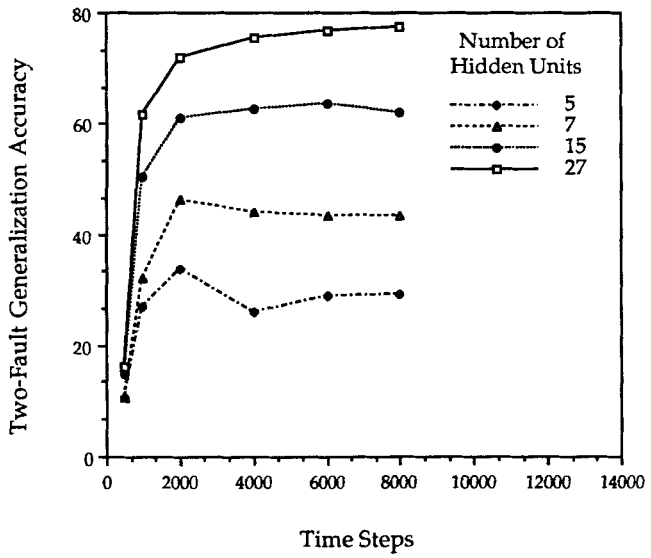

Figure 6. Network performance curves for two-fault generalization.

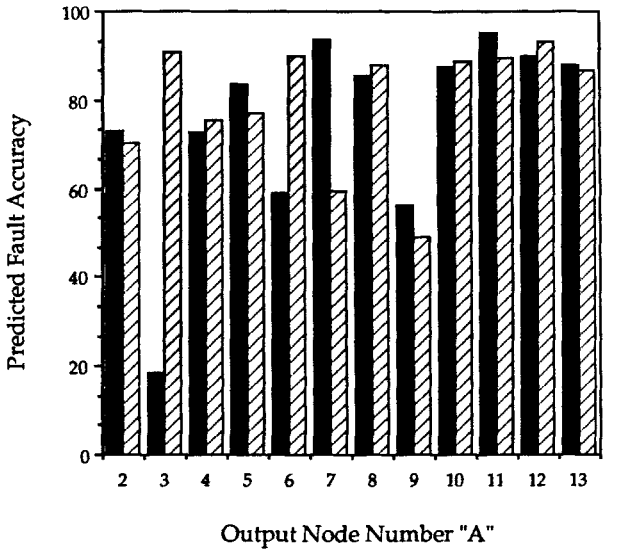

Figure 8. Bar chart for two-fault generalization.
Predicted faults: ■, 1; ▨, "A."

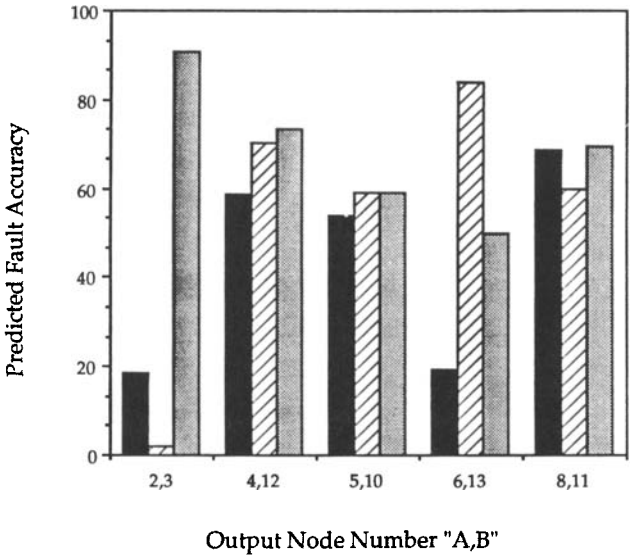

Figure 9. Bar chart for three-fault generalization.
Predicted faults: ■, 1; ▨, "A"; □, "B."

Generalization results for faults sharing common symptoms

From the preceding, it is evident that Neural CATDEX, unlike CATDEX which uncovers only single causal sources for a given symptom pattern, performs multiple fault diagnosis. This section presents the generalization results for faults which share common symptoms. It examines the two-fault generalizations involving fault node 4 with fault node 5–8 for the "high catalyst losses in regenerator" operational problem. The common symptom of faults 4–8, node number 9 ("losses are high and steady"), is explicit in Figure 4.

The generalization results are shown in Figure 10. From this figure, the outputs of fault nodes 4 and 6, which exhibit symptom nodes 5, 9 and 7, 8, 9, 10, respectively, result in a 29% accuracy for fault 4 and a 80% accuracy for fault 6. Since neural networks perform cooperative computation, that is the output from any node depends on the contributions from all other nodes in the network it is connected to, the faults that have more symptoms (or evidences) supporting them than other faults tend to get weighted more in the corresponding outputs generated by the neural network. This is reflected in the accuracy shown in Figure 10, such as in the case of fault 4 and 6. Fault 4 has two symptoms and fault 6 has four symptoms; as a result, when all these symptoms are observed, fault 6 is recommended with a higher accuracy than fault 4. The accuracy may be interpreted in a limited sense as the probability of the recommendations for the different faults predicted by the network.

Generalization results for partial symptom patterns

From the preceding sections, it appears that the neural network gives greater emphasis to faults displaying more symptoms. This behavior is a result of the cooperative nature of the computations performed by the neural networks. In this section, we present the network's response to input patterns with incomplete set of symptoms for fault occurrence. This is examined for the "high catalyst losses in reactor" operational problem which contains four symptom nodes and three fault nodes (Figure 4). The results are presented in Figure 11 in which the abscissa contains selected symptom patterns represented as a vertical binary string with symptom node 1 located at the base. From this figure, it is evident that the network generalizes to a "no fault" condition for an input of no faulty symptoms ("0000"). However, this is not the case for other partial symptom patterns. Even though the symptom patterns are incomplete and thus do not warrant the existence of a fault, the network generalizes to predict the occurrence of an appropriate fault with some nonzero accuracy.

For example, the appearance of the "rate of loss increasing" symptom ("0100"), which is shared by fault nodes 1, 2, and 3

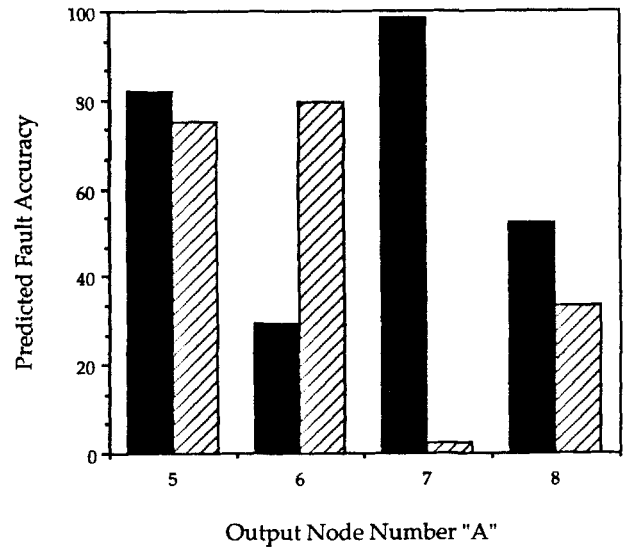

Figure 10. Probabilistic multiple-fault generalization for faults which share common symptoms for the "high catalyst losses in regenerator" operational problem.
Predicted faults: ■, 4; ▨, "A."

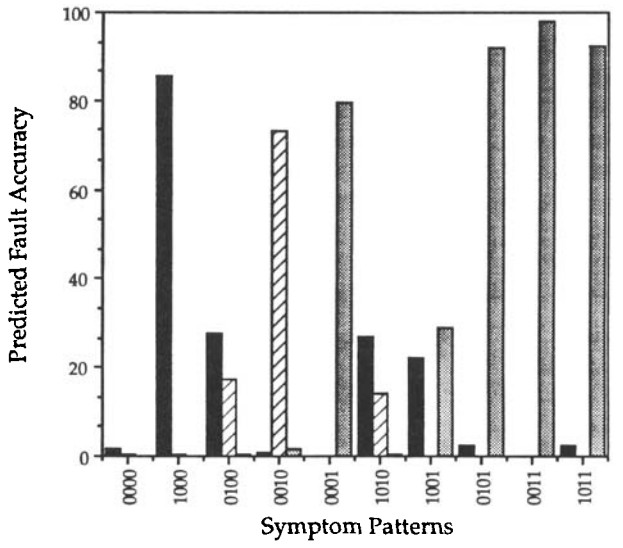

Figure 11. Probabilistic generalization for partial symptoms for the "high catalyst losses in reactor" operational problem.
Predicted faults: ■, 1; ▨, 2; □, 3.

suggests the possible occurrence of any one of the three faults or any other combination among these three. Since fault nodes 1 and 2 exhibit one of two symptoms resulting from a fault while fault node 3 exhibits only one of three symptoms, the probability of fault 1 or 2 occurring is higher and is reflected in the predicted accuracies (i.e., 28%, 17%, and 0.3% for faults 1, 2 and 3, respectively). Compare these accuracies with the case when all the evidences are present for these faults from Table 2 (the average overall accuracy of recall for these faults is 98.18%). Thus, for incomplete symptom patterns, the neural network gives greater emphasis to a fault displaying a higher percentage of symptoms which may result in its occurrence. Therefore, the network is able to generalize its diagnostic knowledge even for cases with incomplete evidence and come up with a probabilistic recommendation which are quite accurate. This is a very useful diagnostic characteristic in practical situations.

Recall and generalization results for incomplete training of single fault patterns

This experiment investigates generalizations produced by networks trained on a subset of the single-fault input-output patterns. Networks in this experiment are trained with one of the 13 faults withheld. However, during the recall phase, symptoms resulting from all the 13 faults are presented to the network. Similarly, during generalization, symptoms resulting from trained and untrained faults are present.

The recall on trained faults were very good, similar to the accuracies shown in Figure 5 and Table 2. However, the recall accuracy for untrained faults were close to zero indicating that the network exhibited no knowledge of the untrained faults. Figure 12 displays the two-fault generalization results for cases where one single fault pattern is withheld during training (i.e., faults 5, 6, 9, 10, and 11 withheld one at a time). In all instances, the network is unable to generalize to the untrained faults. This is illustrated in Figure 12 as all untrained fault output nodes display near zero accuracies (the hatched bars are indistinguishable from the x-axis). As expected, the network is capable of

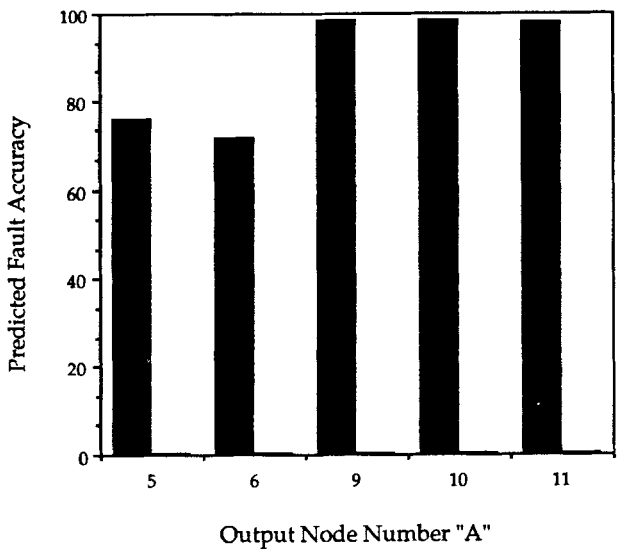

Figure 12. Generalization results for incomplete single-fault training.
Predicted faults: ■, 1; ▨, "A" (untrained single fault).

Table 3. Partial Listing of Two-Fault Generalization for Two Hidden Layer Networks @ 15,000 Time Steps

Network Architecture	Actual Faults	Predicted Faults
18-14-13-13	1, 6	6
	1, 12	9, 12
	1, 8	3, 7, 8
18-27-27-13	1, 3	2, 3
	1, 9	1, 2, 9
	1, 4	1, 4, 12

performing multiple fault diagnosis for all initially-trained faults.

Performance results for two hidden layers

The final series of simulations analyze the decision regions obtained for four-layer neural networks (i.e., two hidden layers). The topology of the networks in consideration contains 14–13 and 27–27 hidden units in each hidden layer, respectively. As in previous cases, the trained networks are tested on both recall and multiple fault generalization. These results are subsequently correlated with those obtained for the 27 hidden-unit single-layer network.

As anticipated, the increased connections of the four-layered networks require significantly more training iterations in attaining comparable single hidden layer recall accuracies. For example, recall accuracies of 98.18% were achieved by an 18–27–13 network in 8,000 time steps, while an 18–14–13–13 (96.12%) and an 18–27–27–13 (97.66%) networks require 15,000 time steps. Therefore, using an additional hidden layer by dividing the single hidden layer units into two layers (i.e., 14–13) or by incorporating another hidden layer of equal size (i.e., 27–27) does not significantly aid the recall process.

While the preceding results are expected, generalization results produced by an additional hidden layer are unpredictable. A partial listing of two- and three-fault generalizations for the four-layer networks trained on 15,000 time steps are in Tables 3 and 4, respectively. Thus the fault discriminatory capability with two hidden layers leads to erroneous fault predictions, while single hidden layer networks offer useful generalizations.

Conclusions and Future Work

In this paper, we have proposed a neural-network-based methodology for developing automated systems for process fault diagnosis. As demonstrated, neural networks are able to acquire diagnostic knowledge from examples of fault scenarios. This

Table 4. Partial Listing of Three-Fault Generalization for Two Hidden Layer Networks @ 15,000 Time Steps

Network Architecture	Actual Faults	Predicted Faults
18-14-13-13	1, 4, 13	13
	1, 6, 11	9, 12
	1, 5, 13	5, 13
18-27-27-13	1, 5, 9	None
	1, 7, 11	6
	1, 8, 12	4, 8, 12

knowledge acquisition is an automatic process driven by a learning algorithm called the back-propagation algorithm. The neural network's recall to trained faults is nearly perfect at 98%. Furthermore, unlike traditional fault tree analysis and inference networks which furnish a single causal source for a set of symptoms, this methodology performs multiple fault diagnosis while being trained on knowledge of single faults. This generalization to two and three faults yields accuracy measures of 78% and 54%, respectively, on the test cases we studied. It is also shown that Neural CATDEX's output can be interpreted as a likelihood for process malfunction. Thus the network has the useful characteristic of being able to deal with incomplete and uncertain evidences. While the network is able to generalize its knowledge for diagnosing multiple fault combinations, it was not explicitly trained upon, it is unable to diagnose single fault cases if that fault was not a member of the initial training set. Thus, it appears that these networks seem to perform well for novel conditions which are somewhat similar to what they had seen before, whereas they fail for other novel scenarios which are totally different from what they had seen before. It also appears that single hidden layer networks seem to outperform two hidden layer networks both in efficiency and accuracy.

This research demonstrates the feasibility for future developments in the application of neural networks to chemical process fault detection and diagnosis. It is also believed that an integration of neural networks and knowledge-based expert systems is essential to resolving the problems inherent in both problem-solving paradigms. Thus, expert systems endowed with a dynamic adaptive ability can greatly simplify the knowledge acquisition process. As a result, instead of supplying the knowledge, the domain expert need only refer to relevant data for subsequent network training. Thus the proposed neural network component can act as a preprocessor whose duties may encompass pattern recognition, sensor fault analysis, and/or signal recognition. It can also be a component of the rule-based system itself which performs learning, generalization, and/or classification.

The ability to learn from example, extract salient features from data, reason in the presence of novel, imprecise or incomplete information, tolerate noisy and random data, and degrade gracefully in performance makes neural networks ideal for chemical process engineering fault detection and diagnosis. The combination of neural-based and knowledge-based systems may improve the performance speed and reduce the complexity and time required for building intelligent systems.

Literature Cited

Amari, S. A., "A Mathematical Approach to Neural Systems," *Systems Neuroscience*, J. Metzler, ed., 67, Academic Press, New York (1977).

Brownston, L., R. Farrel, E. Kant, and N. Martin, *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming*, Addison-Wesley, Reading, MA (1985).

Burr, D. J., "A Neural Network Digit Recognizer," *Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics*, Atlanta, GA (Oct., 1986).

Ellman, J. L., and D. Zipser, "Learning the Hidden Structure of Speech," Tech. Rep. ICS Report No. 8701, University of California at San Diego, Institute for Cognitive Science, La Jolla, CA (1987).

Feldman, J. A., and D. H. Ballard, "Connectionist Models and their Properties," *Cognitive Sci.*, **6**, 205 (1982).

Firebaugh, M. W., *Artificial Intelligence: A Knowledge-Based Approach*, Boyd & Fraser, Boston (1988).

Gorman, R. P., and T. J. Sejnowski, "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets," *Neural Networks*, **1**, 75 (1988).

Harmon, P., and D. King, *Expert Systems*, Wiley, New York (1985).

Himmelblau, D. M., *Fault Detection and Diagnosis in Chemical and Petrochemical Processes*, Elsevier, Amsterdam (1978).

Hoskins, J. C., and D. M. Himmelblau, "Artificial Neural Network Models of Knowledge Representation in Chemical Engineering," *Comp. Chem. Eng.*, **12**(9/10), 881 (1988).

Jones, W. P., and J. C. Hoskins, "Back-Propagation, A Generalized Delta Learning Rule," *BYTE Mag.*, 155, (Oct., 1987).

Kohonen, T., *Self-Organization and Associative Memory*, Springer-Verlag, Berlin (1984).

Kolmogorov, A. N., "On the Representation of Continuous Functions of Several Variables by Superposition of Continuous Functions of a Smaller Number of Variables," *Doklady Akademii Nauk SSSR*, **108**, 179 (1956).

Kramer, M., and B. Palowitch, "Expert System and Knowledge-Based Approaches to Process Malfunction Diagnosis," AIChE Meeting, Chicago (Nov., 1985).

Lippmann, R. P., "An Introduction to Computing with Neural Nets," *IEEE ASSP Mag.*, **4**, 4 (Apr., 1987).

McAvoy, T. J., N. S. Wang, S. Naidu, N. Bhat, J. Gunter, and M. Simmons, "Interpreting Biosensor Data via Back-propagation," *IJCNN Int. Joint Conf. on Neural Networks*, **1**, 227, Washington DC (June, 1989).

Nilsson, N. J., *Principles of Artificial Intelligence*, Morgan Kaufmann Publishers, Los Altos, CA (1980).

Rich, S. H., and V. Venkatasubramanian, "Model-Based Reasoning in Diagnostic Expert Systems for Chemical Process Plants," *Comp. Chem. Eng.*, **11**(2), 111 (1987).

Rumelhart, D. E., G. E. Hinton and J. L. McClelland, "A General Framework for Parallel Distributed Processing," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: 1. Foundations*, D. E. Rumelhart and J. L. McClelland, eds., MIT Press, Cambridge, MA (1986a).

Rumelhart, D. E., G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: 1. Foundations*, D. E. Rumelhart and J. L. McClelland, eds., MIT Press, Cambridge, MA (1986b).

Rumelhart, D. E., G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nat.*, **323**(9), 533 (Oct., 1986c).

Sejnowski, T. J., and C. R. Rosenberg, "NETtalk: A Parallel Network that Learns to Read Aloud," Technical Report 86/01, Department of Electrical Engineering and Computer Science, Johns Hopkins University, Baltimore (1986).

Shreve, R. N., and J. A. Brink, *Chemical Process Industries*, McGraw-Hill, New York (1977).

Shum, S. K., J. F. Davis, W. F. Punch III, and B. Chandrasekaran, "An Expert System Approach to Malfunction Diagnosis in Chemical Plants," *Comp. Chem. Eng.*, **12**(1), 27 (1988).

Venkatasubramanian, V., "CATDEX: An Expert System for Diagnosing a Fluidized Catalytic Cracking Unit," *CACHE Case-Studies Series: Knowledge-Based Systems in Process Engineering*, G. Stephanopoulos, ed., **1**, 41, Austin, TX (1988).

———, "Inexact Reasoning in Expert Systems: A Stochastic Parallel Network Approach," *Conf. on Artificial Intelligence Applications*, 13, Miami Beach, FL (Dec., 1985).

Venkatasubramanian, V., and S. H. Rich, "An Object-Oriented Two-Tier Architecture for Integrating Compiled and Deep-Level Knowledge for Process Diagnosis," *Comp. Chem. Eng.*, **12**(9/10), 903 (1988).

Werbos, P. J., "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," PhD Thesis in Applied Mathematics, Harvard University (1974).

Manuscript received May 22, 1989, and revision received Oct. 3, 1989.