Synergistic Modeling and Applications of Hierarchical Fuzzy Neural Networks

SUN-YUAN KUNG, FELLOW, IEEE, JINSHIUH TAUR, MEMBER, IEEE, AND SHANG-HUNG LIN, MEMBER, IEEE

Invited Paper

Many common foundations exist between neural networks and fuzzy inference systems in terms of their mathematical models and system structures. This paper explores such a rich synergy and uses it to form the basis for a unifying framework under which fuzzy logic processing and neural networks may be integrated to achieve more robust information processing. It in turn leads to a family of hierarchical fuzzy neural networks (FNN's) which incorporate an adaptive and modular design of neural networks into the basic fuzzy logic systems. Several important models which are critical to the development of the hierarchical FNN family are studied carefully so as to gain a better understanding of how the FNN's could benefit from the symbiotic marriage of the learning techniques of neural networks and the inference structure of fuzzy logic systems. Specifically, we demonstrate how existing unsupervised [e.g., competition-based learning, expectation-maximization (EM) algorithm] and supervised (e.g., reinforced/antireinforced learning) training strategies can be an integral part of a fuzzy processing framework. In addition, for robust processing, hierarchical structures involving both expert (rule) modules and class modules are incorporated into the FNN's. Also presented are some promising application examples for biometric authentication, medical image processing, video segmentation, object recognition/detection, and multimedia content-based retrieval.

Keywords—*Fuzzy inference system, fuzzy neural networks, hierarchical neural networks, supervised learning, unsupervised learning.*

I. Introduction

The rapid advances in parallel processing and multimedia processing technologies have revolutionized the conception of information processing. This calls for intelligent systems which are capable of managing complex, uncertain, sometimes even contradictory, information sources. This paper explores such a potential framework for a robust information processing system.

Traditional neural networks (NN's) can be treated as black boxes whose parameters are obtained by training from example data sets. The parameters in the multilayered perceptron NN's have no physical meanings and the initial values are typically selected at random. It is in general impossible to explain or pinpoint the meaning of the network parameters. In contrast, the parameters in fuzzy logic systems have physical meaning. Consequently, it is easier to design the initial values from the input–output data pairs and to incorporate a human expert's knowledge into fuzzy systems. Unlike the NN approach, fuzzy systems only require that we partially complete a linguistic rule base [35]. Although the task is simpler than designing and training an NN, fuzzy systems must face the difficulty of choosing proper fuzzy logic rules and membership functions.

Recently, many researchers have considered modular NN's from a competitive mixture perspective [20], [22], [23], [47]. This approach solves a large complicated task by using smaller and modularized trainable networks (i.e., experts), whose solutions are dynamically integrated into a single coherent one using the trainable gating network. More precisely, the gating network is implemented as a softmax activation function [3], [43]. A local expert's output with a larger (smaller) gating activation will be allocated a greater (lesser) influence on the overall output. In a very similar fashion, fuzzy inference systems (FIS's) perform task decomposition based on fuzzy membership functions. The Sugeno-type FIS learning can be viewed as a variation of the modular network where each local expert is expressed as a rule model. The integrating unit is a fuzzy inference module, which plays the same role as a gating network.

This paper is based on the premise that the differences between neural and fuzzy systems are really not very fundamental in nature. On the contrary, the rich common characteristics shared by fuzzy systems and NN's war-

Manuscript received March 23, 1998; revised March 9, 1999. This work was supported in part by Mitsubishi Electric, ITA, and the R.O.C. National Science Council through Grant NSC 88-2213-E-005-012.

S.-Y. Kung is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA.

J. Taur is with the Department of Electrical Engineering, National Chung-Hsing University, Taichung, Taiwan, R.O.C.

S.-H. Lin is with EPSON Palo Alto Laboratory, ERD, Palo Alto, CA 94304 USA.

Publisher Item Identifier S 0018-9219(99)06912-1.

ıeir integration. An integrated system can enjoy the tages of both NN's (e.g., learning and optimization ilities) and fuzzy systems (e.g., human-like If–Then rules and ease of incorporating expert knowledge).

; paper is organized as follows. In Section II, we be a family of fuzzy neural networks (FNN's) which from the intersection of the two intelligent infor-n processing systems. Specifically, such a baseline structure can be modeled after either a Sugeno-type r a mixture-of-experts (MOE) modular network. The ıtage of incorporating class-level modules into the s can be demonstrated using a decision-based neural ɔrk (DBNN) model. This section helps demonstrate the FNN's can benefit from the marriage of the ng techniques of NN's and inference structure of logic systems. A unified mathematical and struc-foundation for this family of networks is established. ection III, we describe several important models in levelopment of the FNN family. The unsupervised ering expectation-maximization (EM) will serve as al core technology for fuzzy clustering. Combining the ural properties of MOE and DBNN, several multilevel rchical FNN's can be developed. Finally, we present a neural classifier (NEFCAR) to show how to improve m performance by integrating neural learning tech-es and the fuzzy linguistic structure with both positive negative rules. The class-level hierarchical structure fuzzy If–Then logic rules are integrated into this ifier. In Section IV, several applications in image/video essing and pattern recognition for the proposed FNN's ıighlighted. The proposed family of fuzzy NN's are ıtially very appealing to a broader application spec-, including system control and identification as well as rn recognition.

FIS'S AND MODULAR NN'S

ıis section reviews the basic NN's and fuzzy logic ms. Generally speaking, they have different perspec-toward the design of intelligent systems. Nevertheless, ral prominent fuzzy systems can be easily reformu- as NN's and vice versa. The mathematical models system structures for modular NN's, e.g., mixture of rts, and (Sugeno-type) fuzzy inference systems bear eat resemblance. The commonality of their structural eworks and functional units will be studied. Based he study, a unified mathematical and architectural dation for FNN's will be established.

IN's

n NN can be defined as an architecture comprising sively parallel adaptive processing elements intercon-ed via hierarchically structured networks. NN's are acterized by the following features:

ı adaptiveness and self-organization which offers ro-bust processing capabilities;

ı nonlinear processing which enhances the network's approximation, classification, and noise-immunity ca-pabilities;

3) parallel processing which usually employs a large number of processing cells enhanced by extensive interconnectivity.

Under such a broad definition, many systems can be expressed as NN's with different interconnect structures and nonlinear processing elements. There are many design options for the cost function, the learning algorithm, the nonlinear function, and the network structure. Ultimately, the design of the artificial NN's often depends on the chosen applications.

One critical decision involves the selection of the basis functions. There are several alternatives for the basis functions of the basic neural units.

1) The linear basis function (LBF) [52] employs a sigmoid-type threshold function over the linear vector product of the pattern vector and weight vectors. Such a threshold function serves as the basic discriminating unit.
2) The radial basis function (RBF) [44], [50] employs a RBF (e.g., a Gaussian kernel) to serve as the activation function. The weighting parameters in the RBF network are the centers, the widths, and the heights of these kernels.

This paper adopts primarily the RBF-based Gaussian function due to its popularity as well as mathematical succinctness. Nevertheless, the same hierarchical structure and the learning mechanism remain largely applicable to other types of basis or membership functions.

1) Hierarchical Levels: The traditional NN's, exemplified by the multilayer perceptron (MLP), have a connectionist structure. They offer some measure of fault tolerance and distributed representation properties. More importantly, they possess adaptive learning abilities to estimate sampled functions, represent these samples, encode structural knowledge, and infer input–output relationships via association.

The MLP NN can have a large number of interconnected nodes to enhance the ability to learn and generalize from training data. Its main strength lies in that, given sufficient hidden units, an MLP is a universal approximator: an MLP can approximate any continuous function on a compact subset to any desired accuracy [8]. On the other hand, as depicted in Fig. 1, the main weakness of the MLP lies in its totally flat structure. A direct consequence of such structural simplicity is often a bulky network, with an excessively large number of hidden units, which in turn hampers the convergence of the training process. One effective solution is to incorporate proper hierarchical structure into the networks. The notions of modular networks and functional blocks are essential for hierarchical designs. In particular, the following two types of modules will be relevant to the FNN's.

1) *Expert-Level (Rule-Level) Modules:* The effectiveness of designing a modular NN hinges upon a proper designation of local experts. Each expert serves the function of: 1) extracting local features and 2) making local recommendations. For example, one (LBF or RBF) hidden node may be devoted to extract a

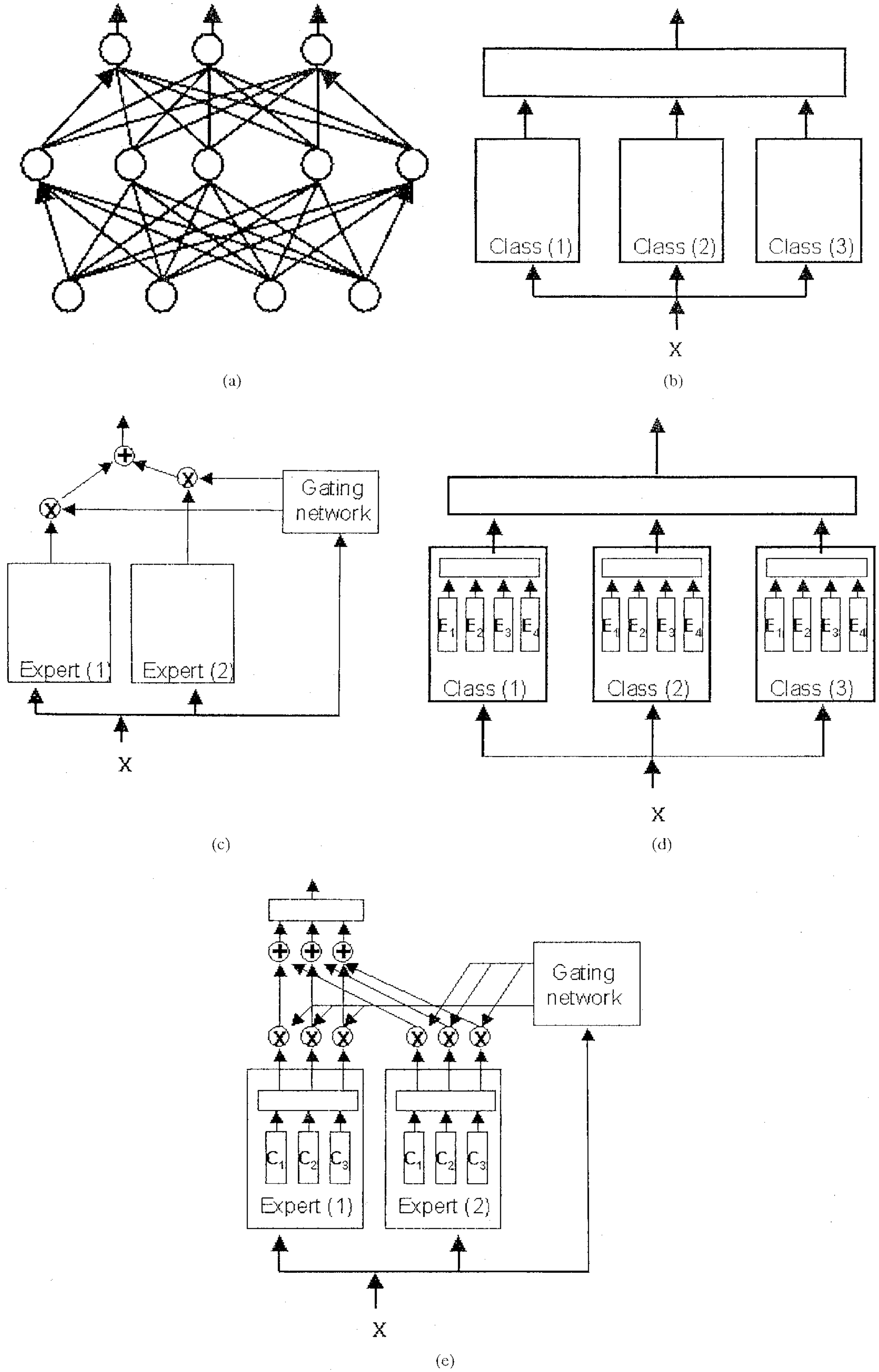


Fig. 1. Several possible hierarchies in NN structures: (a) multilayer perceptrons; (b) DBNN; (c) mixture of experts network; (d) experts-in-class network; and (e) classes-in-expert network.

certain local feature of particular interest to an expert. The expert level in NN's is compatible with the rule level in the fuzzy systems. The rules in the gating network are used to decide how to combine recommendations from several local experts, with corresponding degrees of confidence.

2) *Class-Level Modules:* An important goal of pattern recognition is to determine to which class an input pattern best belongs. Therefore, it is natural to consider class-level modules as the basic partitioning units, where each module specializes in distinguishing its own class from the others. Consequently, the number of hidden nodes (or experts) designated to a particular class is often very small. The class-level modules are adopted by the one-class-one-net (OCON) network. In contrast to expert-level partitioning, this OCON structure facilitates a global (or mutual) supervised training scheme. In global interclass supervised learning, any dispute over a pattern region by (two or more) competing classes may be effectively resolved by resorting to the teacher's guidance. Such a distributed processing structure is also convenient for network upgrading when there is a need for adding or removing memberships. Finally, such a distributed structure is also appealing to the design of the RBF networks.

:cordingly, NN's can be divided structurally into the lowing categories (Fig. 1):

1) *Zero-Level (i.e., Structurally Flat) Networks:* This is exemplified by the traditional MLP which has a "single-network" structure shown in Fig. 1(a).
2) *One-Level Modular Structures:* By adopting the divide-and-conquer principle, the task is first divided into modules and then the individual results are integrated into a final and collective decision. Two typical modular networks are: 1) MOE which utilizes the expert-level modules [cf. Fig. 1(c)] and 2) the DBNN based on the class-level modules [cf. Fig. 1(b)]. See Sections II-A2 and II-A3, respectively.
3) *Two-Level Hierarchical Structures:* To this end, the divide-and-conquer principle needs to be applied twice: one time on the expert-level and another on the class-level. Depending on the order used, this could result in two kinds of hierarchical networks: one has an experts-in-class construct and another a classes-in-expert construct, as depicted in Fig. 1(d) and (e). More on these two hierarchical structures will be elaborated in Section III-B.

?) *MOE:* Among the prominent neural models, the net-rk architecture of the MOE [23] has the closest re-nblance to FIS's. The MOE exhibits an explicit re-ionship with statistical pattern classification methods. ven a pattern, each expert network estimates the pattern's iditional *a posteriori* probability on the (adaptively tuned preassigned) feature space. Each local expert network forms multiway classification over K classes by using ier K independent binomial models, each modeling only : class, or one multinomial model for all classes.

Moreover, the corresponding output of the gating network resents the associated confidence on each expert. The al system output is the weighted sum of the estimated babilities from all of the expert networks.

With reference to Fig. 2, the MOE comprises the follow- subsystems.

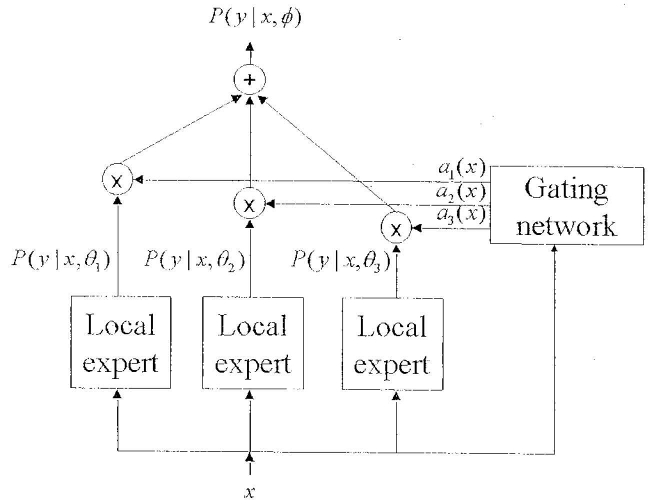


Fig. 2. The baseline MOE architecture. An expert network estimates the pattern's conditional *a posteriori* probability. A baseline MOE comprises two subsystems: local experts and a gating network. The local experts are adaptively trained to extract certain local features particularly relevant to their own local decisions, while the gating network computes the global weights to be applied the local decisions.

1) *Local Experts:* The design of modular NN's hinges upon the choice of local experts.
   a) Usually, a local expert is adaptively trained to extract a certain local feature that is particularly relevant to its local decision. Sometimes, a local expert can be assigned a predetermined feature space.
   b) Based on the local feature, a local expert will process a local recommendation.
2) *Gating Network:* The gating network serves the function of computing the proper weights to be used for the final weighted decision. A probabilistic rule is used to integrate recommendations from several local experts taking into account the experts' confidence levels.

The training of the local experts as well as the confidence levels in the gating network of the MOE network is based on the EM algorithm. The objective is to estimate the model parameters so as to attain the highest probability of the training set given the estimated parameters. For a given input x, the posterior probability of generating class y given x using K experts can be computed as

$$P(\mathbf{y}|\mathbf{x}, \phi) = \sum_{j=1}^{K} P(\mathbf{y}|\mathbf{x}, \Theta_j)a_j(\mathbf{x}) \tag{1}$$

where $\mathbf{y}$ is a binary vector and a_j is the probability for weighting the expert outputs. For example, if we consider two classes for a classification problem, then $\mathbf{y}$ is [1 0] or [0 1]. The parameter ϕ is a vector $[\mathbf{V}, \Theta_j]$; $\mathbf{V} = \{\mathbf{V}_j, j = 1, \ldots, K\}$ is the parameter set for the gating network; Θ_j is the parameter set for the jth expert network $(j = 1, \ldots, K)$; and $P(\mathbf{y}|\mathbf{x}, \Theta_j)$ is the output of the jth expert network. The gating network of MOE can be a linear

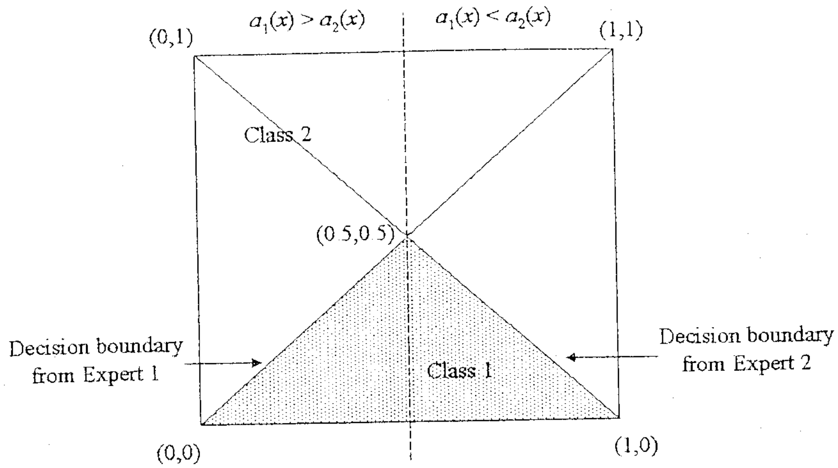


Fig. 3. An example for MOE and FIS. For the MOE, a local expert will get a higher weighting on its proclaimed feature space. In this example, the gating network will assign higher confidence to expert 1 for the region in $\{(x_1, x_2), 0 \le x_1 \le 0.5, 0 \le x_2 \le 1\}$, and to expert 2 for the remaining region. For the FIS, similar confidence weighting is applied.

LBF network (say, linear perceptron) or a nonlinear RBF network.

a) LBF MOE example: An example of the linear gating network is shown in the following [3], [43]:

$$a_j(\mathbf{x}) = \exp[b_j(\mathbf{x})] \Big/ \sum_{k=1}^{K} \exp[b_k(\mathbf{x})] \tag{2}$$

where $b_j(\mathbf{x}) = \mathbf{v}_j^T \mathbf{x} + d_j$. $\mathbf{V}_j = \{\mathbf{v}_j, d_j\}$ denotes the weights of the jth neuron of the gating network when the gating network is a linear perceptron.

Suppose we have patterns from two classes occupying a two-dimensional (2-D) space $\{\mathbf{x} = (x_1, x_2), 0 \le x_1 \le 1, 0 \le x_2 \le 1\}$. Class 1 patterns occupy the lower quarter of the square in Fig. 3, while Class 2 patterns take the remaining three quarters of the space. We adopt a two-expert MOE classifier with the LBF local experts

$$P(\omega_1|\mathbf{x}, \Theta_1) = \frac{1}{1 + e^{\mathbf{w}_1^T \mathbf{x} + c_1}}$$
$$P(\omega_2|\mathbf{x}, \Theta_1) = 1 - P(\omega_1|\mathbf{x}, \Theta_1)$$
$$P(\omega_1|\mathbf{x}, \Theta_2) = \frac{1}{1 + e^{\mathbf{w}_2^T \mathbf{x} + c_2}}$$
$$P(\omega_2|\mathbf{x}, \Theta_2) = 1 - P(\omega_1|\mathbf{x}, \Theta_2)$$
$$a_1(\mathbf{x}) = \frac{e^{\mathbf{v}_1^T \mathbf{x} + d_1}}{\sum_{j=1}^{2} e^{\mathbf{v}_j^T \mathbf{x} + d_j}}$$
$$a_2(\mathbf{x}) = \frac{e^{\mathbf{v}_2^T \mathbf{x} + d_2}}{\sum_{j=1}^{2} e^{\mathbf{v}_j^T \mathbf{x} + d_j}}$$

where $P(\omega_1|\mathbf{x}, \Theta_j)$ is equal to $P(\mathbf{y} = [1\ 0]|\mathbf{x}, \Theta_j)$ in (1). Similarly, $P(\omega_2|\mathbf{x}, \Theta_j)$ is equal to $P(\mathbf{y} = [0\ 1]|\mathbf{x}, \Theta_j)$.

One solution is

$$\mathbf{w}_1 = [-1\ \ 1]^T, \qquad c_1 = 0$$
$$\mathbf{w}_2 = [1\ \ 1]^T, \qquad c_2 = -1$$
$$\mathbf{v}_1 = [-200\ \ 0]^T, \qquad d_1 = 100$$
$$\mathbf{v}_2 = [0\ \ 0]^T, \qquad d_2 = 0.$$

Expert 1 creates a decision boundary along the diagon line from (0, 0) to (1, 1), and expert 2 creates a decisi boundary along the antidiagonal line from (1, 0) to (0, 1 The gating network gives expert 1 higher confidence region $\{(x_1, x_2), 0 \le x_1 \le 0.5, 0 \le x_2 \le 1\}$, and giv expert 2 higher confidence at the other half.

b) RBF MOE networks: The MOE structure is natu for a RBF NN, in which each hidden node represe a receptive field with the following normalized Gaussi activation function:

$$a_j(\mathbf{x}) = \frac{\mu_j}{\sum_{k=1}^{K} \mu_k} \qquad ($$
$$\mu_k = \exp\left(-|\mathbf{x} - \mathbf{m}_k|^2 / 2\sigma_k^2\right)$$

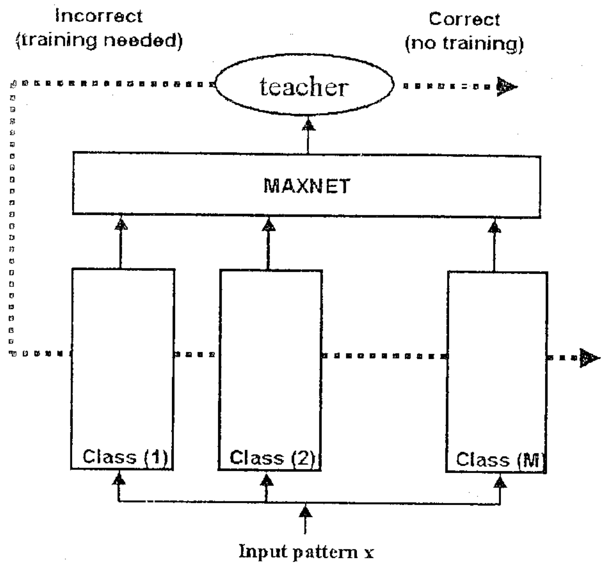
where $\mathbf{x}$ is the n-dimensional input vector and K is number of hidden nodes. The parameters μ_k and σ_k^2 den the mean and variance of the kth Gaussian function. (N that for LBF case, $\mu_k = [1 + \exp(-\mathbf{x}^T \mathbf{m}_k / \sigma_k)]^{-1}$.)

Then the output $y(\cdot)$ can be computed as the weigh sum of the activation values

$$y(\mathbf{x}) = \sum_{j=1}^{K} o_j a_j(\mathbf{x})$$

where o_j is the height of the jth Gaussian kernel.

The basic fuzzy NN defined in (5) and (11) can expressed in terms of the MOE structure. If the ga network output a_j in (1) is defined as the a_j in (3) the $P(\mathbf{y}|\mathbf{x}, \Theta_j)$ is defined to be a constant o_j, then



g. 4. The baseline configuration of a DBNN. In the DBNN, ie subnet is designated to represent one class. The output scores the subnets compete with each other, and the highest scoring bnet claims the identity of the input pattern. The decision-based arning rule is adopted for fine-tuning the decision boundaries tween the classes.

zzy NN becomes a MOE system with radial basis gating nction and constant expert output.

The MOE model has been applied to many applications. or application to sunspot time series prediction, a Bayesian amework for inferring the parameters of an MOE model ised on ensemble learning by variational free energy inimization proves to be very effective and performs gnificantly better than single networks [61]. MOE has also en shown to yield very good performance in automated tology screening applications [21]. For broader applica- on domains, MOE has also been extended to cope with multi-expert-level tree structure, known as hierarchical ixture of experts [25].

3) DBNN's: For most pattern recognition applications, e ultimate goal is to correctly assign an input pattern to hichever class it belongs. Therefore, it is quite natural consider class-level partitioning as part of a hierarchical sign. To introduce a class-level hierarchy into the FNN twork structure, the baseline DBNN serves as an illu- inating design example [31], [36]. As shown in Fig. 4, a BNN has a modular OCON network structure: one subnet designated to represent one object class. For multiclass assification problems, the outputs of the subnets (the scriminant functions) will compete with each other, and e subnet with the largest output values will claim the entity of the input pattern.

The learning scheme of the DBNN consists of two coupled phases: locally unsupervised and globally su- rvised learning. The purpose is to simplify a difficult imation problem by dividing it into several localized bproblems and, thereafter, the fine-tuning process would volve minimal resources.

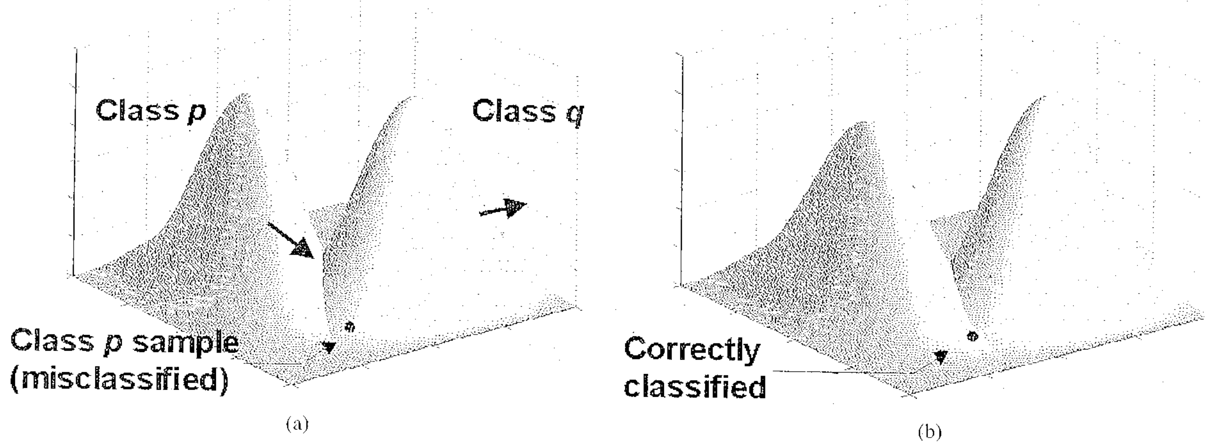
1) *Locally Unsupervised Learning via Vector Quanti- zation (VQ) or EM Clustering Method:* Several approaches can be used to estimate the number of hidden nodes, or an initial clustering can be determined based on VQ or EM clustering methods.
   a) In the hard-decision DBNN, the VQ-type clustering (e.g., K-mean) algorithm can be applied to obtain initial locations of the centroids.
   b) For the probabilistic DBNN, the EM algorithm can be applied to achieve maximum likelihood estimation for each class conditional likelihood density. (Note that once the likelihood densities are available, the posterior probabilities can be obtained easily.)
2) *Globally Supervised Learning Rules:* Based on the VQ or EM clustering, the decision-based learning rule can be applied to further fine tune the decision boundaries. In the second phase of the DBNN learning scheme, the objective of the learning process changes from maximum likelihood estimation to minimum classification error. Interclass mutual information is used to fine tune the decision boundaries (i.e., the globally supervised learning). In this phase, DBNN applies a reinforced–antireinforced learning rule [31] or a discriminative learning rule [28] to adjust network parameters. Only misclassified patterns are involved in this training phase.
3) *Reinforced–Antireinforced Learning Rules:* Suppose that the mth training pattern $\mathbf{x}^{(m)}$ is known to belong to class ω_i, and that the leading challenger is denoted $j = \arg\max_{j \neq i} \psi(\mathbf{x}^{(m)}, \Theta_j)$. The learning rule is

   Reinforced Learning:
   $$\Theta_i^{(m+1)} = \Theta_i^{(m)} + \eta \nabla \psi\left(\mathbf{x}^{(m)}, \Theta_i\right)$$
   Antireinforced Learning:
   $$\Theta_j^{(m+1)} = \Theta_j^{(m)} - \eta \nabla \psi\left(\mathbf{x}^{(m)}, \Theta_j\right) \qquad (6)$$

   where Θ_i is the weight vector of class i and η is a positive learning rate.

   Fig. 5 provides an illustration of the principle behind the decision-based learning rule. The learning rule is based on a minimal updating principle. The rule tends to avoid or minimize unnecessary side-effects due to overtraining. Given one training pattern, there are two scenarios. One is that the pattern is already classified correctly by the current network, in which case there will be no updating attributed to that pattern, and the learning process will proceed with the next training pattern. The second scenario is that the pattern is classified incorrectly to another winning class. In this case, parameters of two classes must be updated. The score of the winning class should be reduced by the antireinforced learning rule, while the score of the correct (but not winning) class should be enhanced by the reinforced learning rule.

a) Comparison of MOE and DBNN: The MOE and DBNN models share many similarities. For example, both modular structures are based on the "divide and conquer"



(a)

(b)

Fig. 5. In the DBNN, the winning class is the one with highest score function. The score functions of two competing classes (class p and class q) are depicted in the figure. In this example, the training pattern at issue is marked as a dark dot. The class p is assumed to be the correct class for the dark dot. So (a) shows a misclassification on the dot pattern as the class q shows a higher score than the class p. The DBNN learning rule adopts a minimal updating principle so that only the two classes immediately involved in the dispute should be updated. In (a), the dot is being incorrectly classified into class q. The action toward correcting such a mistake is twofold: 1) the reinforced learning rule will be applied to class p so as to enhance its score (e.g., by moving its centroid closer to the dark dot) and 2) the antireinforced learning rule will be applied to class q, which would result in a lower score (e.g., by moving its centroid further from the pattern). After multiple training sweeps, the boundary (respecting the pattern) should ultimately be placed on the correct side of pattern (respecting the boundary), as depicted in (b).

principle and both employ the EM algorithm in the training phase. Nevertheless, there are some substantial differences.

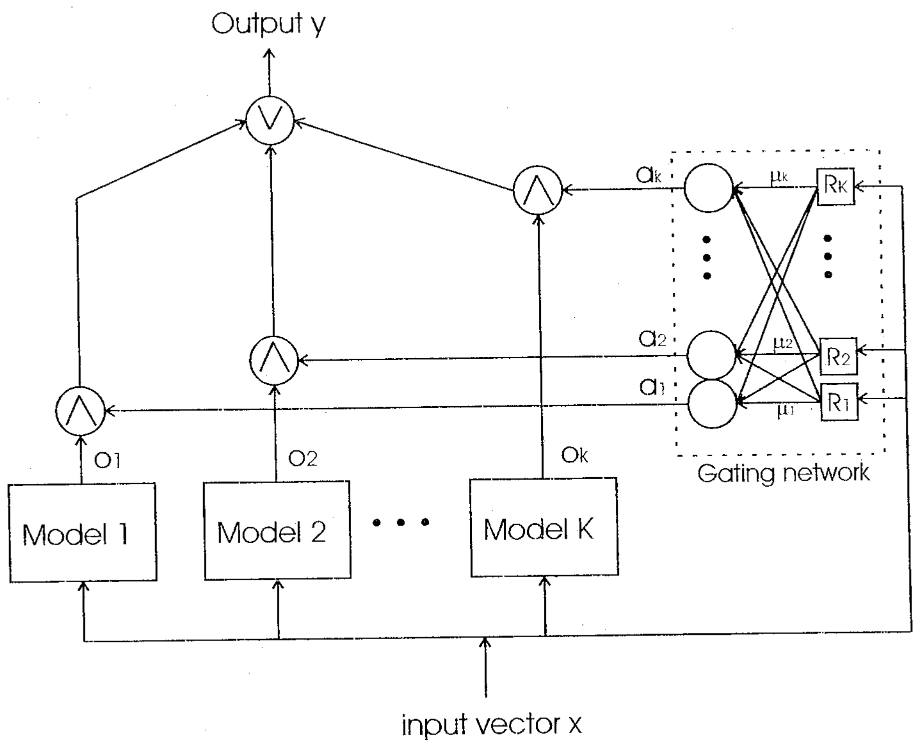
1) *Network Properties*: Each expert network in the MOE estimates the conditional posterior probabilities for all the pattern classes. The output of a local expert is ready to make the classification based on its own local information and expert's perspective. This characteristic suggests that the interclass communication/decision exists in the local network level under a MOE model. In contrast, each neuron in a DBNN estimates the class conditional likelihood density. The interclass communication/decision does not occur until the final subnet output is formed. This makes possible the absence of interclass communication across the (class) modules of the DBNN. (This in a sense achieves a truly distributive processing.)
2) *Training Strategies*: The training strategies of these two models are vastly different. During the MOE training, all the training patterns have the power to update every expert. The influence from the training patterns on each expert is regulated by the gating network (which itself is under training) so that as the training proceeds, the training patterns will have higher influence on the nearby experts, and lower influence on those far away. In contrast, unlike the MOE, the DBNN makes use of both unsupervised (EM-type) and supervised (decision-based) learning rules. The DBNN uses only the misclassified training patterns for its globally supervised learning. Moreover, unlike the MOE, which updates all the classes, the DBNN updates only the "winner" class and the class which the misclassified pattern actually belongs to. Its training strategy is to abide by a "n updating principle," as illustrated in Fig. 5.

B. Fuzzy Inference Systems

The basic idea behind a fuzzy inference systen incorporate a human's "expert experience" into the design. The input–output relationship is describec collection of fuzzy inference rules (e.g., If–Then involving linguistic variables. The typical architec a fuzzy system comprises four principal component

1) a fuzzifier which maps crisp numbers into s linguistic values;
2) a fuzzy rule base which stores the knowledge human experts and the empirical observation;
3) an inference engine which deduces the desired by performing approximate reasoning;
4) a defuzzifier which extracts a crisp value from a set as a representative value.

The fuzzy logic systems, in contrast to connec NN's, offer a structural framework with high-level If–Then rule thinking and reasoning. Fuzzy system their decisions on inputs in the form of linguistic va defined by membership functions which are formula to determine the fuzzy set to which a value belong the degree of membership in that set. The variabl then matched with the preconditions of linguistics If rules (fuzzy logic rules) to calculate the firing str of the rules, and the response of each rule is ob through fuzzy implication. Following the compos rule of inference, the response of each rule is we according to the rule firing strength, and usually the ce



ig. 6. In a Sugeno-type fuzzy classifier, the vector x is input to the models (local experts) and the ıting network. The gating network is corresponding to the "If" part of the "If–Then" rule. The R_i ode is the rule node which computes the matching degree of the input x for rule i. The nodes after e rule nodes normalize the matching degrees from the rule nodes to produce the weighting factor r the outputs of the local experts. The output y is a function of the outputs from local experts d the weighting factor from the gating network. The $\wedge$ and the $\vee$ denote the weighting and the ombining operators, respectively. The multiplication and addition are usually adopted to implement ese two operations. That is, the output is a linear combination of the outputs from local experts d the weighting factor from the gating network. This corresponds to the defuzzification stage. his figure also represents a basic common framework shared by NN's and fuzzy inference system.

f the responses is calculated to generate the appropriate utput.

For example, in Sugeno's fuzzy inference system shown n Fig. 6, the If–Then rules in the fuzzy rule base can be xpressed as

$$\text{If } \mathbf{x} \text{ is } R_j \text{ Then } y \text{ is Model}_j.$$

The fuzzification and firing strength are calculated in the ule node. The inference engine deduces its final output rom the outputs of the models. For example, such a 'defuzzified" output may be calculated as a weighted sum of individual outputs since each model produces crisp output. The weighted average is adopted as the inference procedure. In [35], this procedure is called fuzzy reasoning of the third type).

A multiple-input–single-output (MISO) fuzzy system can ake the following fuzzy rules:

$$\begin{cases} \text{Rule 1:} & \text{If } x_1 \text{ is } S_1^1 \text{ And } x_2 \text{ is } S_2^1 \text{ And } \ldots \\ & \quad \text{And } x_n \text{ is } S_n^1 \text{ Then } y \text{ is } O^1 \\ \text{Rule 2:} & \text{If } x_1 \text{ is } S_1^2 \text{ And } x_2 \text{ is } S_2^2 \text{ And } \ldots \\ & \quad \text{And } x_n \text{ is } S_n^2 \text{ Then } y \text{ is } O^2 \\ & \vdots \\ \text{Rule } K\text{:} & \text{If } x_1 \text{ is } S_1^K \text{ And } x_2 \text{ is } S_2^K \text{ And } \ldots \\ & \quad \text{And } x_n \text{ is } S_n^K \text{ Then } y \text{ is } O^K. \end{cases} \tag{7}$$

Computationally, the "If" part can be implemented as a fuzzy (or soft) weighting factor while a fuzzy "And" can be replaced by arithmetic multiplication.

1) LBF FIS Example: Using the same example as that for the MOE, we can design a Sugeno-type fuzzy classifier to define the correct decision boundary. The rule base contains the following two rules:

$$\begin{cases} \text{Rule 1:} & \text{If } x_1 \text{ is } \textit{Small} \text{ Then } y(\mathbf{x}) = O_s(\mathbf{x}) \\ \text{Rule 2:} & \text{If } x_1 \text{ is } \textit{Large} \text{ Then } y(\mathbf{x}) = O_l(\mathbf{x}) \end{cases} \tag{8}$$

where

$$O_s(\mathbf{x}) = \frac{1 - \exp\left(\mathbf{w}_1^T\mathbf{x} + c_1\right)}{1 + \exp\left(\mathbf{w}_1^T\mathbf{x} + c_1\right)}$$

and

$$O_l(\mathbf{x}) = \frac{1 - \exp\left(\mathbf{w}_2^T\mathbf{x} + c_2\right)}{1 + \exp\left(\mathbf{w}_2^T\mathbf{x} + c_2\right)}.$$

The membership functions for the fuzzy sets *Large* and *Small* are denoted as $m_l(\mathbf{x})$ and $m_s(\mathbf{x})$, respectively, and

$$m_s(\mathbf{x}) = \frac{\exp\left(\mathbf{v}_1^T\mathbf{x} + d_1\right)}{1 + \exp\left(\mathbf{v}_1^T\mathbf{x} + d_1\right)}$$

$$m_l(\mathbf{x}) = \frac{1}{1 + \exp\left(\mathbf{v}_1^T\mathbf{x} + d_1\right)}.$$

The $m_s(\mathrm{x})$ and $m_l(\mathrm{x})$ are sigmoid membership functions and $m_s(\mathrm{x}) + m_l(\mathrm{x}) = 1$.

Then the defuzzified output

$$y(\mathrm{x}) = \frac{O_s(\mathrm{x})m_s(\mathrm{x})}{(m_s(\mathrm{x}) + m_l(\mathrm{x}))} + \frac{O_l(\mathrm{x})m_l(\mathrm{x})}{(m_s(\mathrm{x}) + m_l(\mathrm{x}))}$$
$$= O_s(\mathrm{x})m_s(\mathrm{x}) + O_l(\mathrm{x})m_l(\mathrm{x}).$$

If a crisp decision is desired, the decision rule can be defined as

$$\text{If } y(\mathrm{x}) > 0 \text{ then x is in class 1 otherwise x is in class 2.} \quad (9)$$

A possible solution for the parameters in the FIS in order to separate the two regions is

$$\begin{aligned} \mathbf{w}_1 &= [-1\ \ 1]^T, & c_1 &= 0 \\ \mathbf{w}_2 &= [1\ \ 1]^T, & c_2 &= -1 \\ \mathbf{v}_1 &= [-200\ \ 0]^T, & d_1 &= 100. \end{aligned}$$

This solution is exactly the same as that in the LBF MOE example. The output $y(\mathrm{x})$ for the input region with $\{(x_1, x_2), 0 \le x_1 \le 0.5, 0 \le x_2 \le 1\}$ is dominated by the first rule and the rest by the second rule.

2) RBF Membership Function: It has recently become popular for a fuzzy system to utilize Gaussian membership functions and centroid defuzzification scheme. This is in part due to its capabilities of approximating any real continuous function on a compact set to an arbitrary accuracy, provided sufficient fuzzy logic rules are available [29], [57]. In the neural net literature, it has also been established that NN's with the normalized RBF's as the hidden nodes are universal approximators [50]. Therefore, NN's and fuzzy systems are similar in terms of their approximating capability.

They also bear very sharp structural resemblance. An illuminating evidence is realized by comparing Figs. 2 and 6 representing the MOE modular NN and Sugeno's fuzzy inference system, respectively. Stretching the similarity further, the intersection of fuzzy systems and NN's actually defines a large family of learning networks. The following will show this family of models can be built upon a common mathematical formulation and system architecture. In terms of learning capability, NN's with RBF's as hidden nodes are basically equivalent to fuzzy systems using: 1) Gaussian membership functions; 2) product inference; and 3) fuzzy rules with singleton consequents. This is elaborated below.

Now the membership functions μ_{lj} of the fuzzy sets S_l^j are assumed to be Gaussian-like functions with mean m_{lj} and variance σ_{lj}^2

$$\mu_{lj}(x_l) = \exp\left(-(x_l - m_{lj})^2/2\sigma_{lj}^2\right). \quad (10)$$

The output fuzzy set O^j is assumed to be a fuzzy singleton at o_j from the jth model. With the weighting determined by the If part, the defuzzified output is calculated by taking weighted contributions from different model outputs. Just like (5) in RBF NN, the output y in a centroid defuzzification scheme can be computed as

$$y = \sum_{j=1}^{K} o_j a_j(\mathrm{x}) \quad (11)$$

where

$$a_j = \frac{\mu_j}{\sum_{k=1}^{K} \mu_k} \quad \text{and} \quad \mu_k = \prod_{l=1}^{n} \mu_{lk}(x_l). \quad (12)$$

Therefore, by comparing (3) and (5) with (12) and (11), it is clear that the aforementioned RBF MOE NN and fuzzy inference system are essentially equivalent if the σ_{ij}'s in (10) are identical for all j or the σ_k's in (3) are independent for each input dimension. Having the same mathematical formulation, they naturally share very similar system architectures [cf. Figs. 2 and 6].

3) MOE and FIS Networks: In summary, the Sugeno-type FIS and the MOE networks are basically equivalen as long as the gating network of MOE generates the fuzzy membership values according to the membership functions and the And operation in the fuzzy If–Then rules. There are some subtle differences between the MOE and the Sugeno-type FIS, as listed in the following.

1) For the FIS, there usually exist shared parameters for the rule nodes due to the partition of the inpu space. Thus the rule nodes can be easily interpretec as linguistic rules. In contrast, the parameters in th MOE network are usually independent.
2) The output for each rule of Sugeno-type FIS is usually a (zeroth order or first order) polynomial function o the input, while the output for the expert in MOE i usually a nonlinear function.
3) If the gating network adopts elliptic basis function with a general covariance matrix, it becomes difficul to design an equivalent FIS using the rule: If x_1 is S And x_2 is S_2^i And ... And x_n is S_n^i Then y is O^i.

4) DBNN and FIS Networks: The class-level hierarch of DBNN and the rule-level hierarchy in the FIS ma be properly combined to yield fuzzy NN's useful fo specific applications. One successful design example i the neurofuzzy classifier named NEFCAR [55], which wil be discussed in Section III-C.

III. FNN'S

The learning process for a FNN involves mapping sampl data to the FNN's network parameters via both the un supervised learning and supervised learning. Unsupervise learning can be used to obtain the initial fuzzy rule bas from the sample data while supervised learning is effectiv in fine tuning the decision boundaries of the classifier. Bot the MOE and DBNN employ the EM algorithm whose derivation will be discussed in Section III-A.

In terms of the structural design, an effective implementation of FNN's hinges upon (a combination of) locally distributed and hierarchical networks. Local and distributed processing is critical to fault tolerance and robustness of the FNN's. A hierarchical design on the other hand often results in a more efficient network structure. Proper incorporation of expert-level modules and class-level modules into a hierarchical FNN can prove advantageous in computation and performance. A hierarchical FNN comprises a variety of fuzzy processing modules, for which EM serves as a basic tool. In addition, the decision-based learning rule proves to be effective in implementing a global (i.e., interclass) supervised training scheme. The structural designs and the associated learning algorithms will be elaborated in Section III-B. There are recently many important applications involving fusion of information from completely different sources. The hierarchical FNN structure can be easily extended to cope with multichannel information processing. Hierarchical fuzzy NN's with an embedded fusion agent offer an effective approach to channel fusion.

The FNN's draw inspiration from innovations within both the NN and fuzzy logic communities. Such FNN's strive to preserve the structure of the fuzzy systems and at the same time maximally exploit unsupervised and supervised learning rules in NN's. One should be able to learn rules in a hybrid fashion and calibrate them for better total-system performance. One hybrid design example will be elaborated in Section III-C.

A. EM Fuzzy Classifier

Unsupervised learning or clustering rules can be perceived as a result of natural evolution from the traditional statistical clustering and parameter estimation techniques. These often serve as a promising preprocessing step of a pattern recognition system to enhance the performance. Unsupervised learning algorithms may be applied to determine the initial weights for individual local experts. The initial clusters can be trained by VQ or K-mean clustering techniques [12]. For FNN's, the EM algorithm is a convenient tool to achieve maximum likelihood estimation for each class conditional likelihood density.

K mean and VQ are often used interchangeably: They classify input patterns based on the nearest-neighbor rule. K mean [12] can be treated as a special method for implementing VQ [18]. The task is to cluster a given data set $\mathbf{X} = \{\mathbf{x}_i | i = 1, \ldots, N\}$ into K groups, each represented by its centroid denoted by $\hat{\mathbf{X}} = \{\mathbf{x}'_j | j = 1, \ldots, K\}$. The nearest-neighbor rule assigns a pattern $\mathbf{x}$ to the class associated with its nearest centroid, say $\mathbf{x}'_i$. K mean and VQ have a simple learning rules and the classification scheme is straightforward. Mathematically, it aims at minimizing the following cost function:

$$E(h; \mathbf{X}) = \sum_{i,j} h_j(\mathbf{x}_i) |\mathbf{x}_i - \mathbf{x}'_j|^2 \tag{13}$$

where $h_j(\mathbf{x}_i) = 1$ for the members only, otherwise $h_j(\mathbf{x}_i) = 0$. [For example, $h_j(\mathbf{x}_i) = 1$ could indicate that $\mathbf{x}_i$ is closest to $\mathbf{x}'_j$ among all K centroids in $\hat{\mathbf{X}}$.] The K-mean algorithm [41] provides a simple mechanism for minimizing the sum of squared error with K clusters.

The iterations in the VQ clustering adopt a "hard-decision" rule, i.e., $h_j(\mathbf{x}_i)$ is either one or zero. In this sense, VQ is only a special case of a more general clustering algorithm, i.e., EM. The EM algorithm is a well-established iterative method for maximum likelihood estimation (MLE) [10] and for clustering mixtures of Gaussian distributions. It serves as a useful tool for estimating distribution parameters and thus results in data clustering. Most importantly, EM introduces a notion of "entropy" (or uncertainty) to induce a fuzzy classification, making it very amenable to the notions of fuzzy If–Then rule and fuzzy membership in fuzzy neural models.

1) EM for Fuzzy NN's: The EM algorithm can be perceived as a "soft" version of VQ, thus offering an efficient fuzzy clustering tool for the unsupervised learning phase in training FNN's. In addition, the EM algorithm offers several attractive attributes.

1) EM naturally accommodates model-based clustering formulation with one model corresponding to one rule used in FNN [cf. Fig. 6].
2) The EM allows the final decision to incorporate prior information. This could be instrumental to multiple-expert or multiple-channel information fusion.

The most common clustering is via either an RBF or a more general elliptic basis function. In the latter case, the component density $p(\mathbf{x}_i|\Theta_j)$ is a Gaussian distribution, with the model parameter of the jth cluster $\Theta_j = \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \pi_j\}$ consisting of the mean vector, the full-rank covariance matrix, and the prior probability.

The problem is to find an optimal estimation of a mixture-of-Gaussian likelihood functions $\prod_i p(\mathbf{x}_i)$ with respect to $\Theta = \{\Theta_j, j = 1, \ldots, K\}$, given a set of data $\mathbf{X} = \{\mathbf{x}_i | i = 1, \ldots, N\}$ as independent identically distributed samples

$$p(\mathbf{x}_i) = \sum_{j=1}^{K} \pi_j p(\mathbf{x}_i|\Theta_j) \qquad \sum_{j=1}^{K} \pi_j = 1 \tag{14}$$

where Θ_j represents the jth cluster, and π_j denotes its prior probability.

The optimal estimation is obtained by minimizing an energy function E defined from the negative logarithm of $p(\mathbf{x}_i)$

$$\begin{aligned} E &= -\sum_{i=1}^{N} \log p(\mathbf{x}_i) \\ &= -\sum_{i=1}^{N} \frac{\sum_{j=1}^{K} \pi_j p(\mathbf{x}_i|\Theta_j)}{p(\mathbf{x}_i)} \log p(\mathbf{x}_i). \end{aligned} \tag{15}$$

Define $h_j(\mathbf{x}_i) \equiv (\pi_j p(\mathbf{x}_i|\Theta_j)/p(\mathbf{x}_i))$. Equation (15) becomes

$$
\begin{aligned}
E &= -\sum_{i=1}^{N}\sum_{j=1}^{K} h_j(\mathbf{x}_i)\log p(\mathbf{x}_i)\\
&= \sum_{i=1}^{N}\sum_{j=1}^{K} h_j(\mathbf{x}_i)(-\log p(\mathbf{x}_i) + \log[\pi_j p(\mathbf{x}_i|\Theta_j)]\\
&\qquad\qquad - \log[\pi_j p(\mathbf{x}_i|\Theta_j)])\\
&= \sum_{i=1}^{N}\sum_{j=1}^{K} h_j(\mathbf{x}_i)(\log h_j(\mathbf{x}_i) - \log[\pi_j p(\mathbf{x}_i|\Theta_j)])\\
&= \sum_{i,j} h_j(\mathbf{x}_i)\log h_j(\mathbf{x}_i) - \sum_{i,j} h_j(\mathbf{x}_i)\log \pi_j\\
&\quad - \sum_{i,j} h_j(\mathbf{x}_i)\log p(\mathbf{x}_i|\Theta_j). \qquad (16)
\end{aligned}
$$

Note that $h_j(\mathbf{x}_i)$ equals the probability of $\mathbf{x}_i$ belonging to jth cluster given a prior model ($h_j(\mathbf{x}_i) = \Pr(\mathbf{x}_i \in \Theta_j|\mathbf{x}_i, \Theta)$). It can be considered as a "fuzzy" membership function.

By rearranging (16), we obtain the following energy function [19], [63]:

$$
\begin{aligned}
E &= \sum_{i,j} h_j(\mathbf{x}_i)s_j(\mathbf{x}_i, \mu_j, \Sigma_j) - \sigma_T \sum_{i,j} h_j(\mathbf{x}_i)\log \pi_j\\
&\quad + \sigma_T \sum_{i,j} h_j(\mathbf{x}_i)\log h_j(\mathbf{x}_i). \qquad (17)
\end{aligned}
$$

There are three terms in the energy function E.

1) The first term is the external energy, where $s_j(\mathbf{x}_i, \mu_j, \Sigma_j)$ denotes a weighted squared error
$$s_j(\mathbf{x}_i, \mu_j, \Sigma_j) = (\mathbf{x}_i - \mu_j)\Sigma_j^{-1}(\mathbf{x}_i - \mu_j)^T.$$
2) The second term represents the internal energy. For each sample $\mathbf{x}_i$, the internal energy term grasps the influence (prior probability) of its neighboring clusters.
3) The third term can be interpreted as the (negative) entropy term, which helps induce the membership's fuzziness.

The EM scheme can be seen as a fuzzy extension of K-mean clustering. In other words, K-mean clustering is a special case of the EM scheme. [Note that (17) would be reduced into (13) when Σ_j is an identity matrix and σ_T approaches zero.] By examining (18), this leads to a hard-decision clustering (i.e., with cluster probabilities equal to either one or zero). This demonstrates that σ_T plays the same role as the temperature parameter in the simulated annealing method. Note that the probability density function has the form $p(\mathbf{x}_i|\Theta_j) \propto e^{-s_j(\mathbf{x}_i, \mu_j, \Sigma_j)/\sigma_T}$, so the higher the temperature the greater the entropy. It is a common practice to use some sort of annealing temperature schedule, i.e., starting with a higher σ_T and then gradually decreasing σ_T to a lower value as iterations progress in order to force a more certain classification.

2) EM Iterations: We now modify the optimization formulation slightly (but causing no net effect). The EM problem can be expressed as one which minimizes E with respect to both: 1) the model parameters $\Theta = \text{3-D}\{\Theta_j, \forall j\}$ and 2) the membership function $\{h_j(\mathbf{x}_i), \forall i, j\}$. The interplay of these two variables can hopefully induce a bootstrapping effect facilitating the convergence process. This is further elaborated below.

1) In the E step, while fixing the model parameter $\Theta = \{\Theta_j, \forall j\}$, we find the best cluster probability h_j to optimize E [with constraint $\sum_{j=1}^{K} h_j(\mathbf{x}_i) = 1$]
$$h_j(\mathbf{x}_i) \propto \pi_j e^{-s_j(\mathbf{x}_i, \mu_j, \Sigma_j)/\sigma_T}. \qquad (18)$$
2) In the M step, we search for the best model parameter $\Theta = \{\Theta_j, \forall j\}$ which optimizes E, while fixing the cluster probability $h_j(\mathbf{x}_i), \forall i$.

The EM algorithm discussed above is directly applicable to the hierarchical FNN's as discussed in the next section. Note that in a particular iteration, the parameters to minimize the weighted-squared-error $s_j(\mathbf{x}_i, \mu_j, \Sigma_j)$ can be obtained analytically, which is the special advantage of using RBF-type likelihood functions. On the other hand, if a linear model (e.g., LBF) is chosen to parameterize the likelihood and/or the gating functions, we will need an iterative method to achieve the optimal solutions in the iteration. In other words, the EM algorithm becomes a double-loop optimization. For example, Jordan and Jacobs [26] applied a Fisher scoring method called iteratively reweighted least squares (IRLS) to train the LBF MOE network.

3) Application Examples: As illustrated in Section IV-A, EM has been successfully applied to many pattern classification applications.

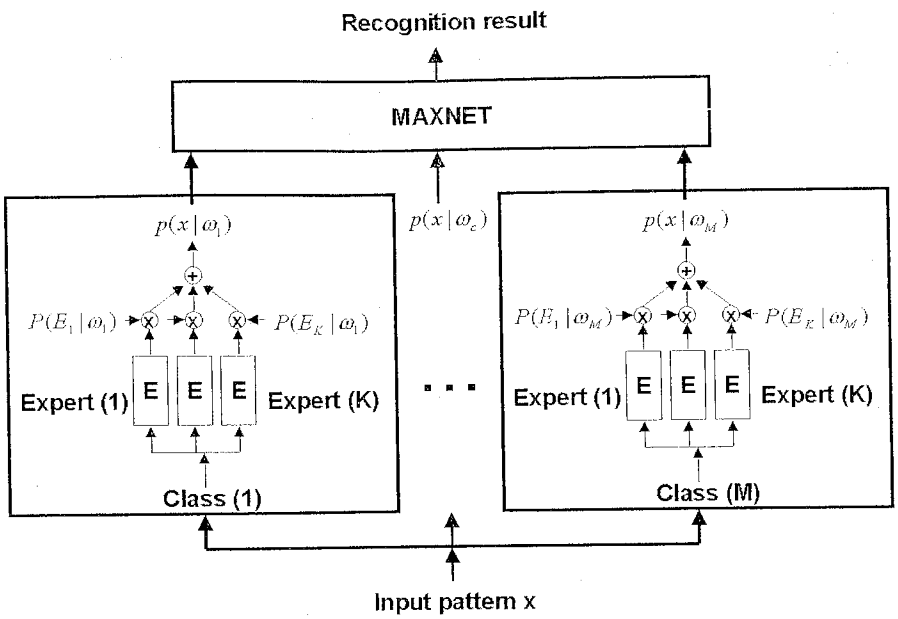
B. Hierarchical FNN for Fuzzy Decision

As mentioned in the previous sections, the MOE and DBNN networks adopt the expert-level and class-level partitioning strategies, respectively. A hierarchical FNN configuration should allow the incorporation of both the expert-level and class-level modules. As elaborated below, the selection of the inner blocks versus the outer blocks will lead to very distinctive hierarchical structures.

1) Experts-in-Class Hierarchical Structures: Fig. 7 depicts such a structure. The inner blocks comprise expert-level modules, while the outer blocks are on the class level. A typical example of this type of network is the hierarchical DBNN [38], which describes the class discriminant function as a mixture of multiple probabilistic distributions. That is, the discriminant function of the class ω_c in the hierarchical DBNN is a class conditional likelihood density $p(\mathbf{x}_i|\omega_c)$

$$p(\mathbf{x}_i|\omega_c) = \sum_{k=1}^{K} P(E_k|\omega_c)p(\mathbf{x}_i|\omega_c, E_k) \qquad (19)$$

where $p(\mathbf{x}_i|\omega_c, E_k)$ is the discriminant function of subnet c in expert k, and $p(\mathbf{x}_i|\omega_c)$ is the combined discriminant function for class ω_c. The expert confidence $P(E_k|\omega_c)$ can



. Experts-in-class hierarchical design. For the applications where there are several experts ormation sources, experts-in-class hierarchical scheme can be applied for classification. (The networks are omitted in the drawing.) Like the baseline DBNN model, the minimal updating ole can be adopted to train the parameters in the network. $P(E_j|\omega_c)$ serves as the confidence tor for the jth expert in class c. It is a trainable parameter, and its value is fixed during ing time.

arned by the EM algorithm described below. Define $= P(E_k|\omega_c)$ and set the initial value of $\alpha_k = 1/K$, $= 1, \ldots, K$. At the iteration step m

$$h_k^{(m)}(\mathbf{x}_i) = \frac{\alpha_k^{(m)} p(\mathbf{x}_i|\omega_c, E_k)}{\sum_l \alpha_l^{(m)} p(\mathbf{x}_i|\omega_c, E_l)},$$

$$\alpha_k^{(m+1)} = \frac{1}{N}\sum_{i=1}^{N} h_k^{(m)}(\mathbf{x}_i). \tag{20}$$

ı expert processes only the local features from its esponding class. The outputs from different experts are ırly combined. The weighting parameters, $P(E_k|\omega_c)$, esent the confidence of expert E_k producing the correct ver for the object class ω_c. Once they are trained he learning phase, their values remain fixed during retrieving (or identification) phase. By definition, $_{=1} P(E_k|\omega_c) = 1$, where K is the number of experts ıe subnet ω_c. So it has the property of a probability tion. Note that, in conjuncture with the expert-level (or -level) hierarchy, each hidden node within one class ıet may be used to model a certain local expert (or) with a varying degree of confidence which reflects ability to interpret a given input vector. The locally ıpervised and globally supervised scheme described in ion II-A3 can be adopted to train the OCON network Fig. 4].

) *Classes-in-Expert Hierarchical Structures:* Fig. 8 des such a structure. The inner blocks comprise class lules while the outer blocks are the expert modules. h expert has its own hierarchical DBNN classifier. The outs of the hierarchical DBNN's are transformed to the posterior probabilities by softmax functions. In this fusion scheme, the expert weighting $P(E_j|\mathbf{x})$ is a function of input pattern $\mathbf{x}$. Therefore, the importance of individual experts may vary with different input patterns observed.

This network adopts the posterior probabilities of electing a class given $\mathbf{x}_i$ [i.e., $P(\omega_c|\mathbf{x}_i, E_k)$]—instead of the likelihood of observing $\mathbf{x}_i$ given a class [i.e., $p(\mathbf{x}_i|\omega_c, E_k)$]—to model the discriminant function of each cluster. For this version of hierarchical FNN, a new confidence $P(E_k|\mathbf{x}_i)$ is assigned, which stands for the confidence on expert k when the input pattern is $\mathbf{x}_i$. Accordingly, the probability model is modified to become

$$P(\omega_c|\mathbf{x}_i) = \sum_{k=1}^{K} P(E_k|\mathbf{x}_i) P(\omega_c|\mathbf{x}_i, E_k) \tag{21}$$

where

$$P(\omega_c|\mathbf{x}_i, E_k) = \frac{P(\omega_c|E_k) p(\mathbf{x}_i|\omega_c, E_k)}{p(\mathbf{x}_i|E_k)},$$

$$p(\mathbf{x}_i|E_k) = \sum_c P(\omega_c|E_k) p(\mathbf{x}_i|\omega_c, E_k) \tag{22}$$

and the confidence $P(E_k|\mathbf{x}_i)$ can be obtained by the following equation:

$$P(E_k|\mathbf{x}_i) = \frac{P(E_k) p(\mathbf{x}_i|E_k)}{\sum_l P(E_l) p(\mathbf{x}_i|E_l)}. \tag{23}$$

Equation (22) is one type of the softmax function. Its weighting parameter $P(\omega_c|E_k)$, just like $P(E_k|\omega_c)$ in (19), is made to satisfy the property of a probability function [i.e., $\sum_c P(\omega_c|E_k) = 1$]. Consequently, $P(\omega_c|E_k)$ can be recursively estimated by EM iterations in (20). The term

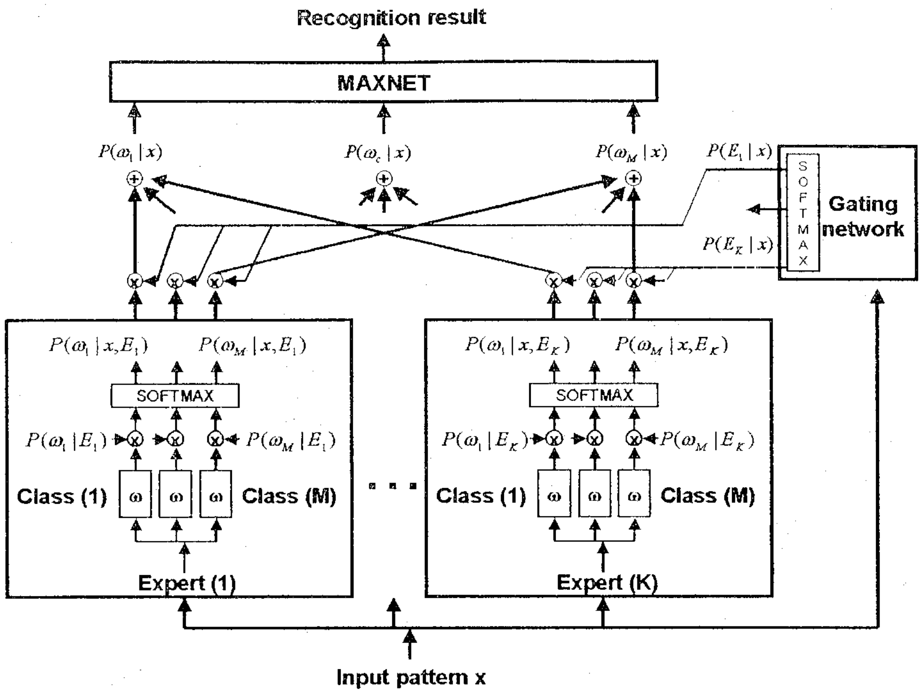


Fig. 8. Classes-in-expert hierarchical design. In this scheme, the expert weighting parameters $(P(E_j|\mathbf{x}))$ are functions of the input pattern $\mathbf{x}$. The prior probability $P(\omega_i|E_j)$ could either be preassigned or be estimated in a similar fashion as other prior probabilities such as $P(\Theta_k|\omega_i, E_j)$ [cf. (20)]. This network can be viewed as a hybrid model of MOE and DBNN, where local expert networks serve as local classifiers. When applying such a hierarchical FNN to channel fusion, each channel module can be regarded a local expert with predetermined feature space.

$P(E_k)$ may be interpreted as the confidence-level on expert k, which can be learned by the EM iterations very similar to (20). The confidence level $P(E_k)$ provides the key factor affecting the fusion weight $P(E_k|\mathbf{x}_i)$ which is the output of the softmax gating network. It can also be shown that $P(E_k)$ can be learned by (20) with slight modification. We note that, unlike the experts-in-class approach, the fusion weights need to be computed for each testing pattern during the retrieving phase.

3) Channel Fusion: The problem of combining the classification power of several classifiers is of great importance to various applications. In many remote sensing, pattern recognition, and multimedia applications, it is not uncommon that different channels or sensors could be provided to facilitate the recognition of an object. In addition, for many applications with very high-dimensional feature data, it provides some computational relief to divide feature vectors into several lower dimensional vectors before integrating them for final decision (i.e., divide and conquer).

Several categories of fusion layers have been studied in [2]. In the present context, the classes-in-expert hierarchical FNN can be naturally extended to cover the sensor fusion network. To this end, the definition of experts needs to be properly expanded. Now the local experts include not only: 1) the adaptive-trained type but also 2) the predetermined type. For the former, the feature space (often a certain local region) represented by an expert is adaptively trained in *a posteriori* fashion. The model parameters depend very much on the initial condition the local expert is assigned. As to the predetermined experts, each local expert has fixed model parameter, as it is designated to extract certain prior known feature space (e.g., high or low frequency components). By regarding each sensor as a predetermined local expert, the classes-in-expert hierarchical FNN can be made amenable to sensor fusion applications.

More exactly, the sensor-fusion FNN consists of several classifier channels, each of which receives input vector from its own sensor. Its structure (although not shown) resembles that of the classes-in-expert FNN. The fusion of different channels can be implemented as a (properly gated) linear combination of outputs, with the gating parameters reflecting the confidence measures of the sensor channels. This scheme can make a good use of the Bayesian estimation formulation and also facilitate adoption of EM training of the channel confidence parameters.

4) Application Examples: Hierarchical FNN's have found numerous important object detection and pattern recognition applications. Some illuminating examples will be provided in Section IV-B.

C. Neurofuzzy Classifiers with Adjustable Rule Importance

The class-level hierarchical structure used in the DBNN and the interpretation of fuzzy If–Then logic rules can be combined to establish the connections between the hidden nodes and the output nodes. One good example is the neurofuzzy classifier called NEFCAR, which is built upon the OCON structure [55]. Based on the interpretation of fuzzy If–Then rules, the connections between the hidden nodes and the output nodes in the OCON structure are

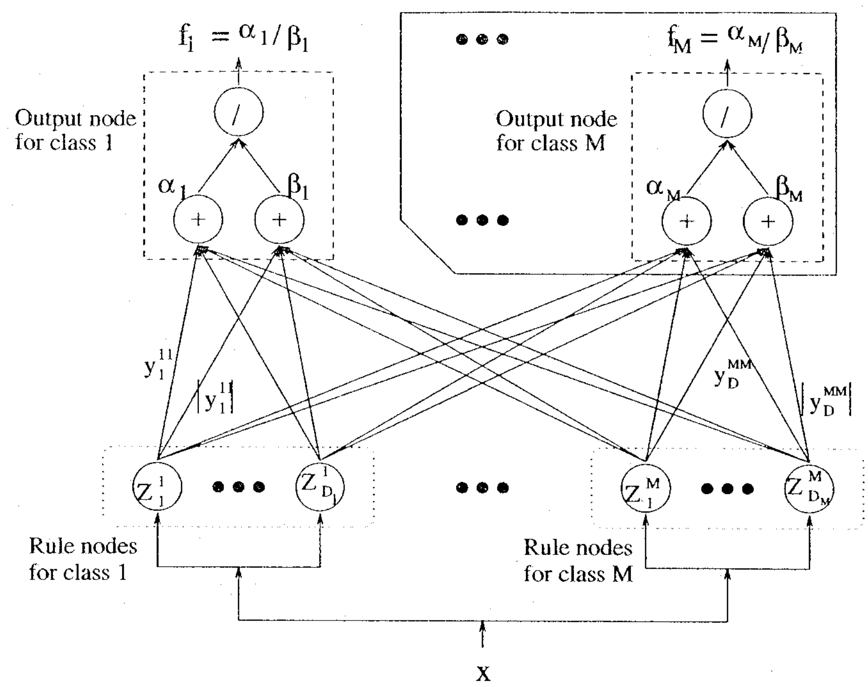


Fig. 9. Block diagram of the neurofuzzy classifier.

established. Both positive and negative rules are utilized and they are given varying and adjustable rule importances. Simulation studies point to the result that the classification rate can be improved by adding the cross connections. This demonstrates that the system performance may be improved by properly combining neural learning techniques and the fuzzy linguistic structure.

The NEFCLASS as proposed in [45] and [46] has similar structure to the NEFCAR design. The NEFCLASS architecture resembles that of a multilayer network whose weights are described as fuzzy sets. The backpropagation learning rule is adopted to train the NEFCLASS model. The design purpose is to derive a fast converging model which: 1) can be interpreted in terms of linguistic rules and 2) can use prior rule-based knowledge. The key differences between NEFCLASS and NEFCAR are the following. In NEFCLASS, the connections between the rule nodes and the output nodes are fixed to one, i.e., there are no "negative" rules. Moreover, some parameters in NEFCLASS are shared by different rules and they cannot be trained independently, unlike NEFCAR.

1) NEFCAR Architecture: To facilitate the discussion, we assume that the input pattern, the dimension of input pattern, the number of classes, and the number of rule nodes in class j are denoted as x, N, M, and D_j, respectively. The fuzzy rules of NEFCAR for an N-dimensional input pattern $\mathbf{x} = \{x_1, \ldots, x_N\}$ are of the form

$$\begin{cases} \text{Positive Rule:} & \text{If } x_1 \text{ is } \mu_1 \text{ And } \ldots \text{ And } x_N \text{ is } \mu_N, \\ & \quad \text{Then x is in class } i \\ \text{Negative Rule:} & \text{If } x_1 \text{ is } \mu_1 \text{ And } \ldots \text{ And } x_N \text{ is } \mu_N, \\ & \quad \text{Then x is } \textit{not} \text{ in class } k \end{cases} \tag{24}$$

where μ_l's are the fuzzy sets with Gaussian-like membership functions. The consequent of the positive rule, "x is in class i," is represented as a classifier output variable $f_i = 1$. On the other hand, the consequent of the negative rules is represented as an output $f_k = -1$. The schematic diagram of the architecture of the classifier is shown in Fig. 9, which is designed for a M-class classifier. In our design, each class has its own set of rule nodes and rule nodes are fully connected to the output nodes. A rule acts as a positive rule when it contributes to the output of its own class. Otherwise, it acts as a negative rule. Therefore, for a rule node in class j, we have effectively one positive rule and $(M-1)$ negative rules.

We assume the height defuzzification approach described in [11] is adopted to compute the output of each class. And the Z_d^j denotes the matching degree of the If part of the dth rule in class j.

The importance of the dth rule in class j to the output of class i is represented by a weighting factor y_d^{ij}. Such a factor can be interpreted as duplicating the same rule y_d^{ij} times in the defuzzification procedure for output i. This notion may be easily extended to the situation of noninteger y_d^{ij}. In general, the corresponding rule becomes more important to output f_i when y_d^{ij} becomes larger.

Assume that the mean and the variance of the lth (Gaussian-like) membership function of the dth rule for the jth class are denoted as m_{dl}^j and σ_{dl}^j, respectively. Moreover, the multiplication is used in lieu of the And operation in the rule in (24). Then the crisp defuzzified output of class i can be computed as follows:

$$f_i^* = \alpha_i/\beta_i, \qquad \alpha_i = \sum_{d=1}^{D_i} y_d^{ii} Z_d^i - \sum_{j=1,\, j\neq i} \sum_{d=1}^{D_j} y_d^{ij} Z_d^j$$

and

$$\beta_i = \sum_{j=1}^{}\sum_{d=1}^{D_j} y_d^{ij} Z_d^j \tag{25}$$

where

$$Z_d^j = \exp\left(-\sum_{l=1}^{N} \frac{(x_l - m_{dl}^j)^2}{(\sigma_{dl}^j)^2}\right). \tag{26}$$

The $(+1)$ and (-1) may be absorbed into y_d^{ij} and this leads to

$$\alpha_i = \sum_{j=1}^{}\sum_{d=1}^{D_j} y_d^{ij} Z_d^j \qquad \beta_i = \sum_{j=1}^{}\sum_{d=1}^{D_j} |y_d^{ij}| Z_d^j.$$

Therefore, we can consider (y_d^{ij})'s as weighting parameters in the network shown in Fig. 9. From the above equations, it is clear that

$$-1 \leq f_i^* \leq 1, \qquad \text{for } i = 1, \ldots, M.$$

The classification procedure is described as follows. The pattern is input to each rule node which computes the matching degree (Z_d^j) of the If part of the rule. Then the output of each class (f_i^*) is computed according to the weighting of each rule. The class c which has the largest output is identified. That is

$$c = \arg\left(\max_i f_i^*\right). \tag{27}$$

The pattern is classified into c if the confidence measure [cf. C_m^i in (29)] is large enough. Similarly, if we are dealing with a two-class application, the region in solid polygon and the weights connected to it, which are shown in Fig. 9, can be removed. In this situation, the sample is classified into class 1 if $f_1^* > 0$; otherwise, it is classified into class 2.

2) Training Algorithm: The locally unsupervised and globally supervised technique is adopted. It means that for the parameters in the network of a given class, we utilize an unsupervised clustering method to obtain the initial values using the patterns in the same class. Training patterns from all classes are used to update the classifier after the initialization. The reinforced and antireinforced learning rules are adopted to training the neurofuzzy networks. The decision-based and approximation-based strategies are combined to provide a balanced amount of supervised training. As a result, the classifier can reach convergence quickly.

3) Initialization: Since the linguistic rules for describing how to classify the data are not available, we adopt the K-mean algorithm to cluster the training data to obtain the initial $(m_{dl}^j)'s$ in this study. The $(\sigma_{dl}^j)'s$ are set to one initially. The importance factors for the positive rules are set to one and those for the negative rules are set to small negative numbers.

4) Combined Training Strategy: The training strategy of NEFCAR combines both the approximation-based and decision-based training strategies. As in the approximation-based approach, each pattern is assigned a target value (-1 or $+1$) for each output. Note that from the formula for computing the output values in (25) and (26), the output values always fall into the closed interval $[-1, 1]$. Let the teacher vector for a pattern from class k be denoted as $T = \{t_1, t_2, \ldots, t_M\}$ and have the value

$$\begin{cases} t_i = -1, & i \neq k \\ t_i = +1, & i = k. \end{cases} \tag{28}$$

Similar to the decision-based training, not all the training patterns are involved in the updating. In our design, if the output reaches certain noise tolerance value, the training is omitted. In this case, the pattern with larger error will be trained with a larger training ratio. This can give each data a proper training weighting without forcing all of them to produce the (artificially given) limiting values (-1 or $+1$). Therefore, the training procedure can reach a compromise between training patterns more easily and can converge faster.

5) Confidence Measure: In the training phase, a confidence measure can be used to avoid the outliers. If the pattern is from class i, we can define the confidence measure as

$$C_m^i = \left(\sum_{d=1}^{D_i} y_d^{ii} Z_d^i\right) \Big/ y_{\max} \tag{29}$$

where $y_{\max}$ is defined as

$$y_{\max} = \max_{d,i,j} y_d^{ij}. \tag{30}$$

The C_m^i measures the normalized sum of matching degrees of the rule antecedent of the rules from class i.

In the training process, if C_m^i is smaller than a threshold for a training sample, then this sample is ignored to avoid the training on the outliers and the numerical difficulties. Let C_{ma} denote the average of the confidence measure, that is

$$C_{ma} = \sum_{i=1} \sum_{x \in \text{Class } i} C_m^i / N_x$$

where N_x denotes the total number of patterns. The C_{ma} is calculated for each epoch of training. When the value of the average of the confidence measure becomes too small, it indicates that the parameters in the rule nodes cannot model the distribution of the training patterns well. This may result in degrading the generalization performance. Therefore, once the C_{ma} is smaller than a threshold, the training process is stopped. Since the Gaussian-like functions are adopted as the membership functions, the backpropagation algorithm can be easily implemented to calculate the gradients for all the parameters in the neurofuzzy classifier.

6) Application Examples: NEFCAR represents a novel fuzzy inference network for many pattern classification problems. For more application details, see Section IV-C.

IV. Applications of FNN's

FNN's have become a competitive means for many applications from low-level image processing to high-level pattern recognition. In particular, we shall highlight how FNN's can play a key role in video/image indexing, browsing, and content search in multimedia information systems [27], [49]. We shall present examples of texture classification and image/video segmentation, with several applications to medical and multimedia processing. The

FNN's can cope well with a variety of cues such as texture, intensity, edge, color, and motion. FNN's are also useful for detection and recognition of high-level features. We shall discuss application examples where FNN's can successfully facilitate recognition tasks such as detection of bank notes, recognition of human faces, or identification of breast cancer cells.

A. Applications of EM Fuzzy Classifiers

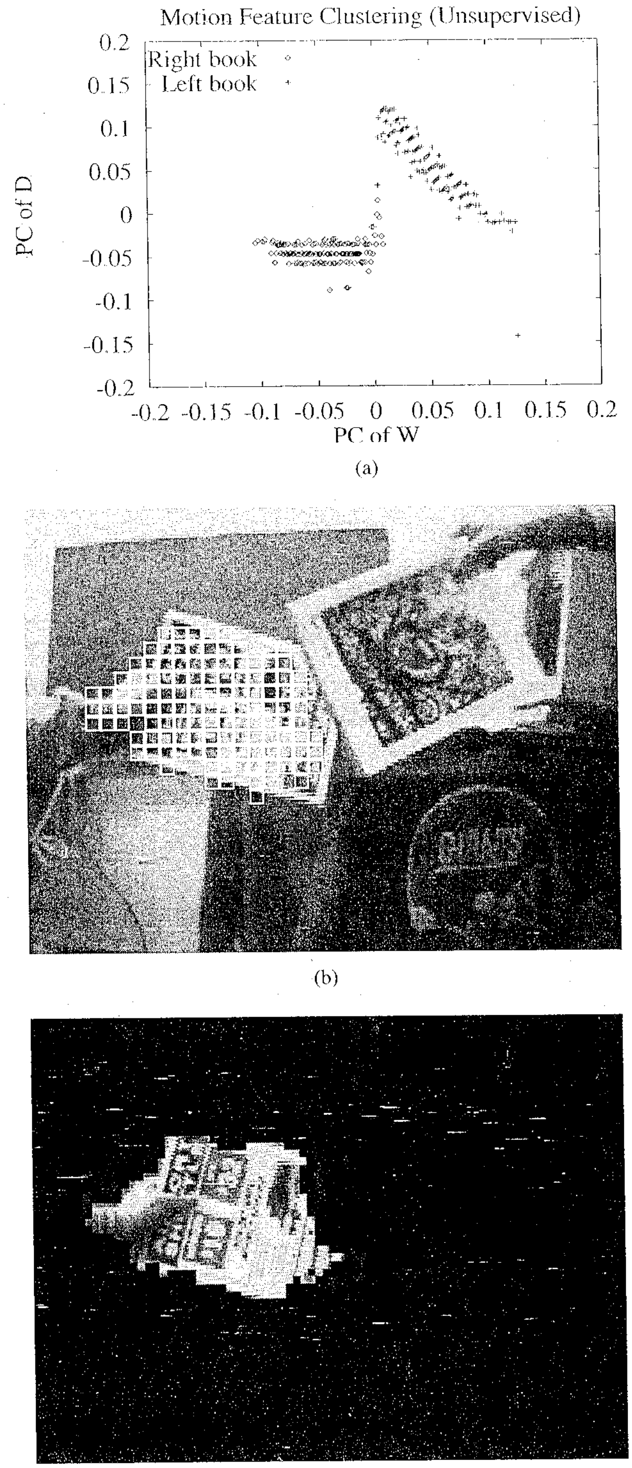
1) Motion-Based Video Segmentation: Robust scene segmentation is a prerequisite for the object-based video processing and high-performance video compression. Various approaches to this complex task have been proposed, including classification of motion flow, color/texture segmentation, and dominant-motion extraction. In [39] and [40], the EM method is applied to object-oriented motion segmentation, which divides a video scene into different motion regions. The procedure is briefly described below.

Initially, motion clustering is performed upon a selected set of motion feature blocks tracked by a true motion tracker. The feature blocks are represented by the principal components (PC) of their position and velocity. The motion parameters for each of the clustered feature blocks may be estimated and used as the initial condition for the final segmentation process. The final segmentation and the corresponding motion parameters are iteratively updated by a model-based EM algorithm. The model-based EM is based on minimization of the following energy function (where b denotes the image blocks):

$$E(\mathbf{A}, h) = \sum_{b,\,j} h_j(b) s_j(b,\, A_j) - \sigma_T^2 \sum_{b,\,j} h_j(b) \log \pi_j(b) + \sigma_T^2 \sum_{b,\,j} h_j(b) \log h_j(b). \tag{31}$$

The first term represents the external (error) energy function, so that each cluster (say jth cluster) would be best fit to a given motion (say affine) model denoted by A_j. The second term represents an internal energy function, which enforces the region smoothness by allowing neighboring blocks of a target block to exercise proportional influence on the classification of that target block. This serves the purpose of forcing the classification to take into account the intensity/texture continuity (i.e., image cues) and resulting in a smoother segmentation. The third term stands for the entropy function which encourages a softer classification.

Fig. 10 shows an example of video scene segmentation containing two moving books shot from a moving camera. Fig. 10(a) shows the distribution of the feature blocks in the PC coordinate. The unsupervised EM clustering scheme is then adopted to cluster the feature blocks. Fig. 10(b) shows the feature blocks classified into the left book. Fig. 10(c) demonstrates that the object region of the left-book as extracted by the EM-based segmentation.



(c)

Fig. 10. Motion-based video segmentation result of sequence with 2-Books with moving camera. (a) Distribution in terms of the PC's of the feature blocks obtained by a true motion tracker. Vertical axis: PC of motion (D). Horizontal axis: PC of position (W). (b) Feature blocks of the left book clustered by EM. (c) Final segmentation (of the left book) by a multicue model-based EM.

2) Texture Classification via Intraclass EM Clustering: The texture information can be the gray level of a pixel or the texture features of the neighborhood around a pixel. The objective of texture classification is to determine the boundary of a region where the image has the same textural characteristics. Both the unsupervised or supervised approaches are useful for the texture classification application. If there is no representative training image available in the application, then unsupervised techniques are the only options available. The EM algorithm described in Section III-A has been adopted as an effective means for texture classification, which can be used as a core unsupervised clustering technique. Satisfactory performances of the unsupervised EM algorithm for segmenting satellite images and medical images have been reported [9], [42], [51]. Nevertheless, the supervised technique can usually deliver much better segmentation results. An example based on intraclass EM clustering will be described here. As a comparison, another example using interclass supervised learning will be discussed in the next section.

It is assumed that the probability distributions of texture information of different regions in an textured image can be represented by statistical models. The EM algorithm can be adopted to estimate the probability density and a Bayesian classification approach can be adopted to segment the images. The neighborhood (spatial) information can be considered in the probability distribution model or used in a post-processing stage to produce a smooth segmentation. In [48], the EM technique is applied to image segmentation. The training images from a (known) texture class are used to obtain a representative statistical model. The classification is accomplished by centering a neighborhood at each pixel location, and choosing for that pixel the class whose conditional probability (given a texture model) is the highest. The preliminary result usually has a "noisy" appearance. One way to smooth the boundary is to make a good use of the spatial homogeneity. More precisely, for a pixel to be assigned to a specific class, its neighborhoods must also have reasonably high conditional likelihood (given the same texture model). According to the experimental report [48], the total error rate is below 5% for segmenting a four-square-region image comprised of four Brodatz textures. It was also observed that the overall error rate could be greatly reduced (to 1%) by first applying lowpass filtering to perform spatial averaging [9], [42], [51].

B. Applications of Hierarchical FNN's

1) Texture Classification via Global (Interclass) Supervision: In Section IV-B, the EM classifier does not make a full use of the interclass supervised training which directly tackle any mutual dispute which might exist between two competing classes. Substantial improvement can be expected by fully utilizing global interclass supervision. This idea has prompted [54] to propose an experts-in-class hierarchical texture classifier, based on the decision-based (reinforced and antireinforced) learning rule. Twelve Brodatz texture images were tested in the experiments and the hierarchical classifier achieved an extremely low classification error rate (as low as 0.2%). Such a superior performance is partially due to the adoption of global supervision. It is also partially due to the adoption of a novel texture feature, called fuzzy texture spectrum. The fuzzy texture spectrum, based on the relative differences of the gray levels between pixels, appears to be fairly insensitive to noise and changing of the background brightness in texture images.

2) Face Recognition and Content-Based Indexing for Video Browsing: Face recognition is a user-friendly, nonintrusive, and convenient approach to biometric identification. Many NN's have been proposed for face recognition [17], [33], [38], [56]. For examples, Brunelli and Poggio has adopted an RBF network for face recognition [4]. Cox *et al.* [7] proposed mixture-distance VQ network for face recognition and reached a 95% rate on a large database (685 persons). Fuzzy decision-based NN's have shown a very good success in face recognition [38]. By combining facial information with other biometric features such as speech, feature fusion not only enhances accuracy but also provides some fault tolerance, i.e., it could tolerate a temporary corruption of one of the bimodal channels. In [36] and [37], NN's have been successfully applied to find such patterns with specific applications to detecting human faces and locating eyes.

In many video applications, browsing through a large amount of video material to find the relevant clips is an extremely important task. The video database indexed by human faces, exemplified by Fig. 11, provides users the facility to efficiently acquire video clips about the person of interest. For example, a film-study student may conveniently extract the clips of his/her favorite actor/actress from movie archive to study his/her performance, and a TV news reporter may quickly find out from news database the clips containing images of some politician in order to edit headline news.

Fuzzy neural processing presents a promising approach to fast access of audiovisual objects, manipulating them, and presenting them in a highly flexible way. By extracting proper information content inherent in video clips, it leads to efficient search schemes for content-based retrieval. A video indexing and browsing scheme based on human faces is proposed in [37] and [38]. The scheme adopts the (experts-in-class) hierarchical FNN for face detection and recognition techniques. The scheme contains three steps. The first step of our face-based video browser is to segment the video sequence by applying a scene change detection algorithm. Scene change detection gives indication of when a new shot starts and ends. Each segment (shot) created by scene change detection can be considered as a story unit of this sequence. From every video shot, we take its representative frame and feed it into a probabilistic DBNN face detector [38]. Those representative frames from which the detector gives high face detection confidence scores are annotated and serve as the indices for browsing.

This scheme can also be helpful for constructing hierarchies of video shots for the purpose of video browsing. One such algorithm [62], for example, proposes using global color and luminance information as similarity measures to cluster video shots in an attempt to build video shot hierarchies. Their similarity metrics enable very fast processing of videos. However, in their demonstration, some shots featuring the same anchorman fail to be grouped together due to insufficient image content understanding. For this type of applications, we believe that the existence of similar objects, and human objects in particular, should

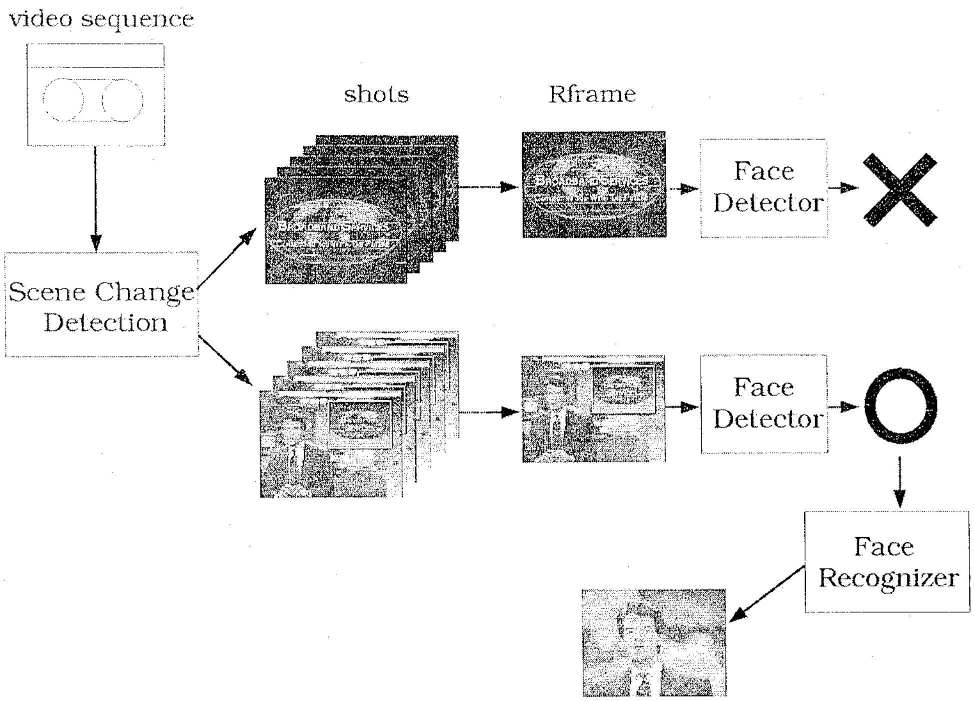


Fig. 11. Probabilistic DBNN face-based video browsing system. Face detector examines all the representative frames to see if there contain human faces and if so the face detector passes the frame to face recognizer to identify whose face it is.

provide a good similarity measure. As reported in [5] and [6], this scheme successfully classifies these shots to the same group.

3) Currency Recognition: With the recent improvements in the picture quality of color scanners and printers, many national laws explicitly prohibit copying of specific originals such as currencies, bank notes, or securities. Therefore, high-end color copiers are required by law to incorporate antiforgery modules so that any attempt of counterfeiting or unauthorized duplications of registered patterns from their copiers will be detected and denied. Bank notes or currency recognition for copiers is a challenging task. Unlike other special-purpose currency discrimination machines, the antiforgery module in copiers cannot use a handling mechanism to detect the physical characteristics of the currencies, nor can it reply on some antiforgery features such as metallic stripes or special ink, unless special sensors are included. Moreover, to recognize currencies from tens of countries by the limited computation resource in the color copiers requires a very efficient and accurate pattern recognition engine.

FNN's have been successfully applied to antiforgery and currency discrimination. A hierarchical DBNN recognizer has been designed to detect currency patterns on the image scanned by the color copier. The recognition engine adopts hierarchical pattern recognition approach for money recognition, i.e., both local texture features and global structural information of the registered patterns are utilized in the recognition process. After image preprocessing, the engine tries to detect the presence of special currency features (e.g., portrait, notes, background patterns) in the scanned image by applying the hierarchical experts-in-class FNN's.

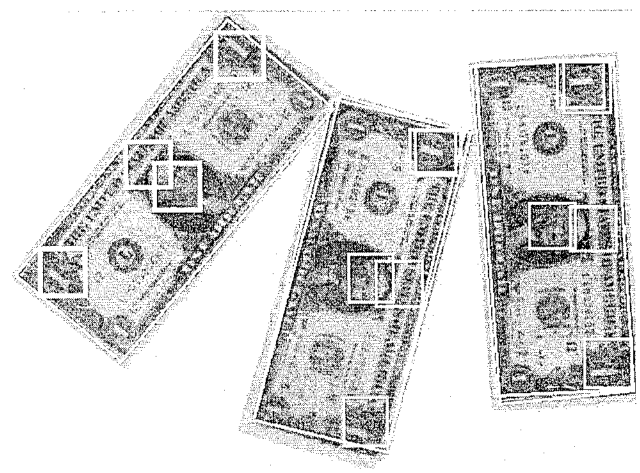
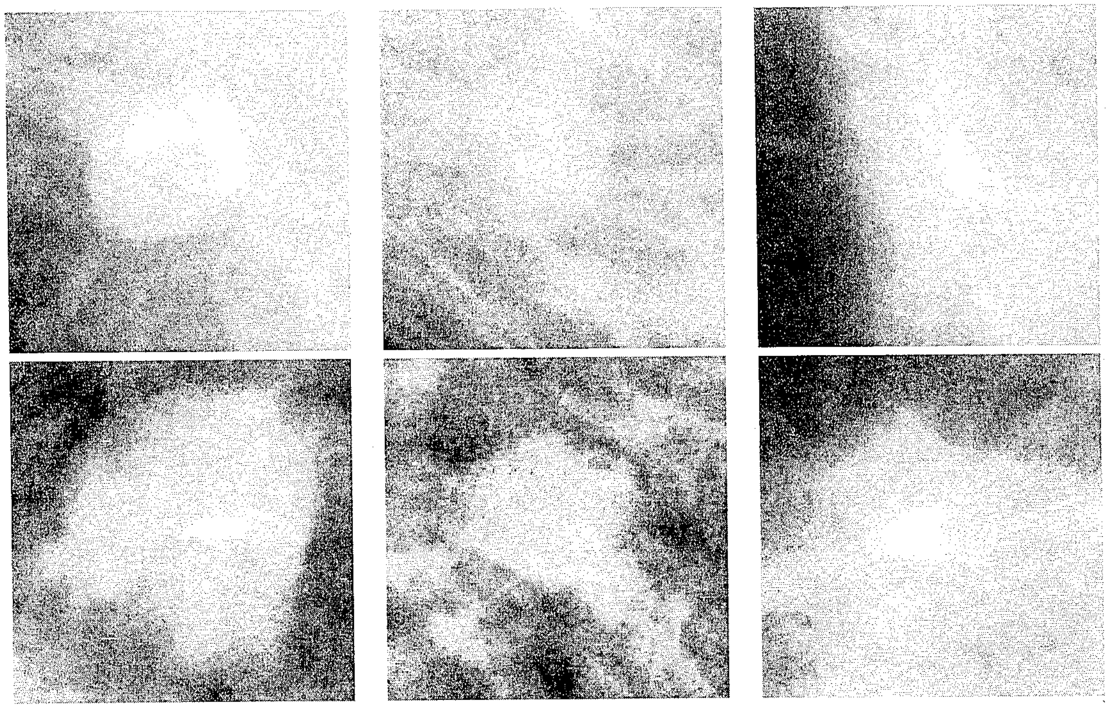


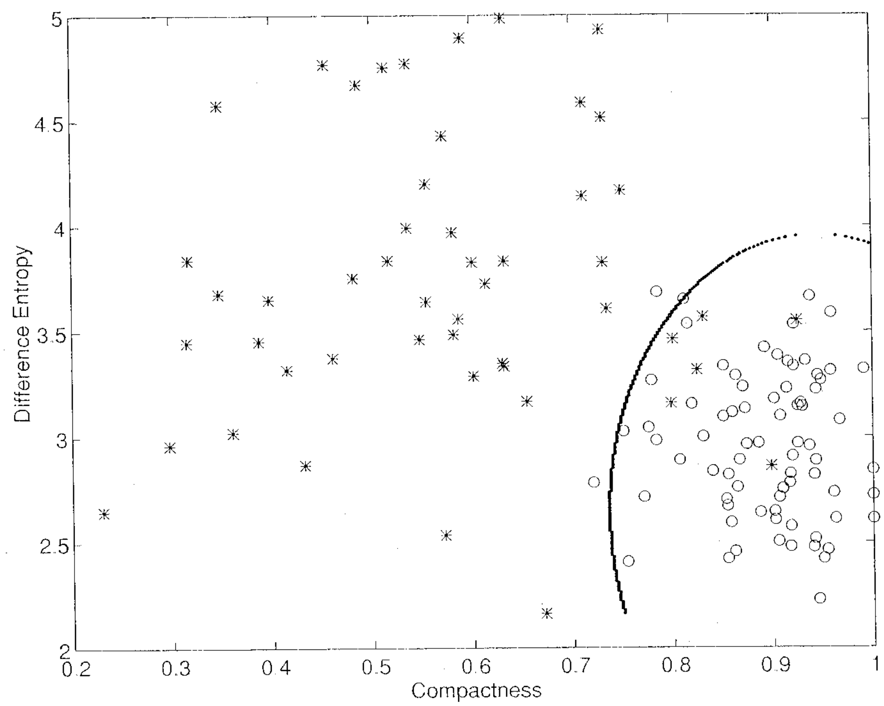
Fig. 12. Money patterns detected by hierarchical DBNN recognizer. The small boxes indicate the locations of the detected local currency texture features, and the large boxes show the locations of the money patterns.

After potential feature blocks are detected, the algorithm then uses a structural matching method, adopted from [36], to group the feature blocks that possibly form a currency pattern. The registered currency patterns at every possible location in the scanned image (translation invariant), with every possible rotation angle (rotation invariant), and with various background images (robust detection), can be successfully recognized. Wrinkled or torn currencies can also be recognized. Fig. 12 shows some money patterns that are detected by the recognition engine.

On a PowerPC, the software version of the hierarchical FNN recognition engine has a misrecognition rate of less than 0.1% (i.e., no currency pattern out of 1000 tests fail



(a)



(b)

Fig. 13. (a) The typical mass appearances in mammograms. (b) The classification results: -o- denotes true mass cases; -∗- denotes false mass cases. Compactness and difference entropy are the features used for classification. Note: Originally, a total 17 features selected by medical from experts were studied in our DBNN clustering, the 2-D plot shows the two most representative features which posses the greatest distinguishability.

to be detected) and the false detection rate is less than 0.001% (i.e., no noncurrency pattern out of 100 000 tests are falsely detected). As to the processing speeds, The 132 MHz PowerPC takes in 3–5 s to detect currencies from more than eight countries on a 50 DPI 8.5-in $\times$ 11-in scanned page.

4) Medical Applications of Experts-in-Class FNN's: FNN's have been successfully applied to several medical applications. The following paragraphs describe some of the examples.

a) Medical image quantification: The problem of image quantification is very different from image segmenta-

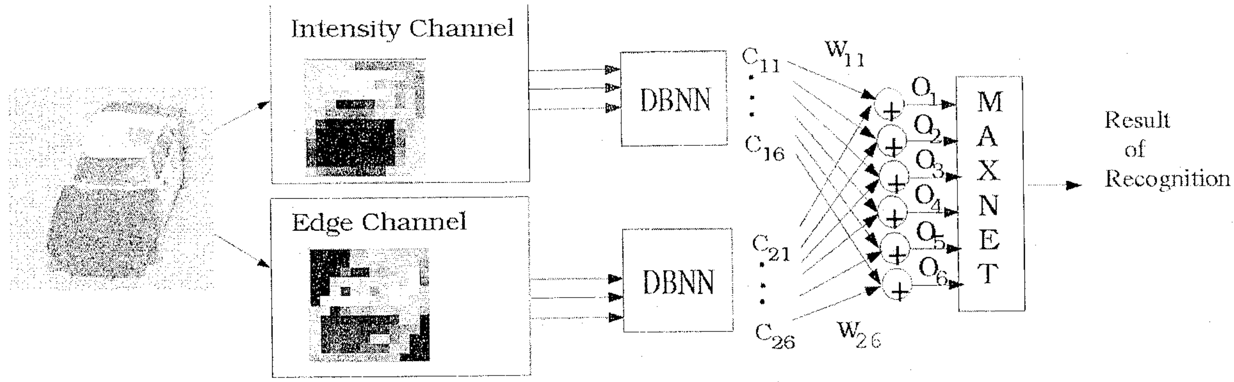


Fig. 14. A classes-in-expert system for car recognition. Given an original image, two experts are adopted to classify various car models using features from different frequency domains. One expert uses the low-frequency information, and the other uses the high-frequency information. The classification results from the two experts are combined by the classes-in-expert fusion scheme.

tion. Compared to image segmentation, which categorizes each image pixel to one of the many image classes (i.e., hard classification), image quantification assigns to pixels the probabilistic (fuzzy) memberships from multiple image classes (i.e., quantitative mapping functions). These memberships carry particular significance beyond the actual segmentation. One example is the determination of the activity involving the transmission of nerve impulses in the selected regions of interest (ROI) in the brain [32]. To assist doctors to analyze the high activity regions in the positron emission tomography (PET) image of the brain, Wang *et al.* [59], [60] assign the image pixels fuzzy membership values from different brain tissue classes. The memberships are created by examining the higher-resolution magnetic resonance (MR) image which is registered to the PET image beforehand.

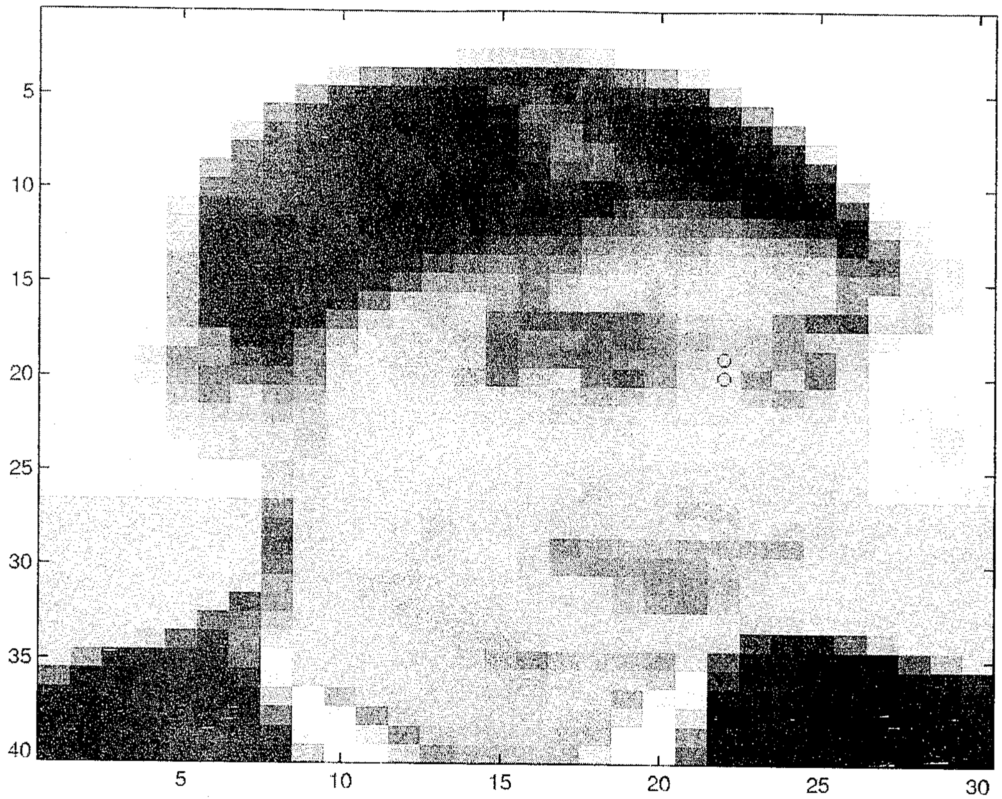
The fuzzy class memberships are realized by the class conditional likelihood functions in the experts-in-class network. A model fitting scheme is used to estimate the number and kernel of local clusters using information theoretic criteria. The class distribution functions are then obtained by learning generalized Gaussian mixtures where a fuzzy classification of the data is performed. Further classification of the data is treated as a hard Bayesian detection problem, where the decision boundaries between the classes are fine-tuned by the decision-based learning scheme. Successful implementation of this framework has been demonstrated in [58] and [59].

b) Computed-aided diagnosis (CAD): The aforementioned experts-in-class networks can be applied to the CAD for breast cancer detection. The objective is to detect masses (suspicious cancer regions) in digital mammography. Some typical mass cluster appearances on mammograms are displayed in Fig. 13. In clinic sites, masses are evaluated based on the location, density, size, shape, margins, and the presence of associated calcifications. Note that the knowledge database involving computer-assisted diagnosis has long recognized to be fairly complex. In fact, the complexity has little to do with "dimensionality" of the feature space. Therefore, the hierarchical DBNN proves to be a very effective tool for classification. For more details, see [34]. A two-class hierarchical DBNN is trained to distinguish the "true masses" from the "false masses" based on the features extracted from the suspected regions.

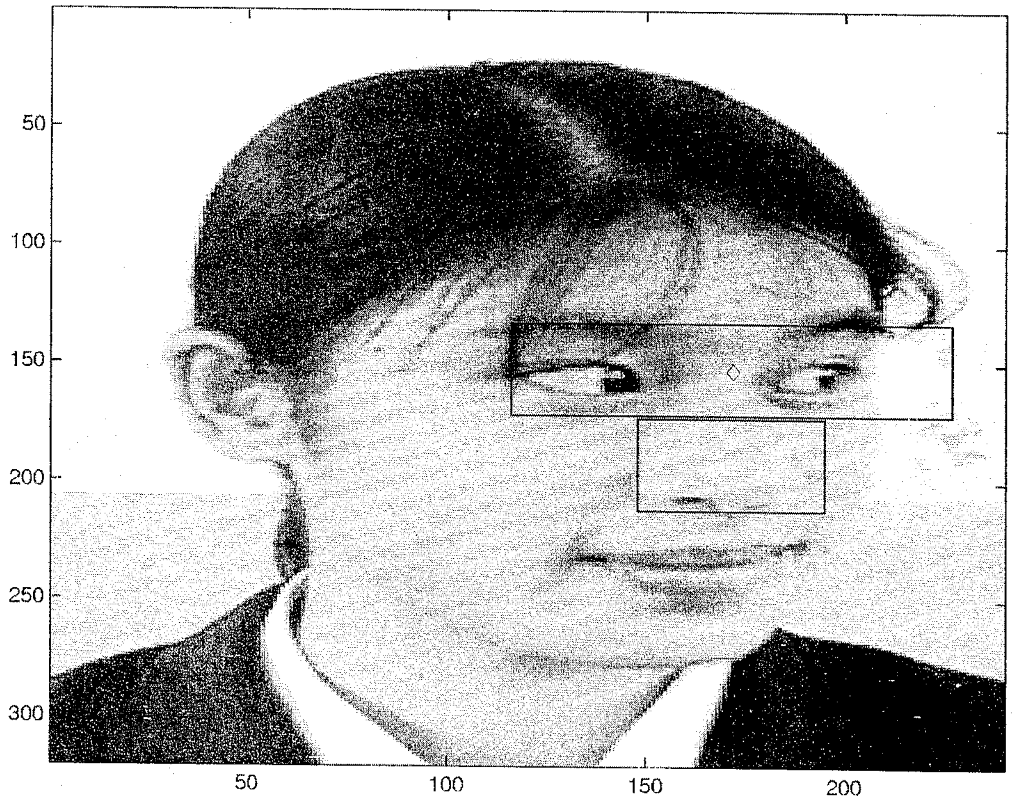
1) A total of 150 mammograms were selected from the mammographic database. Each mammogram contained at least one mass case of varying size and location. Among them, 50 of the mammograms contain the biopsy-proven masses (true masses). Note that one mammogram consists of two breast images taken from different viewing directions. In other words, 300 images are selected.
2) The training set was constructed from 50 true mass ROI's and 50 false mass ROI's. Note that there are more than one mass ROI's in one mammogram image.
3) The test set contained 46 randomly selected single-view mammograms: 23 normal cases and 23 containing biopsy-proven masses.
4) The feature vector contained two features: compactness and difference entropy. According to our investigation, these two features have the better separation (discrimination) between the true and false mass classes.

Fig. 13(b) shows the classification of two classes with our method. In our evaluation study, 6–15 suspected masses per mammogram were detected and recommended to physicians for further evaluation. The receiver operating characteristic (ROC) method is used to evaluate the detection performance. When the two distributions overlap on the decision axis, a cutoff point can be made at an arbitrary decision threshold. The corresponding true-positive fraction (TPF) versus false-positive fraction (FPF) for each threshold can be drawn on a plane, where sensitivity (TPF) and specificity (one minus FPF) are used to evaluate the system performance on the specified point of the ROC curve. The best operating point of the proposed classifier is at a sensitivity of 84% with a specificity of 82%. At this point, the classifier reaches the performance of 1.6 false positive findings per mammogram, which outperforms the conventional rate at 2.0 [58].

5) Three-Dimensional (3-D) Object Recognition via Classes-in-Expert FNN: The classes-in-expert hierarchical



(a)



(b)

Fig. 15. This figure shows a typical detected situation. (a) shows the downsampled images and the detected face positions (marked with circles) and (b) shows the images at the original resolution. The feature region (within boxes) and the final detected position (marked with ◇) are shown.

fusion network has demonstrated classification improvements on 3-D vehicle recognition problems [36]. This is illustrated in Fig. 14. The experiment used six car models from different view angles to create the training and testing database. Around 30 images (each with size 256 × 256 pixels) were taken for each car model

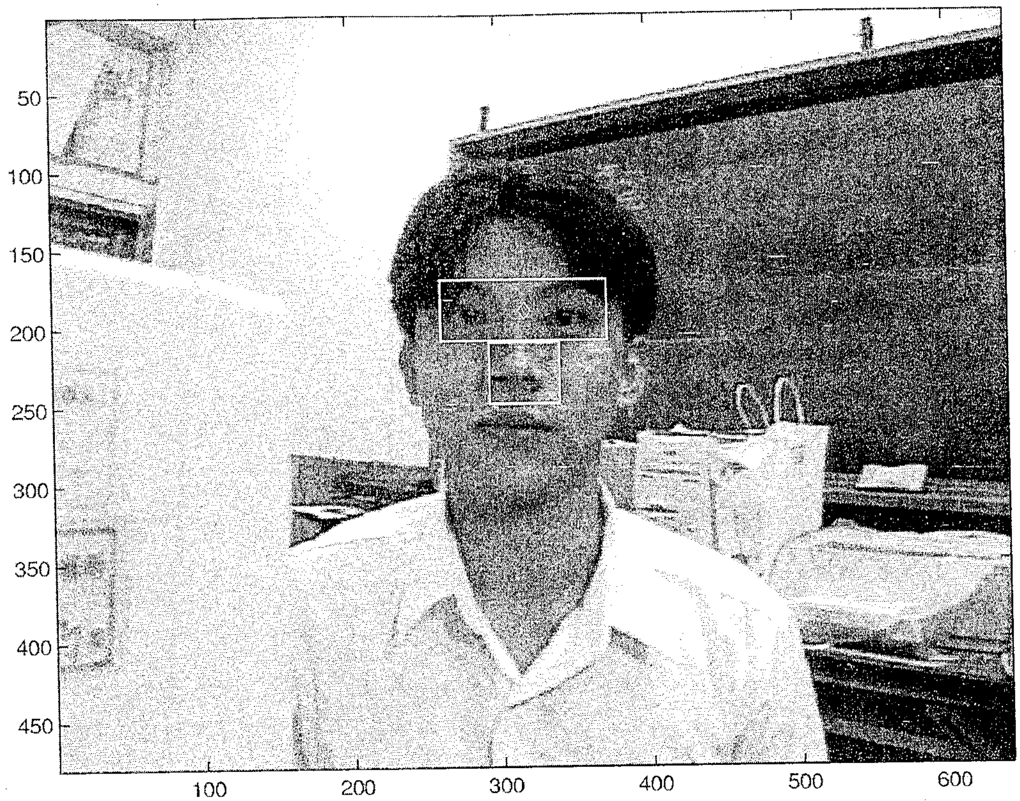


Fig. 16. This figure shows a typical result of the online face detection situation. The feature region is shown with boxes. And the final detected position is marked with $\diamond$.

from various viewing directions. There were in total 172 examples in the data set. Two classifier channels were built from two different feature extraction methods: one used intensity information and the other edge information. The fusion of two channels (with 94 and 85% recognition rate each), the recognition rate reached 100%. The fusion model was compared with a single network classifier. The input vectors of these two networks were formed by cascading the intensity vector with the edge vector. Therefore the input vector dimension becomes $144 \times 2 = 288$. The network is the RBF-typed DBNN. The experimental result shows a performance of around 96% recognition rate which is much inferior to (the perfect rate of) the fusion network.

C. *Applications of NEFCAR Hierarchical Classifier*

1) Face Detection and Localization: An online face localization system, based on the proposed neurofuzzy NEFCAR model, will be discussed. For face localization from any image sequence, an algorithm combining both the heuristic and neurofuzzy approaches may be adopted. The moving objects are found by thresholding the difference image between two consecutive images. Then the approximate ROI in the image that contains the (moving) person can be determined. Therefore, the amount of data required for processing can be greatly reduced. Then the neurofuzzy classifier is applied to locate the face in the ROI. We assume that there exists at most one face in an ROI. Therefore, if more than one face is detected, the position with the largest confidence measure is selected.

To save the storage space and the processing time, we first worked on the still images which are taken using a digital camera. The size of the images is 320×240. (One can simply regard these images as ROI's obtained from the motion information.) In the training and test procedure, images from 75 persons with three different view angles (frontal 45° to the right, 0°, 45° to the left) were used. To alleviate the harmful influences from the hair style, the ornaments, or the facial expressions, we adopted a normalized T-shape image block which covers the eyes and nose as the feature vector [see the image in Fig. 15]. The feature area was down-sampled by a factor of eight in order to lower the feature's sensitivity to alignments. This process appeared to tolerate uncertain variations well, such as different eye glasses wore by the same individual. Furthermore, the mean of the feature vector was removed and the variance renormalized to one. For the training patterns, we manually selected the center positions of the eyes (p_1 and p_2) by using the mouse. The position for a positive sample was obtained by averaging the center positions ($(p_1 + p_2)/2$). The pixels in the image outside a mask m_g centered at the positive sample were considered as the positions of negative patterns. The size of the mask m_g was 3×3 in the downsampled images. When evaluating the performance of the classifier, if the face was detected within the mask m_g, the detection would be considered to be correct. Otherwise, the detection would be considered as false. If the confidence measure was too low for the detection, it was deemed as undecided (rejected). The training accuracy of the still images (images with 45° to the right and 0°) was 100% correct. The test accuracy on the third set of images (45° to the left) was around 93.4% correct, 5.3% undetected, and 1.3% false. A typical detected still face is shown in Fig. 15. For an online test of the classifier, ten experiments were conducted and all the faces were detected. A typical result is shown in Fig. 16. The accuracy of the face position can be further improved if certain postprocessing, such as eye detection, is adopted. In an online application observation, if multiple image frames can be grabbed and processed by the NEFCAR, then more robust face localization may be resulted by fusing the decisions of individual frames.

V. CONCLUSION

This paper has studied the rich synergy between NN's and fuzzy logic systems. For example, it has demonstrated that several prominent NN's can be easily reformulated as fuzzy systems and vice versa. The study leads to a unifying framework to integrate fuzzy logic processing and NN's, which incorporate unsupervised (e.g., EM) learning and interclass supervised learning (e.g., decision-based learning) into a hierarchical FNN structure. Theoretically, the hierarchical FNN family adopts a probabilistic soft-decision training strategy and a hierarchical structure which reflects (expressive) knowledge of experts. Practically, the proposed FNN family represents a robust information processing system for classification and data fusion. It can yield performance advantages and can be useful to a broad spectrum of application domains. As demonstrated in the paper, several exemplifying applications have been built upon such a synergistic model. In order to materialize fully such advantages in terms of practical applications, the main challenge in our opinion lies in achieving a thorough understanding of the input feature space and a proper mapping of such space onto the expert/class modules according to the (unsupervised and supervised) learning strategies proposed.

REFERENCES

[1] S. Abe and R. Thawonmas, "A fuzzy classifier with ellipsoidal regions," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 358–368, Aug. 1997.

[2] M. A. Abidi and R. C. Gonzalez, *Data Fusion in Robotics and Machine Intelligence*. Boston, MA: Academic, 1992.

[3] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neuro-Computing: Algorithms, Architectures, and Applications*, F. Fogelman and J. Herault, Eds. London, U.K.: Springer-Verlag, 1991, pp. 227–236.

[4] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1042–1052, 1993.

[5] Y. Chan, S.-H. Lin, and S. Y. Kung, "Video indexing and retrieval," in *Multimedia Technology for Applications*, B. Sheu and M. Ismail, Eds. Piscataway, NJ: IEEE Press, 1998, pp. 253–281.

[6] Y. Chan, S. H. Lin, Y. P. Tan, and S. Y. Kung, "Video shot classification using human faces," in *Proc. IEEE Int. Conf. Image Processing*, Lausanne, Switzerland, 1996, pp. 843–846.

[7] I. J. Cox, J. Ghosn, and P. Yianilos, "Feature-based face recognition using mixture distance," NEC Res. Inst., Princeton, NJ, Tech. Rep. 95-09, 1995.

[8] G. Cybenko, "Approximation by of a sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, pp. 303–314, 1998.

[9] Y. Delignon, A. Marzouki, and W. Pieczynski, "Estimation of generalized mixtures and its application in image segmentation," *IEEE Trans. Image Processing*, vol. 6, pp. 1364–1375, Oct. 1997.

[10] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statistical Soc.*, *B*, vol. 39, pp. 1–38, 1976.

[11] D. Driankov, H. Hellendoorn, and M. Reinfrank, *An Introduction to Fuzzy Control*. Berlin, Germany: Springer-Verlag, 1993.

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

[13] S.-H. Lai and M. Fang, "Robust and automatic adjustment of display window width and center for MR images," in *Proc. SPIE Symp. Medical Imaging: Image Perception*, vol. 3340, San Diego, CA, Feb. 1998, pp. 105–116.

[14] ——, "A neural network based system for optimal medical image visualization with fast online adaptation capabilities," in *Proc. 32nd Annu. Conf. Information Sciences and Systems*, Princeton, NJ, Mar. 1998, pp. 715–720.

[15] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annu. Eugenics*, vol. 7, pt. II, pp. 179–188, 1936.

[16] K. Funahashi, "On the approximate realization of continuous mappings by Neural Networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[17] F. Goudail, E. Lange, T. Iwamoto, K. Kyuma, and N. Otsu, "Face recognition system using local autocorrelations and multiscale integration," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, Oct. 1996.

[18] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, pp. 4–29, Apr. 1984.

[19] R. J. Hathaway, "Another interpretation of the EM algorithm for mixture distributions," *Statist. Prob. Lett.*, vol. 4, pp. 53–56, 1986.

[20] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.

[21] J. N. Hwang and E. Lin, "Mixture of discriminative learning experts of constant sensitivity for automated cytology screening," in *Proc. 1997 IEEE Workshop Neural Networks for Signal Processing*, Amelia Island, FL, Sept. 1997, pp. 152–161.

[22] R. A. Jacobs, M. I. Jordan, and A. G. Barto, "Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks," *Cognitive Sci.*, vol. 15, pp. 219–250, 1991.

[23] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.

[24] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[25] M. I. Jordan and R. A. Jacobs, "Hierarchies of adaptive experts," in *Kass Advances in Neural Information Systems*, vol. 4. San Mateo, CA: Morgan Kaufmann, 1992, pp. 985–992.

[26] ——, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, pp. 181–214, 1994.

[27] B. H. Juang, S. Y. Kung, and C. A. Kamm, Eds., *Proc. IEEE Workshop Neural Networks for Signal Processing*, vols. I–VII. Piscataway, NJ: IEEE Press, 1991–1997.

[28] B. H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Trans. Signal Processing*, vol. 40, pp. 3043–3054, Dec. 1992.

[29] B. Kosko, "Fuzzy systems are universal approximators," in *Proc. IEEE Int. Conf. Fuzzy Systems*, San Diego, CA, Mar. 1992, pp. 1153–1162.

[30] S. Y. Kung, *Digital Neural Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[31] S. Y. Kung and J. S. Taur, "Decision-based neural networks with signal/image classification applications," *IEEE Trans. Neural Networks*, vol. 6, pp. 170–181, Jan. 1995.

[32] C. Lau, T. Adalı, and Y. Wang, "Co-registration of PET/MR brain images by multi-feature correlation matching," in *Proc. 15th Southern Biomedical Engineering Conf.*, Dayton, OH, Mar. 1996, pp. 301–304.

[33] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural network approach," *IEEE Trans. Neural Networks*, vol. 8, pp. 98–113, Jan. 1997.

[34] H. Li, Y. Wang, K.-J. R. Liu, S.-H. B. Lo, and M. T. Freedman, "Statistical model supported approach to radiographic mass detection; Part I—Improving lesion characterization by

morphological filtering and site segmentation; Part II—Decision making through minimax entropy modeling and modular neural networks," *IEEE Trans. Med. Imaging*, to be published.

5] C. T. Lin and C. S. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

5] S. H. Lin, "Biometric identification for network security and access control," Ph.D. dissertation, Dep. Elect. Eng., Princeton Univ., Princeton, NJ, 1996.

7] S.-H. Lin, Y. Chan, and S. Y. Kung, "A probabilistic decision-based neural network for location deformable objects and its applications to surveillance system and video browsing," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Atlanta, GA, 1996, pp. 3554–3557.

3] S. H. Lin, S. Y. Kung, and L. J. Lin, "Face recognition/detection by probabilistic decision-based neural networks," *IEEE Trans. Neural Networks (Special Issue on Artificial Neural Network and Pattern Recognition)*, vol. 8, pp. 114–132, Jan. 1997.

9] Y. Lin, "Object oriented scene segmentation from visual motion," Ph.D. dissertation, Dep. Elect. Eng., Princeton Univ., Princeton, NJ, 1998.

0] Y. Lin, Y.-K. Chen, and S. Y. Kung, "A principal component clustering approach to object-oriented motion segmentation and estimation," *J. VLSI Signal Processing Syst.*, vol. 17, pp. 163–188, Nov. 1997.

1] M. MacQueen, "Some methods for classification and analysis of multivariate observation," in *Proc. 5th Berkeley Symp. Mathematical Statistics and Probabilities*, vol. 1, L. M. LeCun and J. Neyman, Eds. Berkeley, CA: Univ. California Press, 1967, pp. 281–297.

2] P. Masson and W. Pieczynski, "SEM algorithm and unsupervised statistical segmentation of satellite images," *IEEE Trans. Geosci. Remote Sensing*, vol. 31, pp. 618–633, May 1993.

3] P. McCullagh and J. A. Nelder, *Generalized Linear Models*, 2nd ed. London, U.K.: Chapman & Hall, 1989.

4] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computing*, vol. 1, pp. 281–294.

5] D. Nauck, U. Nauck, and R. Kruse, "Generating classification rules with the neuro-fuzzy system NEFCLASS," in *Proc. Biennial Conf. North American Fuzzy Information Processing Soc.*, Berkeley, CA, 1996.

5] D. Nauck and F. Klawonn, "Neuro-fuzzy classification initialized by fuzzy clustering," in *Proc. 4th Europ. Congr. Intelligent Techniques and Soft Computing*, Aachen, Germany, 1996.

7] S. J. Nowlan and G. E. Hinton, "Evaluation of adaptive mixtures of competing experts," in *Advance in Neural Information Processing Systems*, vol. 3. San Mateo, CA; Morgan Kaufmann, 1991, pp. 774–780.

8] K. Popat and R. W. Picard, "Cluster-based probability model and its applications to image and texture processing," *IEEE Trans. Image Processing*, vol. 6, pp. 268–284, Feb. 1997.

9] Y. Wang, A. Reibman, F. Juang, T. Chen, and S.Y. Kung, Eds., *Proc. IEEE Workshop Multimedia Signal Processing*, Princeton, NJ, 1997.

0] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, pp. 1481–1497, Sept. 1990.

1] J. C. Rajapakse, J. N. Giedd, and J. L. Rapoport, "Statistical approach to segmentation of single-channel cerebral MR images," *IEEE Trans. Med. Imaging*, vol. 16, pp. 176–185, Apr. 1997.

2] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing (PDP): Exploration in the Microstructure of Cognition*, vol. I. Cambridge, MA: MIT Press, 1986.

3] J. S. Taur and S. Y. Kung, "Fuzzy decision neural networks and applications to data fusion," in *Proc. IEEE Workshop Neural Networks for Signal Processing*, vol. III, Linthicum Heights, MD, Sept. 1993, pp. 171–180.

4] J. S. Taur and C. W. Tao, "Texture classification using a fuzzy texture spectrum and neural network," *J. Electron. Imaging*, vol. 7, no. 1, pp. 29–35, Jan. 1998.

5] ——, "Face detection using neuro-fuzzy classifiers," in *Proc. Int. Symp. Multimedia Information Processing*, Taiwan, Dec. 1998, pp. 309–314.

6] D. Valentin, H. Abdi, A. J. O'tool, and G. W. Cottrell, "Connectionist models of face processing: A survey," *Pattern Recognition*, vol. 27, pp. 1209–1230, 1994.

7] L. X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[58] Y. Wang, S. H. Lin, H. Li, and S. Y. Kung, "Data mapping by probabilistic modular networks and information theoretic criteria," *IEEE Trans. Signal Processing*, vol. 46, pp. 3378–3397, Dec. 1998.

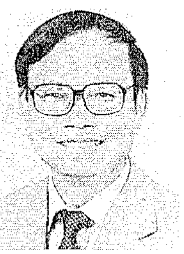
[59] Y. Wang, T. Adalı, S. Y. Kung, and Z. Szabo, "Quantification and segmentation of brain tissue from MR images: A probabilistic neural network approach," *IEEE Trans. Image Processing (Special Issue on Applications of Neural Networks to Image Processing)*, vol. 7, pp. 1165–1181, Aug. 1998.

[60] Y. Wang, T. Adalı, C. Lau, and S. Y. Kung, "Quantitative analysis of MR brain image sequences by adaptive self-organizing mixtures," *J. VLSI Signal Processing Syst. (Special Issue on Neural Networks for Biomedical Image Processing)*, vol. 18, no. 3, pp. 219–2490, Apr. 1998.

[61] S. R. Waterhouse, D. MacKay, and A. J. Robinson, "Bayesian methods for mixtures of experts," *Proc. Advances in Neural Information Processing 8*, Denver, CO, Nov. 1995, pp. 351–357.

[62] M. M. Yeung, B. L. Yeo, W. Wolf, and B. Liu, "Video browsing using clustering and scene transitions on compressed sequences," in *Proc. SPIE Multimedia Computing and Networking*, vol. 2417, Feb. 1995, pp. 399–413.

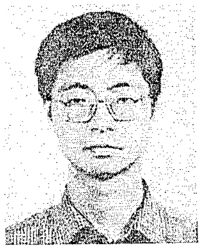
[63] A. L. Yuille, P. Stolorz, and J. Utans, "Statistical physics, mixtures of distributions, and the EM algorithm," *Neural Computation*, vol. 6, pp. 334–340, 1994.



Sun-Yuan Kung (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA.

In 1974, he was an Associate Engineer with Amdahl Corporation, Sunnyvale, CA. From 1977 to 1987, he was a Professor of Electrical Engineering Systems at the University of Southern California, Los Angeles. In 1984, he was a Visiting Professor at Stanford University and Delft University of Technology. In 1994, he was a Toshiba Chair Professor at Waseda University, Tokyo, Japan. Since 1987, he has been a Professor of Electrical Engineering at Princeton University, Princeton, NJ. His research interests include spectrum estimations, digital signal/image processing, very large scale integration (VLSI) array processors, neural networks, and multimedia signal processing. Since 1990, he has been Editor-in-Chief of the *Journal of VLSI Signal Processing*. He has authored more than 300 technical publications and three books, most recently *Principal Component Neural Networks* (New York: Wiley, 1996). He has edited numerous reference and proceedings books, most recently *Application-Specific Array Processors* (New York: IEEE Computer Society Press, 1991).

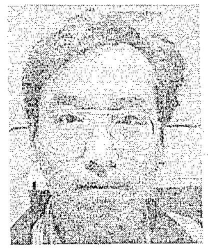
Dr. Kung was the Keynote Speaker for the First International Conference on Systolic Arrays, Oxford, U.K., in 1986. He has been an Associate Editor of several IEEE TRANSACTIONS. He was the first Associate Editor in the areas of VLSI (1984) and neural networks (1991) of IEEE TRANSACTIONS ON SIGNAL PROCESSING. He was a member of the Administration Committee of the IEEE Signal Processing Society (SPS) from 1989 to 1991. He was a founding member of IEEE-SPS Technical Committees on VLSI Signal Processing and on Neural Networks. He is currently a member of the Technical Committee on Multimedia Signal Processing. He was a founding member and General Chairman of various IEEE conferences, including IEEE Workshops on VLSI Signal Processing in 1982 and 1986, the International Conference on Application Specific Array Processors in 1990 and 1991, IEEE Workshops on Neural Networks and Signal Processing in 1991 and 1992, and the first IEEE Workshop on Multimedia Signal Processing in 1997. He became an IEEE-SPS Distinguished Lecturer in 1994. He received the 1996 IEEE SPS's Best Paper Award for his publication on principal component neural networks and the 1992 IEEE SPS Technical Achievement Award for his contributions on "parallel processing and neural network algorithms for signal processing."



Jinshiuh Taur (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1987 and 1989, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, in 1993.

His professional experience includes being a Member of the Technical Staff at Siemens Corporate Research, Inc. Since 1994, he has been an Associate Professor at the National Chung-Hsing University, Taiwan. His primary research interests include neural networks, pattern recognition, computer vision, and fuzzy logic systems.

Dr. Taur received the 1996 IEEE Signal Processing Society's Best Paper Award.



Shang-Hung Lin (Member, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1991. He received the M.S. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 1994 and 1996, respectively.

From 1996 to 1998, he was a Senior Research Member of EPSON Palo Alto Laboratory, Palo Alto, CA. He is currently working with IC Media Corporation, Santa Clara, CA. His primary research interests include neural networks, biometric identification, pattern recognition, computer vision, and image processing.