A fuzzy-recurrent neural network (FRNN) has been constructed by adding some feedback connections to a feedforward fuzzy neural network (FNN). The FRNN expands the modeling ability of a FNN in order to deal with temporal problems. A basic concept of the FRNN is first to use process or expert knowledge, including appropriate fuzzy logic rules and membership functions, to construct an initial structure and to then use parameter-learning algorithms to fine-tune the membership functions and other parameters. Its recurrent property makes it suitable for dealing with temporal problems, such as on-line fault diagnosis. In addition, it also provides human-understandable meaning to the normal feedforward multilayer neural network, in which the internal units are always opaque to users. In a word, the trained FRNN has good interpreting ability and one-step-ahead predicting ability. To demonstrate the performance of the FRNN in diagnosis, a comparison is made with a conventional feedforward network. The efficiency of the FRNN is verified by the results.

**KeyWords:** Fault diagnosis, fuzzy neural networks, recurrent neural networks, time series prediction.

I. INTRODUCTION

Neural networks offer some very salient characteristics and properties, such as learning, associating, generalization, distributed parallel signal processing, and nonlinear input-output mapping. Thus, the fault diagnosis technologies based on neural networks can solve many intrinsic problems existing in the fault diagnosis domains of nonlinear systems, such as process modelling, knowledge expressing and acquiring, parallel reasoning, and so on. Neural networks based fault diagnosis approaches can be mainly categorized into three classes. Firstly, there are neural networks which act as classifiers and conduct fault diagnosis according to the pattern recognition principle. Secondly, there are neural networks which act as predictors and conduct fault diagnosis by evaluating the difference between measured values and predicted values. Thirdly, there are neural networks which act as knowledge-processing modules in fault diagnosis expert systems and extract knowledge from multi-sensors for entire expert systems. The neural networks which have been widely used are mainly BP, RBF, Hopfield, ART, SOFM, RNN, and FNN [1,2,3]. In contrast to the pure neural network or fuzzy system, the fuzzy neural network possesses the advantages of both; it brings the low-level learning and computational power of neural networks into fuzzy systems, and brings the high-level human-like thinking and reasoning of fuzzy systems into neural networks. However, a major drawback of the existing fuzzy neural networks is that their application domain is limited to static problems due to their inherent feedforward network structure. Hence, a recurrent neural network capable of solving temporal problems is needed.

Unlike the feedforward neural network, whose output is a function of its current inputs only and is limited to static mapping, the recurrent neural network can perform dynamic mapping. Recurrent neural networks are needed for problems where there exists at least one system state variable that cannot be observed. Most of the existing recurrent neural networks are obtained by adding trainable temporal elements to feedforward neural networks to make the output history-sensitive. The concept of incorporating fuzzy logic into a recurrent network has been pro-
posed in several papers recently [4,5,6,7]. Since fuzzy neural networks possess the many advantages mentioned above, it seems worthwhile to construct a recurrent network based on a fuzzy neural network. In this paper, we propose such a fuzzy-recurrent neural network. The proposed network can extend the application domain of the normal fuzzy neural networks to include temporal problems.

II. FUZZY-RECURRENT NEURAL NETWORKS

2.1 Structure of fuzzy-recurrent neural networks

The structure of the FRNN is shown in Fig.1. It is a six-layer fuzzy neural network embedded with dynamic feedback connections that bring the temporal processing ability into a feedforward fuzzy neural network. The nodes in layer 1 are input linguistic nodes that represent input linguistic variables. The nodes in layers 2 and 4 are, respectively, input term nodes and output term nodes, which act as membership functions representing the terms of the corresponding linguistic variables. Actually, a layer-2 node can either be a single node that performs a simple membership function (e.g., a triangle-shaped or bell-shaped function) or may be composed of multilayer nodes (a sub network that performs a complex membership function). Each node in layer 3 is a rule node that represents one fuzzy logic rule. Thus, all the layer-3 nodes form a fuzzy rule base. Links in layers 3 and 4 function as a connectionist inference engine, so the rule-matching process can be avoided. Layer-3 links define the preconditions of the rule nodes, and layer-4 links define the consequents of the rule nodes. Therefore, for each rule node, there is at most one link (maybe none) from some term node of a linguistic node. This is true for both precondition links (links in layer 3) and consequent links (links in layer 4). There are two output linguistic nodes for each output variable in layer 5. One is for training data (the desired output) to be fed into the network, and the other is for decision signals (actual output) to be pumped out of the network. There are two types of nodes in the feedback layer, called layer 6: square nodes called context nodes and circle nodes called feedback term node. The nodes in the feedback layer are fully connected to the nodes in layer 4. The links in layers 2 and 5 are fully connected to linguistic nodes and their corresponding term nodes. The arrow on the link indicates the normal signal flow direction when the network is in use after it has been built and trained.

The FRNN expands the basic ability of a fuzzy neural network in order to cope with temporal problems via the inclusion of some internal memories, called context nodes. From the perspective of fuzzy logic, these context nodes are expressed in the form of internal fuzzy reasoning. More precisely, with these context nodes, the network performs the following reasoning[4]:

Rule \( r \): IF \( x_1(t) \) is \( A_{r1} \) and \( \ldots \) and \( x_N(t) \) is \( A_{rN} \) and \( h_{r}(t) \) is \( G \) THEN \( y_1(t+1) \) is \( B_{r1} \) and \( \ldots \) and \( y_M(t+1) \) is \( B_{rM} \) and \( h_1(t+1) \) is \( v_{r1} \) and \( \ldots \) and \( h_R(t+1) \) is \( v_{rR} \),

\[(1)\]

where \( x_n (n = 1, \ldots, N) \) is the input variable; \( y_m (m = 1, \ldots, M) \) is the output variable; \( h_r (r = 1, \ldots, R) \) is the internal variable; \( A_{r1}, \ldots, A_{rN}, G, B_{r1}, \ldots, B_{rM} \) are fuzzy sets; \( v_{r1}, \ldots, v_{rR} \) are fuzzy singletons; \( N, M \) and \( R \) are, respectively, the number of input, output and internal variables (the number of internal variables, equal to the number of rules). To give a clear explanation of the dynamic reasoning, we decompose the above fuzzy rule into two parts, the external rule

![Fig. 1. Structure of the FRNN.](image-url)
and the internal rule, both of which form a hierarchical relation. The external rule is based on the following reasoning:

Rule $r_1$: IF $x_1(t)$ is $A_{i1}$ and ... and $x_d(t)$ is $A_{iN}$ and $h_i(t)$ is $G$ 
THEN $y_1(t + 1)$ is $B_{j1}$ and ... and $y_m(t + 1)$ is $B_{jM}$.

(2)

where the outputs are functions of the internal variables acting as internal memories. The internal variables themselves constitute a dynamic internal rule as follows:

Rule $r_2$: IF $x_1(t)$ is $A_{i1}$ and ... and $x_d(t)$ is $A_{iN}$ and $h_i(t)$ is $G$ 
THEN $h_i(t + 1)$ is $v_{i1}$ and ... and $h_i(t + 1)$ is $v_{iR}$.

(3)

The hierarchical and temporal relationship between the internal and external fuzzy rules can be easily recognized if we unfold the FRNN in the time domain as shown in Fig. 2.

Assume that the symbol $u_i^{(k)}$ denotes the $i$th input of a node in layer $k$; correspondingly, the symbol $z_i^{(k)}$ denotes the node output in layer $k$. We will next describe the functions of the nodes in each of the six layers of the FRNN [3].

**Layer 1.** The nodes in this layer only transmit input values to the next layer directly. That is,

$z_1^{(1)} = u_1^{(1)}$.

(4)

From this equation, the link weight in layer 1 is unity.

**Layer 2.** If we use a single node to perform a simple membership function, then the output function of this node should be this membership function. For example, for a bell-shaped Gaussian function, we have

$z_2^{(2)} = \exp\left\{-\frac{(u_2^{(2)} - m_j)^2}{\sigma_j^2}\right\}$

(5)

where $m_j$ and $\sigma_j$ are, respectively, the center (or mean) and the width (or standard deviation) of the Gaussian membership function of the $j$th term of the $i$th input linguistic variable $x_i$. Hence, the link weight in layer 2 can be interpreted as $m_{ij}$.

**Layer 3.** The links in this layer are used to perform pre-condition matching of the fuzzy logic rules. Hence, the rule nodes perform the fuzzy AND operation:

$z_3^{(3)} = z_1^{(1)} \prod_j u_j^{(1)}$.

(6)

The link weight in layer 3 is then unity.

**Layer 4.** The nodes in this layer have two operation modes: down transmission and up-down transmission. The links in layer 4 perform the simple summation instead of fuzzy OR operation to integrate the fired rules which have the same consequent:

$z_4^{(4)} = \sum_j u_j^{(4)}$.

(7)

Hence, the link weight is also unity. In the up-down transmission mode, the nodes in this layer and the links in layer 5 function in the same way as do those in layer 2 except that only a single node is used to perform a membership function for output linguistic variables.

**Layer 5.** There are two kinds of nodes in this layer. The first kind of node performs up-down transmission of training data that are being fed into the network. For this kind of node,

$z_5^{(5)}_{\text{up-down}} = y_j$.

(8)

The second kind of node performs down-up transmission of the decision signal output. These nodes and the layer-5 links attached to them act as the defuzzifier. If $\tilde{M}_j$ and $\tilde{\sigma}_j$ are, respectively, the center and the width of the Gaussian membership function of the $j$th output linguistic variable, then the following functions can be used to simulate the center of area defuzzification method:

$z_5^{(5)}_{\text{down-up}} = \frac{\sum_j (\tilde{M}_j \tilde{\sigma}_j) u_j^{(5)}}{\sum_j \tilde{\sigma}_j u_j^{(5)}}$.

(9)

Here, the link weight in layer 5 is $\tilde{M}_j \tilde{\sigma}_j$.

**Layer 6.** (Feedback layer): This layer calculates the value of the internal variable $h_i$ and the firing strength of the internal variable with respect to its corresponding membership function, where the firing strength contributes to the matching degree of a rule node in layer 3. The inputs to a context node come from all the output term nodes, and the output of its associated feedback term node is fed

![Fig. 2. Function block diagram of the FRNN unfolded with time.](Image)
to the rule nodes whose consequent is the output term node corresponding to this context node. The context node functions as a defuzzifier:

\[ h_j = \sum_i z_i^{(4)} w_{ji}, \]  

(10)

where \( w_{ji} \) is the link weight from the \( i \)th node in layer 4 to the \( j \)th internal variable. It represents a fuzzy singleton in the consequent part of a rule, and also as a fuzzy term of the internal variable. For an internal variable, a fuzzy singleton instead of a fuzzy membership function is used as its fuzzy term; a fuzzy membership function on an internal variable does not make much sense in the network due to the use of the local mean of maximum (LMOM) [8] defuzzification operation, where only the center of the Gaussian membership function is used. This is different from the situation for the input and output linguistic variables, where the widths of the fuzzy membership functions are used. Instead of using the weighted-sum of each rule’s outputs as the inference result, the following conventional average weighted-sum is used:

\[ z^{(6)} = \frac{1}{1 + e^{-h_j}}. \]  

(11)

This output is connected to the rule nodes in layer 3, which connect to the same output term node in layer 4. The outputs of the feedback term nodes contain the firing history of the fuzzy rules.

### 2.2 Learning algorithm of the FRNN

Two phases of learning, structure initializing and parameter learning, are used to construct the FRNN. It is well-known that an expert can summarize lots of knowledge into fuzzy IF-THEN rule forms during long-term observation and research on the process or plant. According to these rules, which are similar to Eq. (1), the structure can be easily constructed. The free parameters can also be initialized based upon process knowledge and operational experience with noise levels and nominal fluctuation levels of process variables. As soon as the initial structure has been constructed, we can then start to fine-tune the parameters based on on-line measurements.

Since the FRNN is a dynamic system with feedback connections, the parameter learning algorithm used in feedforward radial basis function networks or adaptive fuzzy systems cannot be applied to it directly. In addition, due to the on-line learning property of the FRNN, off-line learning algorithms for recurrent neural networks, like back propagation through time and time-dependent recurrent back propagation algorithms, cannot be applied here. Instead, the ordered derivative, which is a partial derivative whose constant and varying terms are defined using an ordered set of equations, is used here to derive our learning algorithm. The ordered set of equations is described in Eqs. (4)–(11), and our goal is to minimize the error function [3,4,8]:

\[ E(t) = \frac{1}{2}(y_m(t) - y_m^d(t))^2, \]  

(12)

where \( y_m^d(t) \) is the desired output and \( y_m(t) \) is the current output. For each training pattern, starting at the input nodes, a forward pass is used to compute the activity levels of all the nodes in the network to obtain the current output \( y_m(t) \). In the following, dependency on time \( t \) will be omitted unless emphasis on temporal relationships is required.

With the error function defined in Eq. (12) and the node functions defined in Eqs. (4)–(11), we can derive the update rules for the free parameters in the FRNN as follows.

1. Update rule of \( \tilde{m}_{mq} \) and \( \sigma_{mq} \) (the center and width of the output membership function):

\[ \tilde{m}_{mq}(t+1) = \tilde{m}_{mq}(t) - \eta \frac{\partial E(t)}{\partial \tilde{m}_{mq}}, \]  

(13)

where

\[ \frac{\partial E(t)}{\partial \tilde{m}_{mq}} = \frac{\partial E(t)}{\partial \tilde{c}^{(5)}} \cdot \frac{\partial \tilde{c}^{(5)}}{\partial \tilde{m}_{mq}} = \left( y_m(t) - y_m^d(t) \right) \cdot \frac{\tilde{\sigma}_{mq} \tilde{u}^{(5)}_q}{\sum_{q=1}^{M} \tilde{\sigma}_{mq} \tilde{u}^{(5)}_q}, \]

\[ \sigma_{mq}(t+1) = \sigma_{mq}(t) - \eta \frac{\partial E(t)}{\partial \sigma_{mq}}, \]

\[ \frac{\partial E(t)}{\partial \sigma_{mq}} = \frac{\partial E(t)}{\partial c^{(5)}} \cdot \frac{\partial c^{(5)}}{\partial \sigma_{mq}} = \left( y_m(t) - y_m^d(t) \right) \frac{\tilde{m}_{mq} \tilde{u}^{(5)}_q}{\sum_{q=1}^{M} \tilde{\sigma}_{mq} \tilde{u}^{(5)}_q} \left( \sum_{q=1}^{M} \tilde{\sigma}_{mq} \tilde{u}^{(5)}_q \right)^2. \]

2. Update the rule of \( m_{np} \) and \( \sigma_{np} \) (the center and width of the membership function in the precondition part):

\[ m_{np}(t+1) = m_{np}(t) - \eta \frac{\partial E(t)}{\partial m_{np}}, \]  

(14)

The value of \( \frac{\partial E(t)}{\partial m_{np}} \) is computed by
\[
\frac{\partial E}{\partial m_{np}} = (y_m(t) - y_m^d(t)) \sum_{r=1}^{M} \frac{\partial y_r(t)}{\partial c_r^{(3)}} \frac{\partial c_r^{(3)}}{\partial m_{np}}.
\]

where

\[
\frac{\partial y_r(t)}{\partial c_r^{(3)}} = \eta_{rq} \delta_{rq}^{(2)} \mu_r(t) + \zeta^{(6)} \mu_r(t) \cdot 2 \frac{x_r(t) - m_{np}}{\sigma_{np}} \cdot \delta_{pr},
\]

\[
\mu_r(t) = \exp \left( - \sum_{n=1}^{N} \delta_{pr} \left( \frac{x_r(t) - m_{np}}{\sigma_{np}} \right)^2 \right),
\]

\[
\frac{\partial c_r^{(6)}}{\partial m_{np}} = \frac{\exp(h_r(t-1)) \cdot \partial h_r(t-1)}{(1 + \exp(h_r(t-1)))^2},
\]

\[
\frac{\partial h_r(t-1)}{\partial m_{np}} = \sum_{q=1}^{M} \left( w_q(t-1) \sum_{r=1}^{R} \delta_{rq}^{(3)} \right).
\]

where \( \delta_{pr} = 1 \) if the \( p \)th node in layer 2 is connected to the \( r \)th node in layer 3 and \( \delta_{pr} = 1 \) if the \( r \)th node in layer 3 is connected to the \( q \)th node in layer 4; otherwise, they are equal to zero. As for \( \frac{\partial c_r^{(3)}}{\partial m_{np}} \), we can instead iterate it forward from the initial condition \( \frac{\partial c_r^{(3)}}{\partial m_{np}} \bigg|_{t=0} = 0 \). Similarly,

\[
\sigma_{np}(t+1) = \sigma_{np}(t) - \eta_{np} \frac{\partial E(t)}{\partial \sigma_{np}}.
\]

The value of \( \frac{\partial E(t)}{\partial \sigma_{np}} \) is computed by

\[
\frac{\partial E(t)}{\partial \sigma_{np}} = (y_m(t) - y_m^d(t)) \sum_{r=1}^{M} \frac{\partial y_r(t)}{\partial c_r^{(3)}} \frac{\partial c_r^{(3)}}{\partial \sigma_{np}},
\]

where

\[
\frac{\partial c_r^{(3)}}{\partial \sigma_{np}} = \frac{\partial c_r^{(6)}}{\partial \sigma_{np}} \cdot \mu_r(t) + \zeta^{(6)} \mu_r(t) \cdot 2 \frac{x_r(t) - m_{np}}{\sigma_{np}} \cdot \delta_{pr},
\]

\[
\frac{\partial c_r^{(6)}}{\partial \sigma_{np}} = \frac{\exp(h_r(t-1)) \partial h_r(t-1)}{(1 + \exp(h_r(t-1)))^2},
\]

\[
\frac{\partial h_r(t-1)}{\partial \sigma_{np}} = \sum_{q=1}^{M} \left( w_q(t-1) \sum_{r=1}^{R} \delta_{rq}^{(3)} \right).
\]

3. Update the rule of \( w_q^{(l)} \) (the memory weight parameter in the feedback layer):

\[
w_q^{(l)}(t+1) = w_q^{(l)}(t) - \eta_{l_{wq}} \frac{\partial E(t)}{\partial w_q^{(l)}}.
\]

The value of \( \frac{\partial E(t)}{\partial w_q^{(l)}} \) is computed by

\[
\frac{\partial E(t)}{\partial w_q^{(l)}} = (y_m(t) - y_m^d(t)) \sum_{r=1}^{M} \frac{\partial y_r(t)}{\partial c_r^{(3)}} \frac{\partial c_r^{(3)}}{\partial \sigma_{np}}
\]

and

\[
\frac{\partial c_r^{(3)}}{\partial \sigma_{np}} = \mu_r \frac{\partial c_r^{(6)}}{\partial \sigma_{np}},
\]

\[
\frac{\partial c_r^{(6)}}{\partial \sigma_{np}} = \frac{\exp(h_r(t-1)) \partial h_r(t-1)}{(1 + \exp(h_r(t-1)))^2},
\]

\[
\frac{\partial h_r(t-1)}{\partial \sigma_{np}} = \delta_{hr}(t-1),
\]

where \( \delta_{hr} = 1 \) if \( l = r \); otherwise, it is equal to zero.

### III. APPLICATION OF THE FRNN IN ON-LINE FAULT DIAGNOSIS

To verify the performance of the FRNN in fault diagnosis, an example is presented in this section, and performance comparisons with a feedforward neural network are made. Consider the following dynamical mathematical model for a nonlinear synchronous alternating current dynamo [9]:

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3
\end{bmatrix} =
\begin{bmatrix}
x_2 \\
-0.27x_2 - 12.01u_1 \sin x_3 + 24.02 \sin 2x_1 + 39.19u_1 \\
-0.32x_3 + 19 \cos x_1 + u_2
\end{bmatrix},
\]

\[
\begin{bmatrix}
y_1 \\
y_2 \\
y_3
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} + \begin{bmatrix}
\xi_1 \\
\xi_2 \\
\xi_3
\end{bmatrix},
\]

where \( u_1 \) and \( u_2 \) are the inputs; \( x_1, x_2 \) and \( x_3 \) are state variables; \( y_1 \) and \( y_2 \) are the outputs; \( \xi_1 \) and \( \xi_2 \) are white noise.

As mentioned above, experts can summarize lots of knowledge into fuzzy IF-THEN rule forms during long-term observation and research on the process or plant. According to historical records and expert suggestions, we have established six fuzzy rules for the dynamo, and each
linguistic variable has to be considered as three terms. Thus, the structure can be easily constructed, and the free parameters can also be easily initialized based upon process knowledge and operational experience with noise levels and nominal fluctuation levels of process variables. As soon as the initial structure has been constructed, we can then start to fine-tune the parameters based on training data. Training data are on-line measurements covering the events of the faults being considered and the normal operating conditions. In our application, all the training data were sampled approximately every 0.1s, and then they were normalized in the range (0, 1). As soon as the desired prediction accuracy had been achieved by means of training, we simulated three faults. The first fault was an incipient fault, which was achieved by linearly changing the structure parameter by about 20% from time-step 120 to 150. The second fault was an abrupt amplitude fault, which was achieved by suddenly changing the input amplitude by about 40% at time-step 260. The third fault was an abrupt frequency fault, which was achieved by suddenly changing the input frequency about 5 times at time-step 340. The test results are shown in Fig. 3. It is obvious that the FRNN based approach can detect fault much earlier than the BP based method.

In addition, it also provides human-understandable meanings for the normal feedforward multilayer neural network, in which the internal units are always opaque to users. The fuzzy rules are fine-tuned as follows:

Rule 1: IF $x_1(t)$ is $\mu(0.41, 0.52)$ and $x_2(t)$ is $\mu(0.19, 0.24)$ and $h_4(t)$ is $G$ THEN $y_1(t+1)$ is $\mu(0.63, 0.35)$ and $y_2(t+1)$ is $\mu(0.25, 0.47)$ and $h_4(t+1)$ is $-0.39$ and $h_5(t+1)$ is $-0.48$ and $h_6(t+1)$ is $1.16$ and $h_7(t+1)$ is $1.67$ and $h_8(t+1)$ is $1.54$ and $h_9(t+1)$ is $-0.44$

Rule 6: IF $x_1(t)$ is $\mu(0.65, 0.13)$ and $x_2(t)$ is $\mu(0.69, 0.52)$ and $h_4(t)$ is $G$ THEN $y_1(t+1)$ is $\mu(0.87, 0.36)$ and $y_2(t+1)$ is $\mu(0.44, 0.41)$ and $h_4(t+1)$ is $0.21$ and $h_5(t+1)$ is $0.93$ and $h_6(t+1)$ is $-0.74$ and $h_7(t+1)$ is $0.88$ and $h_8(t+1)$ is $1.57$ and $h_9(t+1)$ is $0.16$

IV. CONCLUSION

An FRNN with process or expert knowledge and online parameter learning capability along with its application to fault diagnosis has been presented in this paper. Basically, this network is constructed by extending the powerful ability of a fuzzy neural network to deal with temporal problems. The FRNN itself realizes dynamic fuzzy reasoning by creating recursive fuzzy rules, which are fine-tuned automatically during on-line operation via concurrent structure and parameter learning. In the FRNN based diagnosis approach, the network is used to learn the relationships between residuals and faults based on the process data. After training is finished, the learned residual-fault relationships are stored as the trained network parameters, and the network is used to identify whether or not a fault has occurred. The test results show that the diagnosis approach based on the FRNN is more effective than the BP based method.

REFERENCES

Zhao Xiang was born in 1973, Ph.D. He works in Logistics Engineering University of P.L.A. from 1991 and acts as a visiting scholar in Department of Automation, Tsinghua University from March 1999. Now his research interests include intelligent fault diagnosis based on neural network, information fusion, evolution computing and so on.

Xiao Deyun was born in 1945, graduated from Tsinghua University in 1970. Now he is a professor and Ph.D. advisor in Department of Automation, Tsinghua University. His main area of interest and expertise is identification modeling, fault diagnosis, multi-sensor fusion, computer control, CIMS in process industry and so on.