A Recurrent Fuzzy Cellular Neural Network System With Automatic Structure and Template Learning

Chin-Teng Lin, *Senior Member, IEEE*, Chun-Lung Chang, and Wen-Chang Cheng

***Abstract*—It is widely accepted that using a set of cellular neural networks (CNNs) in parallel can achieve higher level information processing and reasoning functions either from application or biologics points of views. Such an integrated CNN system can solve more complex intelligent problems. In this paper, we propose a novel framework for automatically constructing a multiple-CNN integrated neural system in the form of a recurrent fuzzy neural network. This system, called recurrent fuzzy CNN (RFCNN), can automatically learn its proper network structure and parameters simultaneously. The structure learning includes the fuzzy division of the problem domain and the creation of fuzzy rules and CNNs. The parameter learning includes the tuning of fuzzy membership functions and CNN templates. In the RFCNN, each learned fuzzy rule corresponds to a CNN. Hence, each CNN takes care of a fuzzily separated problem region, and the functions of all CNNs are integrated through the fuzzy inference mechanism. A new online adaptive independent component analysis mixture-model technique is proposed for the structure learning of RFCNN, and the ordered-derivative calculus is applied to derive the recurrent learning rules of CNN templates in the parameter-learning phase. The proposed RFCNN provides a solution to the current dilemma on the decision of templates and/or fuzzy rules in the existing integrated (fuzzy) CNN systems. The capability of the proposed RFCNN is demonstrated on the real-world defect inspection problems. Experimental results show that the proposed scheme is effective and promising.**

***Index Terms*—Cellular neural networks (CNN) template design, defect inspection, fuzzy clustering, fuzzy neural network (FNN), independent component analysis (ICA), ordered derivative, recurrent neural network.**

I. Introduction

A CELLULAR neural network (CNN) [1], [2] is a locally interconnected analog processor array arranged to a regular two-dimensional (2-D) grid. Its 2-D inputs and outputs make it very suitable for image processing. It possesses some important characteristics such as efficient real-time processing capability and feasible very large-scale integration (VLSI) implementation. A CNN has a space invariant local interconnection structure associated with 19 free parameters (neighborhood within a $\text{radius} = 1$). This parameter set called template exclusively determines its dynamic behavior. The CNN has been used to mimic the local function of biological neural circuits, especially the human visual pathway system [3]. According to a current biological study [4], mammalian visual systems process the world through a set of separate parallel channels. Each subchannel can be regarded as a unique CNN. The output of these subchannels is then combined to form the new channel responses. As a result, it is widely accepted that using a set of CNNs in parallel can achieve higher level information processing and reasoning functions either from biologics or application points of views. Such an integrated CNN system can solve more complex intelligent problems.

For designing an integrated CNN system, in addition to the determination of a set of templates, another kernel problem is the way of integration. To solve this problem, the fuzzy inference system (FIS) is gaining attention. The FIS is a popular computing framework based on the concept of fuzzy set theory, fuzzy IF-THEN rules, and fuzzy reasoning. With crisp inputs and outputs, FIS implements a nonlinear mapping from its input space to output space by a number of IF-THEN rules. It is very useful in image processing when it is difficult to specify, in a crisp mathematical form, the operation that is needed to yield a satisfying result from a complex image. For example, the boundary detection of different regions strongly depends on a subjective decision, especially in medical image. It cannot be clearly defined what is an edge-like and what is a noise-like pattern. In many cases, both statements might be true, therefore, a fuzzy-type linguistic description of all patterns is better than a crisp set approach. Therefore, FIS can play an important role to integrate a set of CNNs into a system.

To make a CNN or a set of CNNs having the ability of reasoning functions, several fuzzy-based CNN models were proposed [5]–[9], which are fuzzy CNN (FCNN) proposed by Yang *et al.* [5] and Yang and Yang [6], and fuzzy reasoning implemented on CNN proposed by Balsi and Voci [7], [8]. To make a set of CNNs in parallel achieve higher level information processing, several integrated CNN systems are proposed [4], [9]–[11], which are cellular neuro-fuzzy networks (CNFNs) proposed by Colodro and Torralba [9], and fuzzy-type CNN proposed by Rekeczky *et al.* [10], [11] and Szatmári *et al.* [4]. The common drawbacks of these approaches are that the corresponding templates cannot be learned and the fuzzy rules must be obtained by domain experts. Although according to Nossek's survey [12], the template coefficients of a CNN can be found by design [12], [13] or by learning [12], [14], these techniques cannot be applied to the design or learning of an integrated CNN system directly.

An observation on the works of Colodro *et al.* [9], Rekeczky *et al.* [10], and Szatmári *et al.* [4], they have two common characteristics. First, they all used many CNNs in parallel to solve a complex problem such as edge detection with impulse noise, the detection of fuzzy boundary, and features extraction, etc.

Manuscript received July 30, 2003; revised January 10, 2004. This work supported by the Brain Research Center, University System of Taiwan, under Grant 92B-711. This paper was recommended by Guest Editor A. Zarándy.

The authors are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: ctlin@mail.nctu.edu.tw; VincentChang@itri.org.tw; wcc.ece88g@nctu.edu.tw).

Digital Object Identifier 10.1109/TCSI.2004.827622

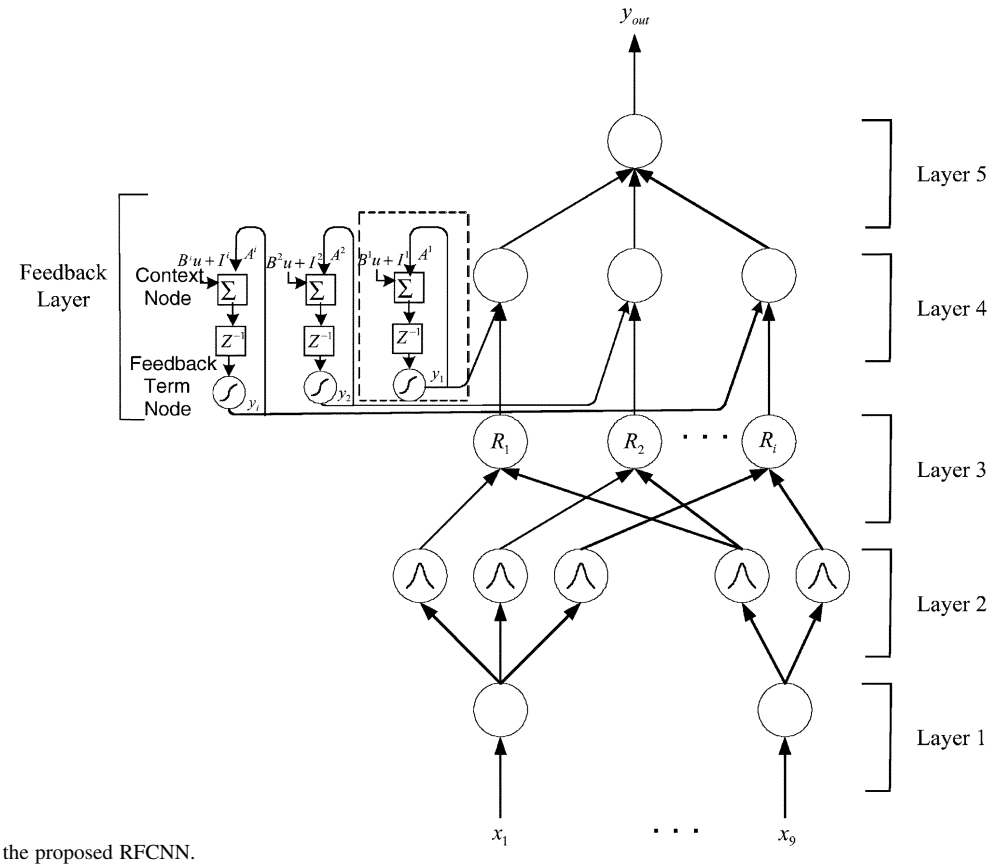

Fig. 1. Structure of the proposed RFCNN.

Second, they all used an FIS to make a decision. For building an FIS, we have to specify the fuzzy sets, fuzzy operators and the knowledge base. However, the existing methods [4], [9], [11] all need to take the fuzzy rules manually by domain experts, which is difficult, even for domain experts, to examine all the input–output data from a complex system to find a number of proper fuzzy rules. In addition, they all need to assign the corresponding templates of CNNs in advance (i.e., templates cannot be learned). To cope with these drawbacks, we propose a novel framework for automatically constructing a multiple-CNN integrated neural system in the form of a recurrent fuzzy neural network (FNN). This system, called recurrent fuzzy CNN (RFCNN), can automatically learn its proper network structure and parameters simultaneously. The structure learning includes the fuzzy division of the problem domain and the creation of fuzzy rules and CNNs. The parameter learning includes the tuning of fuzzy membership functions and CNN templates. In the RFCNN, each learned fuzzy rule corresponds to a CNN. Hence, each CNN takes care of a fuzzily separated problem region, and the functions of all CNNs are integrated through the fuzzy inference mechanism.

The RFCNN is constructed in the form of a recurrent FNN. Two important learning tasks of a FNN are the structure identification and the parameters identification [15]–[19]. The structure identification is the partition of the input–output space [20]–[23], which influences the number of generated fuzzy rules, each corresponding to a CNN. Efficient partition of input–output data will result in faster convergence and better performance for FNN. In this paper, a new online adaptive independent component analysis (ICA) mixture-model technique is proposed for the structure learning of the RFCNN. Basically, ICA finds directions in the input space which lead to independent components instead of just uncorrelated ones, as principle component analysis (PCA) does [24], [25], so it reduces not only the number of rules (i.e., CNN) but also the number of membership functions under a pre-specified accuracy requirement dynamically. In the parameter learning of the RFCNN, the ordered derivative calculus is applied to derive the recurrent learning rules due to the recurrent structure of the RFCNN inherited from CNNs [1], [2]. The derived rules can learn the CNN templates and other parameters in the RFCNN efficiently. The proposed RFCNN provides a solution to the current dilemma on the decision of templates and/or fuzzy rules in the existing integrated (fuzzy) CNN systems. It has been applied to solve the real-world defect inspection problems, which contain multiple types of defects (faults) with different features on a single image. Experimental results successfully demonstrate that the proposed scheme is very effective and promising.

The paper is organized as follows. Section II describes the structure and functions of the proposed RFCNN. Section III describes the online structure and parameters learning algorithm for the RFCNN. Section IV gives experimental results and discussions. Finally, conclusions are summarized in the last section.

II. Structure of the RFCNN

In this section, the structure of the proposed RFCNN shown in Fig. 1 is introduced. For clarity, we consider a CNN, with

time constant $= 1$, time step $= 1$, and neighborhood within a radius $= 1$, which is characterized by the following templates:

$$A^i = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a^i_{0,0} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
$$B^i = \begin{bmatrix} b^i_{-1,-1} & b^i_{-1,0} & b^i_{-1,1} \\ b^i_{0,-1} & b^i_{0,0} & b^i_{0,1} \\ b^i_{1,-1} & b^i_{1,0} & b^i_{1,1} \end{bmatrix}$$
$$I^i = z^i \tag{1}$$

where A^i, B^i, and z^i is the feedback template, control template, and bias of the ith CNN, respectively. By defining a CNN as above, the six-layered RFCNN network will realize a fuzzy model of the following form:

$$\begin{aligned} &\text{Rule } i: \text{IF} x_1 \text{ is } M^i_1 \text{ and } \ldots x_j \text{ is } M^i_j \ldots \text{ and } x_9 \text{ is } M^i_9 \\ &\text{THEN } y_i(t+1) \text{ is } f'\left(A^i y_i(t) + B^i_u(t) + z^i(t)\right) \end{aligned} \tag{2}$$

or

$$\begin{aligned} &\text{Rule } i: \text{IF } x_1 \text{ is } M^i_1 \text{ and } \ldots x_j \text{ is } M^i_j \ldots \text{ and } x_9 \text{ is } M^i_9 \\ &\text{THEN } y_i(t+1) \text{ is } f'\left(a^i_{0,0} y_i(t) + b^i_{-1,-1} x_1(t)\right. \\ &\quad \left. + b^i_{-1,-0} x_2(t) + \cdots + b^i_{1,1} x_9(t) + z^i(t)\right) \end{aligned} \tag{3}$$

where the current input vector is $u = \mathbf{x}_t = [x_1, \ldots, x_9]^T$, $A^i y_i(t)$ is $a^i_{0,0} y_i(t), B^i u(t)$ is $\sum b^i u = b^i_{-1,-1} x_1(t) + b^i_{-1,-0} x_2(t) + \cdots + b^i_{1,1} x_9(t), f'$ is a sigmoid function, M^i_j is a fuzzy set, and $a^i_{0,0}, b^i_{k,l}$, and z^i are consequent parameters representing feedback template, control template, and bias of the ith CNN, respectively. The number of input dimension of the RFCNN will be $(2r+1)^2$ if the neighborhood of a CNN cell is within a radius $= r$. As shown in (3), we focus on uncoupled CNN cells in this paper. With this six-layered network structure of the RFCNN, we shall define the function of each node and use the proposed online ICA mixture model described in the next section to construct the structure of the RFCNN.

The RFCNN consists of nodes, each of which has some finite "fan-in" of connections represented by weight values from other nodes and "fan-out" of connections to other nodes. Associated with the fan-in of a node is an integration function f, which serves to combine information, activation, or evidence from other nodes. This function provides the net input for this node

$$\text{node}-\text{input} = f\left[u^{(k)}_1, u^{(k)}_2, \ldots, u^{(k)}_p; w^{(k)}_1, w^{(k)}_2, \ldots, w^{(k)}_p\right] \tag{4}$$

where $u^{(k)}_1, u^{(k)}_2, \ldots, u^{(k)}_p$ are inputs to this node and $w^{(k)}_1, w^{(k)}_2, \ldots, w^{(k)}_p$ are the associated link weights. The superscript (k) in (4) indicates the layer number. This notation will also be used in the following equations. A second action of each node is to output an activation value as a function of its net input

$$\text{node}-\text{output}^{(k)} = o^{(k)}_i = a^{(k)}(\text{node}-\text{input}) = a^{(k)}(f) \tag{5}$$

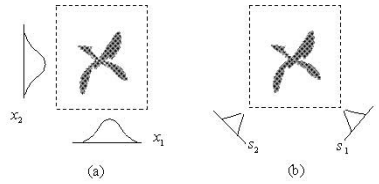

Fig. 2. Transformation by the online ICA mixture model for the proposed RFCNN. (a) Regions covered by the original axes. (b) Covered regions by the independent axes obtained by the online ICA mixture model transformation.

where $a(\cdot)$ denotes the activation function. We shall next describe the functions of the nodes in each of the six layers of the RFCNN, which include five feedforward layers and one feedback layer.

Layer 1: No computation is done in this layer. Each node in this layer, which corresponds to one input variable, only transmits input values to the next layer directly. That is

$$f = u^{(1)}_i$$

and

$$o^{(1)}_i = a^{(1)}(f) = f. \tag{6}$$

From the above equation, the link weight in layer one $[w^{(1)}_i]$ is unity.

Layer 2: Each node in this layer corresponds to one linguistic value (small, large, etc.) of one of the input variables in Layer 1. In other words, the membership value which specifies the degree to which an input value belongs a fuzzy set is calculated in Layer 2. There are many choices for the types of membership functions for use, such as triangular, trapezoidal, or Gaussian ones. In this paper, the membership functions are determined by the online ICA mixture model, which are either super-Gaussian function or sub-Gaussian function. It is noted that the output $\mathbf{x}$ from Layer 1 is projected into the independent axes obtained by the online ICA mixture model (as shown in Fig. 2) such that

$$\mathbf{s}_i = \mathbf{B}_i \mathbf{x} \tag{7}$$

where $\mathbf{B}_i$ is the basis matrix determined by the online ICA mixture model, $i = 1, 2, \ldots, J$, and J is the number of clusters. That is, if the input data are classified into J clusters, the number of rules will be J.

With the choice of non-Gaussian membership function, the operation performed in this layer is

$$\mathbf{u}^{(2)}_i = \mathbf{s}_i,$$
$$f\left[u^{(2)}_{ij}\right] = p\left(u^{(2)}_{ij}\right)$$

where

$$p(u_{ij}) \propto \exp(-|u_{ij}|), \text{for super-Guassian}$$
$$p(u_{ij}) \propto \exp\left(\log(\cosh(u_{ij})) - \frac{u^2_{ij}}{2}\right), \text{for sub-Gaussian}$$

and

$$o^{(2)}_i = a^{(2)}(f) = f \tag{8}$$

where u_{ij} is the transformed value of the jth term of the ith input variable $\mathbf{x}_i$. The transformation can be regarded as a change of

input coordinates, where the parameters of each membership function are kept unchanged, i.e., the center and the width of each membership function on the new coordinate axes are the same as the old ones.

Layer 3: A node in this layer represents one fuzzy logic rule and performs precondition matching of a rule. Here, we use the following AND operation for each Layer-3 node

$$f\left[u_i^{(3)}\right] = \prod_i u_i^{(3)}$$

and

$$o_i^{(3)} = a^{(3)}(f) = f. \tag{9}$$

The link weight in the Layer 3 $[w_i^{(3)}]$ is unity. The output f of a Layer-3 node represents the firing strength of the corresponding fuzzy rule.

Layer 4: This layer is called the consequent layer. Different nodes in Layer 3 may be connected to the same node in Layer 4, meaning that the same consequent fuzzy set is specified for different rules. One of the inputs to each node is the output delivered from Layer 3 (firing strength) and the other inputs are CNN related inputs $a^{(6)}$, which are the output of feedback term node. The feedback term node will be described in the feedback layer part in this section. Combining the two kinds of inputs in Layer 4, we obtain the whole function performed by this layer as

$$f\left[u_i^{(4)}\right] = u_i^{(4)}$$

and

$$\begin{aligned} o_i^{(4)} &= a^{(4)}(f) = a^{(6)} \cdot f \\ &= \text{sigmoid}\left(A^i y_i(t) + B^i u(t) + z^i(t)\right) \cdot f \end{aligned} \tag{10}$$

where A^i, B^i, z^i is the feedback template, control template, bias of the ith CNN, respectively, as defined in (1), and $\text{sigmoid}(\ )$ is a sigmoid function, as defined in (13).

Layer 5: Each node in this layer corresponds to one output variable. The node integrates all the actions recommended by Layer 4 and acts as a defuzzifier with

$$f\left[u_i^{(5)}\right] = \sum_i u_i^{(5)}$$

and

$$y_{\text{out}} = a^{(5)}(f) = f. \tag{11}$$

Feedback Layer: As shown as Fig. 1, this self-feedback layer characterizes the consequents of the RFCNN as a CNN template. Two types of nodes are used in this layer, the square node named as *context node* and the circle node named as *feedback term node*, where each context node is associated with a feedback term node. The number of context nodes (and thus the number of feedback term nodes) is the same as that of output term nodes in layer 4. The inputs to a context node are from its corresponding output term nodes $(y_i(t))$, the input variables from Layer 1 $(u(t) = \mathbf{x}_t(t) = [x_1, \ldots, x_9]^T)$, and template bias $(z^i(t))$. The output of its associated feedback term node is fed to the original node in layer 4. The context node functions as the state (the summation of input part) of the ith CNN

$$x^i(t+1) = A^i y_i(t) + B^i u(t) + z^i(t). \tag{12}$$

As to the feedback term node, the membership function $f(u) = 1/(1+e^{-u})$ is used to approximate piecewise-linear function used in CNN. With this choice, the feedback term node evaluates the output by

$$a^{(6)} = \frac{1}{1+e^{-x^i}}. \tag{13}$$

This output is connected to its corresponding node in layer 4, which characterizes the consequents of the RFCNN as a CNN template.

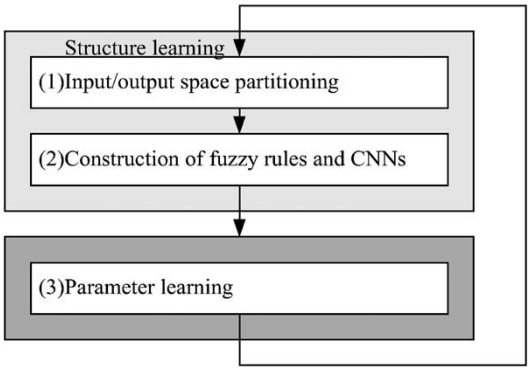

Fig. 3. Learning algorithm for the proposed RFCNN.

III. Learning Algorithms for RFCNN

Two types of learning, structure and parameter learning, are used concurrently for the RFCNN. The structure learning includes both the precondition and consequent structure identification of a fuzzy IF-THEN rule. In the RFCNN, the structure learning includes the fuzzy division of the problem domain (precondition structure identification), and the creation of fuzzy rules and CNNs (consequent structure identification). The precondition structure identification corresponds to the input–space partitioning and can be formulated as a combinational optimization problem with the following two objectives: to reduce the number of rules generated and to reduce the number of fuzzy sets on the universe of discourse of each input variable. As to the consequent structure identification, the main task is to decide when to generate a new consequent term (or a new CNN) for the output variable. In our system, we propose an online ICA mixture model to realize the precondition and consequent structure identification part of the RFCNN.

For the parameter learning, the parameters of each CNN template in the consequent parts are adjusted by the ordered derivative algorithm to minimize a given cost function. The parameters in the precondition part are adjusted by the online ICA mixture model algorithm. The RFCNN can be used for normal operation at any time during the learning process without repeated training on the input–output patterns when online operation is required. There are no rules (i.e., no nodes in the network except the input–output nodes) in this network initially. They are created dynamically as learning proceeds upon receiving online incoming training data by performing the following learning processes simultaneously (see Fig. 3).

As shown in Fig. 3, learning processes (1) and (2) belong to the structure learning phase and (3) belongs to the parameter learning phase. The details of these learning processes are described in the rest of this section.

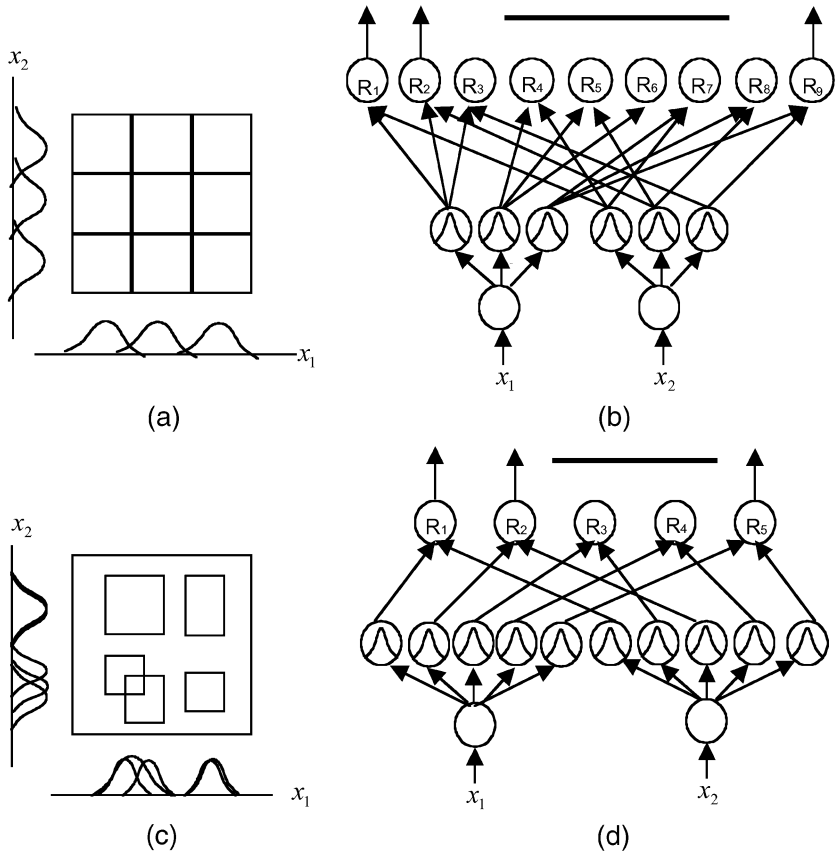

Fig. 4. Fuzzy partitions of 2-D input space. (a) Grid-based partitioning. (b) IF-THEN rules based on grid-based partitioning. (c) Clustering-based partitioning. (d) IF-THEN rules based on clustering-based partitioning.

A. Input–Output Space Partitioning

Efficient partition of input–output data will result in faster convergence and better performance for the RFCNN. The most direct way is to partition the input space into grid types and each grid represents a fuzzy IF-THEN rule [see Fig. 4(a)]. This is called grid-based partitioning. The major problem of such kind of partition is that the number of fuzzy rules (and thus, the number of CNNs) increases exponentially if the number of input variables or that of partition increases. A flexible partition method, the clustering-based approach, which clusters the input training vectors in the input space, will reduce the rule and CNN numbers [20]–[23]. In fact, by observing the projected membership functions in Fig. 4(c), although the number of membership functions in Fig. 4(d) is more than that in Fig. 4(b), there are only five rules in Fig. 4(d); however, there are nine rules in Fig. 4(b). By observing the projected membership functions in Fig. 4(c), we find that some membership functions projected from different clusters have high similarity degrees. These highly similar membership functions can be checked and merged by similarity measure. In this paper, we propose a clustering method based on a new online ICA mixture model to provide a better partition of the input–output space for the proposed RFCNN. The background and algorithm of the proposed online ICA mixture model for clustering will be described in the following subsections.

1) ICA Mixture Model: Several methods for input space partition have been proposed to cluster the input training vectors in the input space, such as Kohonen learning rule, hyperbox method, product–space partitioning, fuzzy c-mean method, electromagentic algorithm, etc., [26]–[29]. Those methods are usually based on Gaussian membership functions. In general, the observed data can be categorized into several mutually exclusive classes [30]. When the data in each class are modeled as multivariate Gaussian, it is called a Gaussian mixture model (GMM) which is widely used throughout the fields of machine learning and statistics. One major drawback of GMMs is that if the dimension d of the problem space increases, the size of each covariance matrix, d^2, becomes prohibitively large. This problem has been solved by Tipping and Bishop [31] who replaced each Gaussian with a probabilistic principal component analysis (PCA) model. This allowed the dimensionality of each covariance to be effectively reduced while maintaining the richness of the model class. ICA [24] is a technique that exploits higher-order statistical structure of the data, which has recently gained attention due to its successful applications to signal processing problems including speech enhancement, discrete signal processing and image processing, etc. The goal of ICA is to linearly transform the data such that the transformed variables are as statistically independent from each other as possible. Basically, it finds direction in the input space which lead to independent components instead of just uncorrelated ones as PCA does, so it can be used to reduce not only the number of rules but also the number of membership functions under a pre-specified accuracy requirement dynamically.

Another drawback of GMMs is that it is based on Gaussian function. In some situation, it could not be separated from each other. It is generalized by assuming the data in each class are generated by a linear combination of independent non-Gaussian source [33]. This model is called the ICA mixture model. This allows modeling of classes with non-Gaussian structure such as platykurtic or leptokurtic probability density functions, and the model uses the gradient ascent method to maximize the log-likelihood function. In previous applications, this approach showed improved performance in data classification problems [34] and learning efficient codes for representing different types of images [25]. The advantage of this model is that the input data with increasing numbers of classes can provide greater flexibility in modeling structure and in finding more features compared with GMMs or standard ICA algorithms. Although the ICA mixture model has many advantages, its cluster number should be given beforehand and the learning scheme is only suitable for off-line instead of online operation. Therefore, in the following section, we shall propose an online ICA mixture model to provide better dynamic partitioning of the input–output space for the proposed RFCNN.

2) Online ICA Mixture Model for Dynamic Clustering: The proposed online ICA mixture model is derived from the conventional ICA mixture model. To enable the online operation, we will define a criterion to determine whether the number of clusters should be increased or not for any incoming training pattern. For each incoming pattern to the RFCNN, the resulting firing strength of a fuzzy rule can be interpreted as the degree that the incoming pattern belongs to the corresponding cluster. This likelihood can be represented as

$$F^j(\mathbf{x}_t) = \ln p(\mathbf{x}_t \,|\, C_j) \tag{14}$$

where $\mathbf{x}_t$ denotes the incoming pattern at time t, and $\ln p(\mathbf{x}_t \,|\, C_j)$ is the log likelihood value indicating the degree that the input data, $\mathbf{x}_t$, belongs to the jth cluster for $j \in [1, J]$.

Now, we assume that the number of clusters at time t is $J(t)$. Then, the total probability at time t is

$$p(\mathbf{x}_t) = \sum_{j=1}^{J(t)} p(\mathbf{x}_t \,|\, C_j) p_t(C_j). \tag{15}$$

Therefore, the posterior probability is

$$p(C_j \,|\, \mathbf{x}_t) = \frac{p(\mathbf{x}_t \,|\, C_j) p_{t-1}(C_j)}{p(\mathbf{x}_t)} \tag{16}$$

where $p_{t-1}(C_j)$ is the prior probability at preceding time, which can be obtained by former calculation result of the jth cluster. Hence, the probability $p_t(C_j)$ at this moment can be calculated by the following:

$$\begin{aligned} p_t(C_j) &= \frac{1}{t} \sum_{i=1}^{t} p(C_j \,|\, \mathbf{x}_i) \\ &= \frac{1}{t} \left[\sum_{i=1}^{t-1} p(C_j \,|\, \mathbf{x}_i) + p(C_j \,|\, \mathbf{x}_t) \right] \\ &= \frac{1}{t} \left[(t-1) \cdot p_{t-1}(C_j) + p(C_j \,|\, \mathbf{x}_t) \right]. \end{aligned} \tag{17}$$

Then, the posterior probability $p(C_j \,|\, \mathbf{x}_t)$ in (16) can be obtained.

Based on the above derivation, we can obtain the following criterion for the generation of a new fuzzy rule (i.e., a new CNN). Let $\mathbf{x}_t$ be the newly incoming pattern at time t. Defining

$$F^{J_{\max}}(\mathbf{x}_t) = \max_{1 \le j \le J(t)} F^j(\mathbf{x}_t). \tag{18}$$

If $F^{J_{\max}}(\mathbf{x}_t) < F$, then, a new rule is generated, where F is a pre-specified threshold value that decays during the learning process. Once a new rule is generated, the next step is to assign initial values of the corresponding membership functions. If $F^{J_{\max}}(\mathbf{x}_t) \ge F$, a new incoming data is added to an existed cluster and we have to update the parameters of each cluster such as mean $(\mathbf{M}_j)$, covariance matrix $(\mathbf{Cov}_j)$, and the criterion of data distribution $(k_{j,p})$ that determines if the distribution of data is super-Gaussian or sub-Gaussian with the previous calculation results. They are defined in (19)–(21), shown at the bottom of the next page. In these, the function $k_{j,p}(t)$ is defined as the function of criterion which allows for automatic switching between super-Gaussian and sub-Gaussian models and (21) can be further derived as

$$k_{j,p}(t) = \operatorname{sign} \left\{ \frac{T_1(t) T_2(t)}{\sum_{i=1}^{t} p(C_j \,|\, \mathbf{x}_i)} - \frac{T_3(t)}{\sum_{i=1}^{t} p(C_j \,|\, \mathbf{x}_i)} \right\} \tag{22}$$

where

$$\begin{aligned} T_1(t) &= \sum_{i=1}^{t} p(C_j \,|\, \mathbf{x}_i) \operatorname{sech}^2(y_{j,p}(i)) \\ &= T_1(t-1) + p(C_j \,|\, \mathbf{x}_t) \operatorname{sech}^2(y_{j,p}(t)), \\ T_2(t) &= \sum_{i=1}^{t} p(C_j \,|\, \mathbf{x}_i) (y_{j,p}(i))^2 \\ &= T_2(t-1) + p(C_j \,|\, \mathbf{x}_t) y_{j,p}(t), \\ T_3(t) &= \sum_{i=1}^{t} p(C_j \,|\, \mathbf{x}_i) y_{j,p}(i) \cdot \tanh^2(y_{j,p}(i)) \\ &= T_3(t-1) + p(C_j \,|\, \mathbf{x}_t) y_{j,p}(t) \tanh^2(y_{j,p}(t)). \end{aligned} \tag{23}$$

Finally, the independent axes $\mathbf{B}_j(t)$, representing the axis of the jth cluster, can be obtained by the following formulations:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{B}_j} \ln p(\mathbf{x}_t) &= p(C_j \,|\, \mathbf{x}_t) \cdot \frac{\partial}{\partial \mathbf{B}_j} \\ &\quad \times \ln p(\mathbf{x}_t \,|\, \mathbf{B}_j(t-1), M_j(t-1)) \\ &= p(C_j \,|\, \mathbf{x}_t) \cdot \left\{ \left[\mathbf{I} - g(\mathbf{y}_j(t)) \right. \right. \\ &\quad \left. \left. \cdot \mathbf{y}_j^T(t) \right] \cdot \mathbf{B}_j(t-1) \right\} \end{aligned} \tag{24}$$

and

$$\mathbf{B}_j(t) = \mathbf{B}_j(t-1) + \frac{\partial}{\partial \mathbf{B}_j} \ln p(\mathbf{x}_t). \tag{25}$$

In (24), the function $g(x)$ is called component-wise nonlinearity function. If the distribution of data is appearing the super-Gaussian distribution, then it will be defined as $g(x) = -2\tanh(x)$. Otherwise, if the distribution of data is appearing the sub-Gaussian distribution, then it will be defined as $g(x) = \tanh(x) - x$.

Since the algorithm of online ICA mixture model can automatically determine the number of clusters according to new incoming data, it solves the problem of conventional ICA mixture model that the number of clusters has to be given beforehand.

B. Structure Learning Algorithm of RFCNN With On-Line ICA Mixture Model

The way the input space is partitioned determines the number of rules extracted from training data as well as the number of fuzzy sets on the universal of discourse of each input variable. We will define a criterion to determine whether a new cluster (i.e., a new fuzzy rule or a new CNN) should be added or not. Let $\mathbf{x}_t$ of cluster j be the newly incoming pattern at time t. Defining

$$F^{J_{\max}}(\mathbf{x}_t) = \max_{1\le j\le J(t)} F^j(\mathbf{x}_t) \tag{26}$$

where $F^j(\mathbf{x}_t) = \ln p(\mathbf{x}_t \mid C_j)$ is the log likelihood value indicating the degree that input data, $\mathbf{x}_t$, belongs to the jth cluster, and the superscript $J_{\max}$ is a maximum log likelihood value among all log likelihood values. If $F^{J_{\max}}(\mathbf{x}_t) \ge F$, the number of cluster is not increased, where F is a pre-specified threshold value that decays during the learning process. In this case, the new incoming pattern is added to an existed cluster and the parameters of this cluster will be updated properly. Oppositely, if $F^{J_{\max}}(\mathbf{x}_t) < F$, the number of cluster will be increased. The threshold value F is determined by experiments.

The whole algorithm for the generation of new fuzzy rules as well as fuzzy sets in each input variable is shown in Fig. 5 step by step. In PART 2 of Fig. 5, the threshold F_{in} determines how many rules will be generated, where F_{in} should be negative since it is taken in natural log. For a lower value of F_{in}, more rules will be generated. Similarly, F_{out} determines how many output clusters will be generated and a lower value of F_{out} will result in more output clusters. For the output space partitioning, the same approach in (14) is used. The generation of a new output cluster corresponds to the generation of a new CNN.

IF $\mathbf{x}_t$ is the first incoming pattern THEN do

PART 1. { Generate a new rule with center $\mathbf{M}_1 = \mathbf{x}_t$,
THEN set the parameter of $\mathbf{Cov}_1 = [\mathbf{0}]$, $p_1(C_1) = 1$, $\mathbf{B}_1 = \mathbf{I}$,
and $k_{1,p}$, where $p = 1, 2, ..., n$.
}

ELSE for each newly incoming $\mathbf{x}_t$, do

PART 2. { Find $J_{\max} = \arg\max_{1\le j\le J(t)} F^j(\mathbf{x}_t)$,
IF $F^{J_{\max}}(\mathbf{x}_t) \ge F_{in}$,
do the following step of on-line ICA mixture model.
ELSE
{ $J(t+1) = J(t) + 1$,
generate a new fuzzy rule with $\mathbf{M}_{J(t+1)} = \mathbf{x}_t$,
and set the parameter of the new cluster.
}
}

Fig. 5. Algorithm of input space partitioning.

Suppose a new input cluster is formed after the presentation of the current input–output training pair $(\mathbf{x}, \mathbf{d})$; then, the consequent part is constructed by the algorithms shown in Fig. 6.

The above algorithm is based on the fact that different precondition of different rules may be mapped to the same consequent term, i.e., CNN. Since only the center of each output membership function is used for defuzzification, the consequent part of each rule may simply be regarded as a singleton. Compared to the general fuzzy rule-based models with singleton output where each rule has its own individual singleton value, fewer parameters are needed in the consequent part of the RFCNN, especially for the case with a large number of rules.

$$\begin{aligned}\mathbf{M}_j(t) &\cong \frac{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)} = \frac{\sum_{i=1}^{t-1} p(C_j \mid \mathbf{x}_i)\mathbf{x}_i + p(C_j \mid \mathbf{x}_t)\mathbf{x}_t}{\sum_{i=1}^{t} p(C_j|\mathbf{x}_i)} \\ &= \frac{1}{tp_t(C_j)}[(t-1)p_{t-1}(C_j)\mathbf{M}_j(t-1) + p(C_j)|\mathbf{x}_t)\mathbf{x}_t] \end{aligned} \tag{19}$$

$$\begin{aligned}\mathbf{Cov}_j(t) &= \frac{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)(\mathbf{x}_i - \mathbf{M}_j(t))(\mathbf{x}_i - \mathbf{M}_j(t))^T}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)} \\ &\cong \frac{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)\mathbf{x}_i\mathbf{x}_i^T}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)} - \mathbf{M}_j(t)\mathbf{M}_j(t)^T \\ &= \frac{\sum_{i=1}^{t-1} p(C_j \mid \mathbf{x}_i)\mathbf{x}_i\mathbf{x}_i^T + p(C_j \mid \mathbf{x}_i)\mathbf{x}_t\mathbf{x}_t^T}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)} - \mathbf{M}_j(t)\mathbf{M}_j(t)^T \\ &= \frac{1}{tp_t(C_j)}\left[(t-1)p_{t-1}(C_j)\mathbf{Cov}_j(t-1) + (t-1)p_{t-1}(C_j)\right. \\ &\quad \left.\times\mathbf{M}_j(t-1)\mathbf{M}_j(t-1)^T + p(C_j \mid \mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T\right] - \mathbf{M}_j(t)\mathbf{M}_j(t)^T \end{aligned} \tag{20}$$

$$\begin{aligned}k_{j,p}(t) = \operatorname{sign}\left\{\right. &\frac{\left[\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)\operatorname{sech}^2(y_{j,p}(i))\right]\left[\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)(y_{j,p}(i))^2\right]}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)} \\ &\left. - \frac{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)y_{j,p}(i)\cdot\tanh^2(y_{j,p}(i))}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)}\right\} \end{aligned} \tag{21}$$

IF there are no output clusters
  do { PART 1 in Fig:5, with $\mathbf{x}$ replaced by $\mathbf{d}$ }
ELSE
  do { Find $J_{\max} = \arg\max_{1 \le j \le J(t)} F^j(\mathbf{x}_t)$,
    IF $F^{J_{\max}}(\mathbf{x}_t) \ge F_{out}$,
      connect input cluster $J(t+1)$ to the existing output
      cluster $J_{\max}$.
    ELSE
      generate a new output cluster,
      connect input cluster $J(t+1)$ to the newly,
      and generated output cluster.
  }

Fig. 6. Algorithm of output space partitioning.

C. Parameter Learning Algorithm of RFCNN by Ordered Derivative Calculus

After the network structure is adjusted according to the current training pattern, the network then enters the parameter identification phase to adjust the parameters of the network optimally based on the same training pattern. Notice that the following parameter learning is performed on the whole network after structure learning; no matter whether the nodes (links) are newly added or are existent originally. Since the RFCNN is a dynamic system with feedback connections, the backpropagation learning algorithm cannot be applied to it directly. Also, due to the online learning property of the RFCNN, the off-line learning algorithms for the recurrent neural networks, like backpropagation through time and time-dependent recurrent backpropagation [17], cannot be applied here. Instead, the ordered derivative [34], which is a partial derivative whose constant and varying terms are defined using an ordered set of equations, is used to derive our learning algorithm. The ordered set of equations, described in Section II in each layer, is summarized in (28)–(33). Our goal is to minimize the error function

$$E(t+1) = \frac{1}{2}\left[y_{\text{out}}(t+1) - y^d_{\text{out}}(t+1)\right]^2 = \frac{1}{2}\varepsilon(t+1)^2 \tag{27}$$

where $y^d_{\text{out}}(t+1)$ is the desired output, $y_{\text{out}}(t+1)$ is the current output, and $\varepsilon(t+1)$ is $(y_{\text{out}}(t+1) - y^d_{\text{out}}(t+1))$. For each training data set, 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 $y_{\text{out}}(t+1)$. In the followings, dependency on time will be omitted unless emphasis on temporal relationships is required.

Summarizing the node functions defined in Section II, the function performed by the network is

$$y_{\text{out}}(t+1) = \sum_i u_i^{(5)} \tag{28}$$

$$u_i^{(5)} = a^{(4)} = a^{(6)} \cdot h^i \tag{29}$$

where

$$h^i = f\left[u_i^{(3)}\right] = \prod_i u_i^{(3)} \tag{30}$$

$$a^{(6)} = \frac{1}{1+e^{-x^i}} \tag{31}$$

$$x^i(t+1) = A^i y_i(t) + B^i u(t) + z^i(t) \tag{32}$$

and (1) is redefined as the following equation for clarity:

$$A^i = [0,0,0;0,a^i,0;0,0,0], \quad B^i_j = [b^i_1, b^i_2, b^i_3; b^i_4, b^i_5, b^i_6; b^i_7, b^i_8, b^i_9]. \tag{33}$$

With the above formula and the error function defined in (27), we can derive the update rules for the free parameters in the RFCNN as follows.

Update rule of a^i (the parameter of feedback template of the ith CNN) is

$$a^i(t+1) = a^i(t) - \eta\frac{\partial^+ E(t+1)}{\partial a^i} \tag{34}$$

$$\begin{aligned}\frac{\partial^+ E(t+1)}{\partial a^i} &= \frac{\partial E(t+1)}{\partial a^i} + \sum_k \frac{\partial E(t+1)}{\partial y_{\text{out},k}(t+1)}\frac{\partial^+ y_{\text{out},k}(t+1)}{\partial a^i} \\ &= \frac{\partial E(t+1)}{\partial y_{\text{out}}(t+1)}\frac{\partial^+ y_{\text{out}}(t+1)}{\partial a^i}\end{aligned} \tag{35}$$

where

$$\frac{\partial E(t+1)}{\partial y_{\text{out}}(t+1)} = \varepsilon(t+1) \tag{36}$$

and

$$\begin{aligned}\frac{\partial^+ y_{\text{out}}(t+1)}{\partial a^i} &= \sum_k \frac{\partial y_{\text{out}}(t+1)}{\partial a_k^{(4)}}\frac{\partial^+ a_k^{(4)}}{\partial a^i} \\ &= \frac{\partial y_{\text{out}}(t+1)}{\partial a_i^{(4)}}\frac{\partial^+ a_i^{(4)}}{\partial a^i}\end{aligned} \tag{37}$$

where

$$\frac{\partial y_{\text{out}}(t+1)}{\partial a_i^{(4)}} = \frac{\partial}{\partial a_i^{(4)}}\sum_k a_k^{(4)}(t+1) = 1 \tag{38}$$

and

$$\begin{aligned}\frac{\partial^+ a_i^{(4)}}{\partial a^i} &= \frac{\partial}{\partial a^i}\left[a_i^{(6)}(A^i y_i(t) + B^i u(t) + z^i(t))\right] h^i \\ &= h^i a_i^{(6)}\left(1 - a_i^{(6)}\right)\left[y_i(t) + a^i\frac{\partial y_i(t)}{\partial a^i}\right].\end{aligned} \tag{39}$$

Hence, the parameter a^i is updated by

$$a^i(t+1) = a^i(t) - \eta\varepsilon(t+1)\left\{h^i a_i^{(6)}\left(1 - a_i^{(6)}\right) \times \left[y_i(t) + a^i\frac{\partial y_i(t)}{\partial a^i}\right]\right\}. \tag{40}$$

Similarly, the parameter b^i_j (the parameters of control template of the ith CNN) is updated by

$$b^i_j(t+1) = b^i_j(t) - \eta\varepsilon(t+1)\left[h^i a^{(6)}\left(1 - a^{(6)}\right)x_j(t)\right] \tag{41}$$

and the parameter z_i (the bias of the ith CNN) is updated by

$$z^i(t+1) = z^i(t) - \eta\varepsilon(t+1)\left[h^i a^{(6)}\left(1 - a^{(6)}\right)\right]. \tag{42}$$

As shown in (37) to (39), the update rules are in recursive form. The value $(\partial^+ y/\partial a)$ is equal to zero initially. For the rest free parameters in the RFCNN, they are obtained during the structure-learning phase by the online ICA mixture model algorithm proposed in the last section. Notice that according to the real-time recurrent learning (RTRL) scheme [35], we can also obtain the same parameter learning rules for the RFCNN.

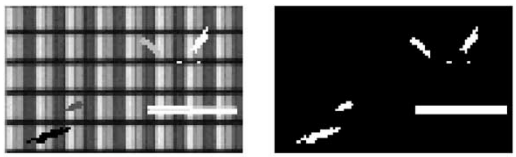

Fig. 7. Training images. (a) Input image. (b) Desired output.

Of course, other existing online learning algorithms [36], [37] for tuning the weights of recurrent neural networks can be possibly adopted for tuning the RFCNN, too.

IV. Experimental Results and Discussions

The capability of the proposed RFCNN is demonstrated on the real-world defect inspection problems. Automatic defect inspection systems are becoming more and more important in industrial production lines. Especially in the electronics industry, an attempt is often made to achieve almost 100% quality control of all components and final goods. Here, we are interested in the defect inspection of color filter, which is one of components in thin film transistor liquid crystal display (TFT-LCD) module and gives each pixel of LCD its own color. The difficulties in the defect inspection of color filter are its complex texture and need for high-speed processing. For high-speed processing, the CNN is a good way to achieve defect inspection. Besides, different kinds of defects in color filter need different CNN templates and some complex defects cannot be detected by a single CNN. Therefore, the proposed RFCNN is a good alternative to detect defect of color filter images. To train the RFCNN, we use a 3×3 window to get the system inputs and set the whole image as the inputs of the RFCNN. The 3×3 window covers the central pixel and its eight connected neighbors. The training image and corresponding desired output are shown in Fig. 7(a) and (b). We set the threshold $F_{\text{in}} = F_{\text{out}} = -50$ and learning rate as $\eta = 0.001$ for the clustering algorithm. As mentioned in Section III, there are no rules (and no CNNs) in the RFCNN initially. They are created dynamically as learning proceeds upon receiving online incoming training data by performing the learning processes shown in Fig. 3. When the learning processes are done, three clusters (three fuzzy rules and CNN templates) were obtained. For an example of color filter, it takes about 1 min to learn the structure (interconnection set) and 2 minutes to learn the parameters with a Pentium IV 2.0-GHz PC. However, the training can be done off-line, so it is not a problem for the online processing of CNN, which causes just little time.

Fig. 8 shows the outputs of Layer 3, 4, and feedback layer for the training image. Fig. 8(a) to (c) shows the outputs of the three Layer-4 nodes, respectively, i.e., the outputs of the three CNNs in the feedback layer multiplied by the outputs of the three Layer-3 nodes (i.e., firing strength of each rule), respectively. Fig. 8(d) to (f) shows the outputs of the three CNNs in the feedback layer, respectively. Fig. 8(g) to (i) shows the outputs of the three Layer-3 nodes, respectively, (firing strength of each rule). The sum of the outputs of the three Layer-4 nodes [i.e., Fig. 8(a) to (c)] forms the RFCNN final output. From Fig. 8(a) to (c), we can see that CNN 1 takes care of the defect texture in the right side of the training image, and CNNs 2 and 3 mainly

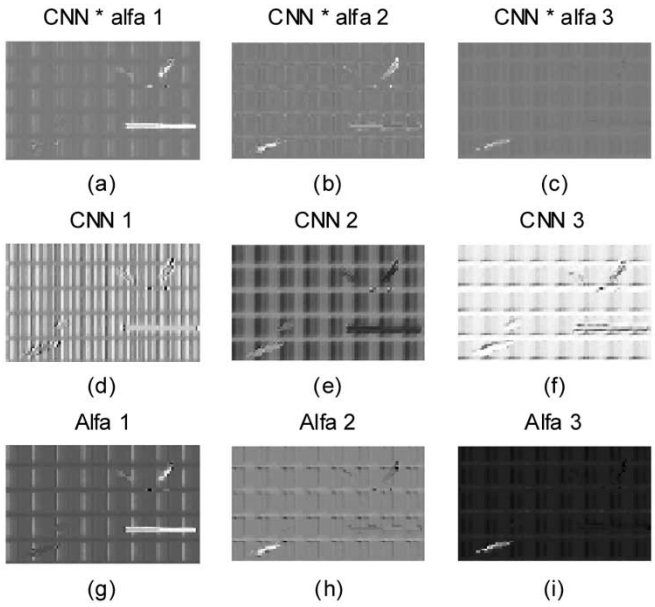

Fig. 8. Outputs of Layer 3, 4, and feedback layer for the training image. (a)–(c) Outputs of the three Layer-4 nodes, respectively. (d)–(f) Outputs of the three CNNs in the Feedback Layer, respectively. (g)–(i) Outputs of the three Layer-3 nodes, respectively (firing strength of each rule).

take care of the defect textures in the left side of the training image. The template of each learned CNN is given as follows:

$$A^1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -0.64 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B^1 = \begin{bmatrix} 0.27 & -0.20 & 0.12 \\ 1.58 & -2.45 & 1.29 \\ 0.29 & -0.58 & 0.47 \end{bmatrix}, \qquad z^1 = -0.02.$$

$$A^2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -0.83 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B^2 = \begin{bmatrix} 0 & -0.17 & -0.68 \\ 0.20 & -0.65 & 0.40 \\ -0.11 & 0.08 & -0.15 \end{bmatrix}, \qquad z^2 = 0.37.$$

$$A^3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.53 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B^3 = \begin{bmatrix} 0.09 & 1.26 & -1.22 \\ -1.58 & -1.70 & -0.57 \\ 0.59 & 0.50 & 1.54 \end{bmatrix}, \qquad z^3 = 1.78.$$

Based on the learned structure and parameters of the RFCNN, we test three images as shown in Fig. 9. Fig. 9(a), (c), and 9(e) shows the testing images and Fig. 9(b), (d), and 9(f) shows the corresponding results of defect inspection. From Fig. 9(a) to (f), we can see that the learned structure and CNN templates of the RFCNN are well suited to detect the defects of color filer images. It has also been tested that detection results are still good if the images are shifted, that is because that the RFCNN only considers the central pixel and its eight connected neighbors and they are still regular patterns after images are shifted. Therefore, if the images are shifted, we need not reteach the network.

The conventional methods using CNN for defect inspection [38]–[41] are using one or a set of CNN templates, which can

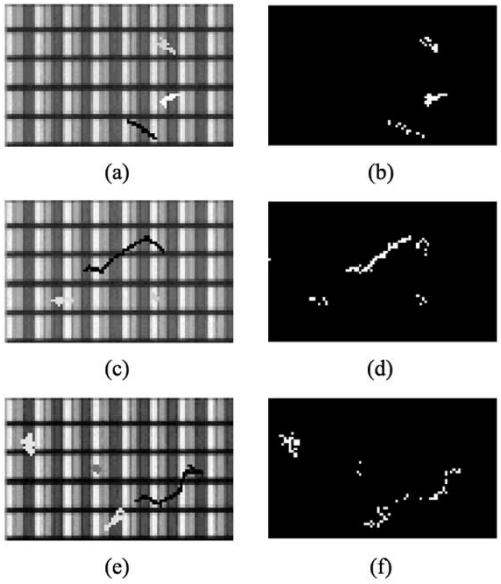

Fig. 9. Experimental (Testing) results of the learned RFCNN. (a), (c), and (e) are input testing images. (b), (d), and (f) are corresponding detection results.

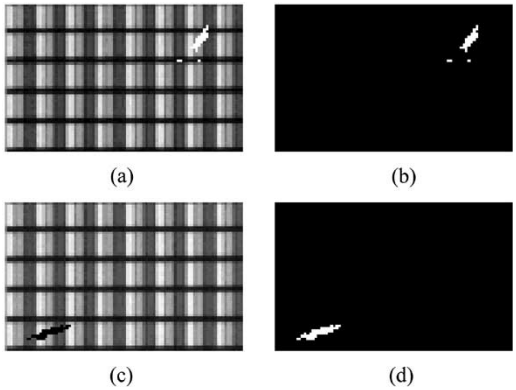

Fig. 10. Training images by GA. (a) and (c) are input images. (b) and (d) are corresponding desired outputs.

be obtained by experiential engineers or learned by examples, to detect defect. To compare the RFCNN with conventional methods, we performed some experiments using a single CNN whose template is learned by the genetic algorithm (GA). We find that the training image, shown in Fig. 7(a), cannot be learned well by using only a single CNN. However, if we have the training images and corresponding desired outputs as shown in Fig. 10(a) to (d), the CNN template can be learned well by GA. This fact implies that different kinds of defects in color filter need different CNN templates. That is, we can first identify the categories of defects and make each CNN template of defect category learned by GA. However, this will cause related questions as follows. First, how many defect categories, which determine how many CNN templates, should be classified? Second, how can we be sure which defects belong to the same category? In other words, what is the corresponding desired output for the uncategorized defects of color filter? Therefore it is difficult to manually use the divide-and-conquer principle to learn the templates of CNNs by GA. For the dilemma mentioned above, the proposed RFCNN provides a good alternative to solve this kind of problem.

To make the RFCNN converge more quickly during learning, GA can be used to learn some CNN templates to initialize the consequent part of the RFCNN. Though this experiment focuses on defect inspection of color filter, the proposed RFCNN can be also applied to those images with regular pattern, such as texture webs.

The main idea of the proposed RFCNN is an integrated system of FIS and CNNs, which can construct fuzzy rules and CNN templates automatically. The example for the defect inspection of color filter has been demonstrated to verify the capability of the RFCNN. In addition to the defect inspection of color filter, we believe such an integrated CNN system, the RFCNN, has potential to solve more complex intelligent problems such as biological phenomena or other applications. Since CNN bears the characteristic of high-speed processing based on analog circuit realization, it will be very useful to realize the RFCNN by analog circuits. As studied in [7], [11], the elementary fuzzy-logic computations, such as the $\min$, $\max$, and fuzzification operator in a fixed neighborhood, have already been designed in CNN. Therefore, it is very promising and feasible to implement the RFCNN in the future work. An implementation scheme to realize the RFCNN includes the following two steps. First, use the RFCNN to learn the fuzzy rules and CNN templates. Second, construct a FIS based on the learned fuzzy rules and CNN templates.

For taking into account the nonidealities or mismatch due to the manufacturing, there are some ways can be done as follows:

1) We may add constraints with upper bound and lower bound to the learned parameter in learning algorithm.
2) The interval parameter learning is also available in [42] such that a tolerant range of parameters (weights) deviation can be achieved.
3) Since the proposed RFCNN can learn the structure and parameters automatically, we can increase the number of CNN and other nodes automatically to achieve the required accuracy if the target accuracy has not been satisfied.

V. Conclusion

In this paper, we propose a novel framework, called RFCNN, for automatically constructing a multiple-CNN integrated neural system. This CNN-based FNN can automatically learn its proper network structure and parameters simultaneously. The structure learning includes the creation of fuzzy rules and CNNs with a new online adaptive ICA mixture-model technique. The parameter learning includes the tuning of fuzzy membership functions and CNN templates based on the ordered derivative calculus. The proposed RFCNN provides a solution to the current dilemma on the decision of templates and/or fuzzy rules in the existing integrated (fuzzy) CNN systems. In order to verify the capability of the RFCNN, a real-world defect inspection problem has been demonstrated. The experimental results show that the proposed scheme is effective and promising. Our future work includes extending the RFCNN to include the coupled CNNs and finding more application examples.

REFERENCES

[1] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257–1272, Oct. 1988.

[2] ——, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1273–1290, Oct. 1988.

[3] J. Hámori and T. Roska, *Receptive Field Atlas of the Retinotopic Visual Pathway and Some Other Sensory Organs Using Dynamic Cellular Network Models*, Budapest, Hungary: Analogical and Neural Computing Laboratory, MTA-SZTAKI, DNS-8-2000.

[4] I. Szatmári, D. Bálya, G. Tımár, C. Rekeczky, and T. Roska, "Multichannel spatio-temporal topographic processing for visual search and navigation," in *Proc. SPIE Microtechnologies for the New Millenium*, Gran Canaria, Spain, May 2003, Paper 5119-38.

[5] T. Yang, L. B. Yang, C. W. Wu, and L. O. Chua, "Fuzzy cellular neural networks: Applications," in *Proc. Cellular Neural Networks Application (CNNA'96)*, pp. 225–230.

[6] T. Yang and L. B. Yang, "Fuzzy cellular neural network: A new paradigm for image processing," *Int. J. Circuit Theory Applicat.*, vol. 25, no. 6, pp. 469–481, 1997.

[7] M. Balsi and F. Voci, "Implementation of fuzzy rule based image processing on the CNN universal machine," in *Proc. Eur. Conf. Circuit Theory and Design (ECCTD'99)*, 1999, pp. 1167–1170.

[8] ——, "Fuzzy reasoning for the design of CNN-based image processing systems," in *Proc. IEEE Symp. Circuits and System (ISCAS'00)*, Geneva, Switzerland, May 28–31, 2000, pp. 405–408.

[9] F. Colodro and A. Torralba, "Cellular neuro-fuzzy networks (CNFNs), a new class of cellular networks," in *Proc. 5th IEEE Int. Conf. Fuzzy Systems*, vol. 1, Sept. 8–11, 1996, pp. 517–521.

[10] Cs. Rekeczky, T. Roska, and A. Ushida, "CNN-based difference-controlled adaptive nonlinear image filters," *Int. J. Circuit Theory Applicat.*, vol. 26, pp. 375–423, 1998.

[11] Cs. Rekeczky, Á. Tahy, Z. Végh, and T. Roska, "CNN based spatio-temporal nonlinear filtering and endocardial boundary detection in echocardiography," *Int. J. Circuit Theory Applicat.*, vol. 27, pp. 171–207, 1999.

[12] J. A. Nossek, "Design and learning with cellular neural networks," *Int. J. Circuit Theory Applicat.*, no. 24, pp. 15–24, 1996.

[13] A. Zarandy, "The art of CNN template design," *Int. J. Circuit Theory Applicat.*, no. 27, pp. 5–23, 1999.

[14] T. Kozek, T. Roska, and L. O. Chua, "Genetic algorithm for CNN template learning," *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 392–402, June 1993.

[15] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[16] C. T. Lin, *Neural Fuzzy Control Systems With Structure and Parameter Learning*. New York: World Scientific, 1994.

[17] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hell, 1996.

[18] R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[19] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro-Fuzzy Systems*. New York: Wiley, 1997.

[20] L. Wang and R. Langari, "Building Sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 454–458, Nov. 1995.

[21] E. H. Ruspini, "Recent development in fuzzy clustering," *Fuzzy Set and Possibility Theory*, pp. 113–147, 1982.

[22] C. T. Sun and J. S. Jang, "A neuro-fuzzy classifier and its applications," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, vol. 1, San Francisco, CA, Mar. 1993, pp. 94–98.

[23] C. F. Juang and C. T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 12–32, Feb. 1998.

[24] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: Wiley, 2001.

[25] C. Jutten and J. Herault *et al.*, "Independent components analysis (ICA) versus principal components analysis," in *Signal Processing IV: Theories and Applications, EUSIPCO-88*, J. Lacoume *et al.*, Eds, Amsterdam, The Netherlands: Elsevier, 1988, pp. 643–646.

[26] J. C. Bezdek, J. Keller, R. Krisnapuram, and N. R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Boston, MA: Kluwer, 1999.

[27] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[28] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy Cluster Analysis*. New York: Wiley, 1999.

[29] G. J. McLachlan and T. Krishnan, *The EM Algorithms and Extensions*. New York: Wiley, 1997.

[30] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[31] L. X. Wang, *Adaptive Fuzzy Systems and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[32] T. W. Lee, M. S. Lewicki, and T. J. Sejnowski, "ICA mixture models for unsupervised classification of non-Gaussian classes and automatic context switching in blind signal separation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 1078–1089, Oct. 2000.

[33] ——, "ICA mixture models for unsupervised and automatic context switching," in *Proc. Int. Workshop Independent Component Analysis*, 1999, pp. 209–214.

[34] P. Werbos, "Beyond regression: New Tools for prediction and analysis in the behavior sciences," Ph.D. dissertation, Dep. Appl. Math., Harvard Univ., Cambridge, MA, Aug. 1974.

[35] R. J. Williams and D. Zipser, "A learning algorithm for continually running recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.

[36] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Trans. Neural Networks*, vol. 6, pp. 1212–1228, Sept. 1995.

[37] S. W. Piché, "Steepest descent algorithms for neural-network controllers and filters," *IEEE Trans. Neural Networks*, vol. 5, pp. 198–212, Mar. 1994.

[38] V. Preciado, D. Guinea, R. Montufar, and J. Vicente, "Real-time inspection of metal laminates by means of CNNs," *Proc. SPIE*, vol. 4301, no. 39, pp. 260–270, 2001.

[39] D. Guinea, A. Gordaliza, J. Vicente, and M. C. Garcıa-Alegre, "CNN based visual processing for industrial inspection," *Proc. SPIE*, vol. 3966, no. 45, pp. 315–322, 2000.

[40] C. L. Chang and C. T. Lin, "CNN-based defect inspection in images with regular pattern," in *Proc. 16th Eur. Conf. Circuit Theory and Design (ECCTD'03)*, 2003, pp. I221–I224.

[41] R. Perfetti and L. Terzoli, "Analogic CNN algorithms for textile applications," *Int. J. Circuit Theory Applicat.*, no. 28, pp. 77–85, 2000.

[42] C. T. Lin and Y. C. Lu, "A neural fuzzy system with fuzzy supervised learning," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 744–763, May 1996.

Chin-Teng Lin (S'88–M'91–SM'99) received the B.S. degree in control engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, R.O.C., and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, Lafayette, IN, in 1986, 1989, and 1992, respectively.

Since August 1992, he has been with the College of Electrical Engineering and Computer Science, NCTU, where he is currently the Associate Dean of the college and a Professor in the Electrical and Control Engineering Department. He has also served as the Director of Brain Research Center, NCTU Branch, University System of Taiwan, since September 2003. He served as the Director of the Research and Development Office, NCTU, from 1998 to 2000, and the Chairman of Electrical and Control Engineering Department from 2000 to 2003. His current research interests are neural networks, fuzzy systems, cellular neural networks, FNNs, neural engineering, algorithms and very-large-scale integration design for pattern recognition, intelligent control, and multimedia (including image/video and speech/audio) signal processing, and intelligent transportation system. He is the coauthor of the book *Neural Fuzzy Systems- A Neuro-Fuzzy Synergism to Intelligent Systems* (Englewood Cliffs, NJ: Prentice-Hall 1996), and the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (Singapore, World Scientific). He has published over 75 journal papers in the areas of neural networks, fuzzy systems, multimedia hardware/software, and soft computing, including 56 IEEE journal papers.

Dr. Lin has won the Outstanding Research Award granted by National Science Council, Taiwan, since 1997 to the present; the Outstanding Electrical Engineering Professor Award granted by the Chinese Institute of Electrical Engineering, in 1997; the Outstanding Engineering Professor Award granted by the Chinese Institute of Engineering, in 2000; and the 2002 Taiwan Outstanding Information-Technology Expert Award. He was elected as one of 38th Ten Outstanding Rising Stars in Taiwan, R.O.C., in 2000. He currently serves as the Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, IEEE TRANSACTIONS ON SYSTEMS, MAN, CYBERNETICS—B, and IEEE TRANSACTIONS ON FUZZY SYSTEMS. He is also a member of the IEEE Circuit and Systems Society (CASS), the IEEE Neural Network Society, the IEEE Computer Society, the IEEE Robotics and Automation Society, and the IEEE System, Man, Cybernetics Society. He is the Distinguished Lecturer representing the NSATC of IEEE CASS from 2003 to 2005. He has been the Council member of International Fuzzy System Association, since 2000, the member of the Board of Government, of Asia Pacific Neural Network Assembly, since 2000, and the Executive Council member (Supervisor) of the Chinese Automation Association, since 1998. He is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi.

Chun-Lung Chang received the B.S. degree in automatic control engineering from the Feng-Chia University, Taichung, Taiwan, R.O.C., and the M.S. degree in power mechanical engineering from the National Tsing-Hua University, Hsinchu, Taiwan, R.O.C., in 1990 and 1992, respectively. He is currently working toward the Ph.D. degree in electrical and control engineering at National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

He is also a Researcher in Mechanical Industry Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan, R.O.C. His current research interests are neural networks, fuzzy control, image processing, and computer vision.

Wen-Chang Cheng received the B.S. degree in electronics engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C., the M.S. degree in electronics engineering from National Chung Cheng University, Chiayi, Taiwan, R.O.C., in 1997 and 1999, respectively. He is currently working toward the Ph.D. degree in electrical and control engineering at National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

His current research interests include neuro-fuzzy systems, neural networks, image processing, machine learning, and artificial intelligence.