Papers

A Fuzzy Neural Network and its Application to Pattern Recognition

Hon Keung Kwan, *Senior Member, IEEE* and Yaling Cai, *Student Member, IEEE*

***Abstract*—In this paper, we define four types of fuzzy neurons and propose the structure of a four-layer feedforward fuzzy neural network (FNN) and its associated learning algorithm. The proposed four-layer FNN performs well when used to recognize shifted and distorted training patterns. When an input pattern is provided, the network first fuzzifies this pattern and then computes the similarities of this pattern to all of the learned patterns. The network then reaches a conclusion by selecting the learned pattern with the highest similarity and gives a nonfuzzy output. The 26 English alphabets and the 10 Arabic numerals, each represented by 16×16 pixels, were used as original training patterns. In the simulation experiments, the original 36 exemplar patterns were shifted in eight directions by 1 pixel (6.25% to 8.84%) and 2 pixels (12.5% to 17.68%). After the FNN has been trained by the 36 exemplar patterns, the FNN can recall all of the learned patterns with 100% recognition rate. It can also recognize patterns shifted by 1 pixel in eight directions with 100% recognition rate and patterns shifted by 2 pixels in eight directions with an average recognition rate of 92.01%. After the FNN has been trained by the 36 exemplar patterns and 72 shifted patterns, it can recognize patterns shifted by 1 pixel with 100% recognition rate and patterns shifted by 2 pixels with an average recognition rate of 98.61%. We have also tested the FNN with 10 kinds of distorted patterns for each of the 36 exemplars. The FNN can recognize all of the distorted patterns with 100% recognition rate. The proposed FNN can also be adapted for applications in some other pattern recognition problems.**

I. INTRODUCTION

A NEURAL NETWORK (NN) has a massively parallel structure which is composed of many processing elements connected to each other through weights [1]–[3]. Neural networks (NN's) are built after biological neural systems. A NN stores patterns with distributed coding and is a trainable nonlinear dynamic system. A NN has a faster response and a higher performance than those of a sequential digital computer in emulating the capabilities of the human brain. Recently, NN's have been used in pattern recognition problems, especially where input patterns are shifted in position and scale-changed. Fukushima *et al.* [4], [5] have presented the Neocognitron, which is insensitive to translation and deformation of input patterns, and used it to recognize hand-printed characters. However, the Neocognitron is complex and needs many cells. Carpenter and Grossberg [6] have proposed a self-organizing system which can classify patterns by adaptive resonance theory. However, a lot of internal exemplars including noise patterns are formed in the network. Martin and Pittman [7] have used a backpropagation (BP) learning network to recognize hand-printed letters and digits. Guyon *et al.* [8] have designed a system for on-line recognition of handwritten characters for a touch terminal using a time-delay neural network and the BP algorithm. Fukumi *et al.* [9] have proposed a neural pattern recognition system trained by the BP algorithm which can be used to recognize rotated patterns. In [7]–[9], the major problem lies in the lengthy training time of the BP algorithm which does not exist in the proposed fuzzy neural network. Perantonis and Lisboa [10] have constructed a pattern recognition system which is invariant to the translation, rotation, and scale of an input pattern by high-order neural networks. However, the number of weights in such a network increases greatly with the order of the network.

On the other hand, fuzzy logic [11]–[15] is a powerful tool for modeling human thinking and perception. Instead of bivalent propositions, fuzzy systems reason with multivalued sets. Fuzzy systems store rules and estimate sampled functions from linguistic input to linguistic output. It is believed that the effectiveness of the human brain is not only from precise cognition, but also from fuzzy concept, fuzzy judgment and fuzzy reasoning. Dealing with uncertainty is a common problem in pattern recognition. Fuzzy set theory has proved itself to be of significant importance in pattern recognition problems [12]–[19]. Fuzzy methods are particularly useful when it is not reasonable to assume class density functions and statistical independence of features.

Some work have been carried out on fuzzy neural systems for pattern recognition. Kosko [14] has proposed a Fuzzy Associate Memory (FAM) which defined mappings between fuzzy sets. FAM used fuzzy matrices instead of fuzzy neurons to represent fuzzy associations. Yamakawa and Tomoda [17] have described a simple fuzzy neuron model and used in a neural network for application in character recognition problems. However, they did not describe the specific learning algorithm for this network. Takagi *et al.* [18] have constructed a structured neural network using the structure of fuzzy inference rules. This structured NN has better performance than ordinary NN's when used in pattern recognition problems. However, it is complicated to train this NN as it is composed of

Manuscript received July 2, 1992; revised February 14, 1994; accepted July 5, 1993. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada

The authors are with the Department of Electrical Engineering, University of Windsor, Windsor, Ontario, Canada N9B 3P4.

IEEE Log Number 9401794.

many small NN's. Machado and Rocha [20] have constructed a fuzzy connectionist expert system using combinatorial neural model (CNM). The CNM uses fuzzy numbers, fuzzy-AND neurons and fuzzy-OR neurons to classify patterns. It uses Hebb's rule to train the network. The attempt of building fuzzy neural systems for pattern recognition has achieved some successes. However, the existing systems are complex and there is not yet a simple fuzzy neural structure which can effectively deal with pattern recognition problems.

The objective of this research is to combine the features of fuzzy systems (with a ability to process fuzzy information using fuzzy algorithms) and the features of neural networks (with a learning ability and a high-speed parallel structure) to form a fuzzy neural network which can learn from environments. In this paper, we define a fuzzy neuron (FN), introduce four types of fuzzy neurons (FN's), and use them to construct a four-layer feedforward fuzzy neural network which can be applied to pattern recognition. We also propose a self-organizing learning algorithm for the four-layer feedforward FNN. The structure of this FNN is simple and its learning and recall speeds are fast. The third and fourth layers of the FNN are self-organized during learning.

In the next section, the definition of a FN and four types of FN's are given. In Section III, we use different types of FN's in different layers to construct a four-layer feedforward FNN for recognizing shifted or distorted training patterns. A self-organizing learning algorithm for the four-layer feedforward FNN is presented in Section IV. In Section V, we give an analysis of this FNN. In Section VI, simulation results are given and the proposed feedforward FNN is compared with the Hamming network, which is composed of nonfuzzy neurons, in terms of recognition rates. Section VII gives conclusions on the proposed FNN when used in pattern recognition.

II. Fuzzy Neurons

A typical nonfuzzy neuron has N weighted inputs and one output. The neuron sums these inputs x_i (for $i = 1$ to N) through the corresponding weights w_i ($i = 1$ to N) and transfers the result to a nonlinear activation function $f[\,]$. The output of such a can neuron be expressed as:

$$y = f\left[\sum_{i=1}^{N} w_i x_i - T\right] \tag{1}$$

where T is the internal threshold of the neuron.

A. Definition of Fuzzy Neuron

A FN has N weighted inputs, x_i for $i = 1$ to N, with w_i ($i = 1$ to N) as the weights, and M outputs, y_j for $j = 1$ to M. All the inputs and weights are real values and the outputs are real values in interval [0, 1]. Each output could be associated with the membership value of a fuzzy concept, i.e., it expresses to what degree the pattern with the inputs $\{x_1, x_2, \ldots, x_N\}$ belongs to a fuzzy set. Moreover, we have:

$$z = h[w_1x_1, w_2x_2, \ldots, w_Nx_N] \tag{2}$$

$$s = f[z - T] \tag{3}$$

$$y_j = g_j[s] \text{ for } j = 1 \text{ to } M \tag{4}$$

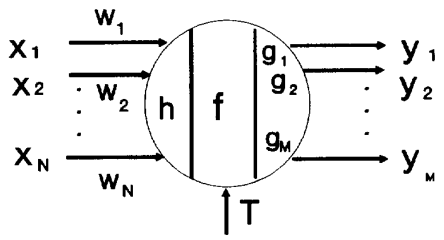


Fig. 1. A fuzzy neuron.

where z is the net input of the FN; $h[\,]$ is the aggregation function; s is the state of the FN; $f[\,]$ is the activation function; T is the activating threshold; and $\{g_j[\,], j = 1, 2, \ldots, M\}$ are the M output functions of the FN which represent the membership functions of the input pattern $\{x_1, x_2, \ldots, x_N\}$ in all the M fuzzy sets. Consequently, FN's can express and process fuzzy information.

In general, the weights, the activating threshold, and the output functions, which describe the interactions among FN's, could be adjusted during learning procedure. So FN's are adaptive and a FNN which is consisted of FN's can learn from environments. The aggregation function and the activation function are the intrinsic features of a FN. If different functions of $h[\,]$ and $f[\,]$ are used in different FN's, their properties will be different. Many types of FN's can be defined by changing functions $h[\,]$ and $f[\,]$. In a FN, the weights could be weight functions, and the threshold could be threshold function. Fig. 1 illustrates a FN. In the following, four types of FN's are defined.

B. Input-FN

If a FN is used in the input layer of a FNN and it has only one input x such that

$$z = x \tag{5}$$

then this FN is called an INPUT-FN.

C. Maximum-FN (Max-FN)

If a maximum function is used as the aggregation function of a FN such that

$$z = \max_{i=1}^{N}(w_i x_i) \tag{6}$$

then this FN is called a MAXIMUM-FN or MAX-FN.

D. Minimum-FN (Min-FN)

If a minimum function is used as the aggregation function of a FN such that

$$z = \min_{i=1}^{N}(w_i x_i) \tag{7}$$

then this FN is called a MINIMUM-FN or MIN-FN.

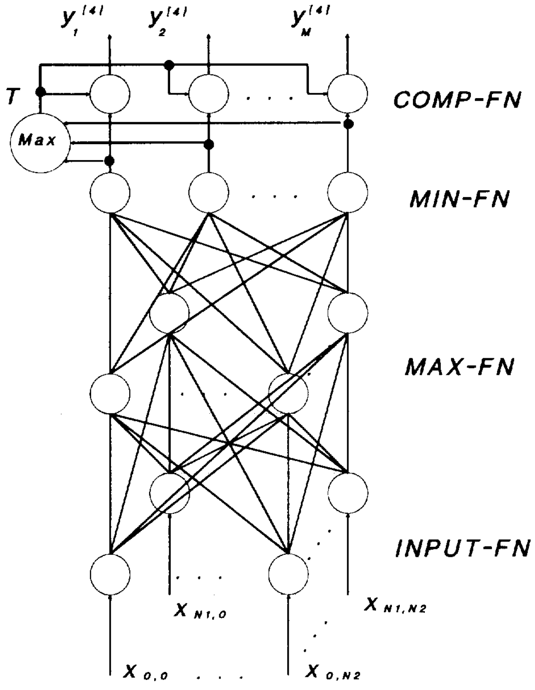


Fig. 2. Four-layer feedforward FNN.

E. Competitive-FN (Comp-FN)

If a FN has a variable threshold T and only one output such that

$$y = g[s - T] = \begin{bmatrix} 0 & \text{if } s < T \\ 1 & \text{if } s \geq T \end{bmatrix} \tag{8}$$

$$T = t[c_1, c_2, \ldots, c_K] \tag{9}$$

where s is the state of the FN; $t[\;]$ is the threshold function; and c_k ($k = 1$ to K) are competitive variables of the FN. This FN is called a COMPETITIVE-FN or COMP-FN.

III. Structure of The FNN

The proposed FNN is a four-layer feedforward FNN as shown in Fig. 2. The first layer is the input layer which accepts patterns into the network. We use INPUT-FN's in this layer. Each INPUT-FN in this layer corresponds to one pixel of an input pattern. The INPUT-FN's are displayed and indexed in two-dimension and the number of FN's in this layer is equal to the the total number of pixels of an input pattern. Assuming each input pattern has $N_1 \times N_2$ pixels, then the first layer has $N_1 \times N_2$ INPUT-FN's. The algorithm of the (i, j)th INPUT-FN in the first layer is:

$$s_{ij}^{[1]} = z_{ij}^{[1]} = x_{ij}, \text{ for } i = 1 \text{ to } N_1, j = 1 \text{ to } N_2 \tag{10}$$

$$y_{ij}^{[1]} = s_{ij}^{[1]}/P_{v\max}, \text{ for } i = 1 \text{ to } N_1, j = 1 \text{ to } N_2 \tag{11}$$

where x_{ij} is the (i, j)th pixel value of an input pattern ($x_{ij} \geq 0$) and $P_{v\max}$ is the maximum pixel value among all input patterns.

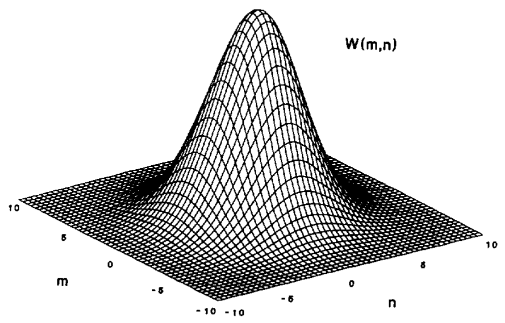


Fig. 3. Fuzzification function ($\beta = 0.3$).

The second layer is also displayed in two-dimension and consisted of $N_1 \times N_2$ MAX-FN's. The purpose of this layer is to fuzzify input patterns through a weight function $w[m, n]$. The state of the (p, q)th MAX-FN in this layer is:

$$s_{pq}^{[2]} = \max_{i=1}^{N_1}\left(\max_{j=1}^{N_2}\left(w[p - i, q - j]y_{ij}^{[1]}\right)\right)$$
$$\text{for } p = 1 \text{ to } N_1, q = 1 \text{ to } N_2 \tag{12}$$

where $w[p - i, q - j]$ is the weight connecting the (i, j)th INPUT-FN in the first layer to the (p, q)th MAX-FN in the second layer which is defined by:

$$w[m, n] = \exp(-\beta^2(m^2 + n^2))$$
$$\text{for } m = -(N_1 - 1) \text{ to } (N_1 - 1),$$
$$n = -(N_2 - 1) \text{ to } (N_2 - 1) \tag{13}$$

A plot of (13) for $\beta = 0.3$ is shown in Fig. 3. By using this weight function, each FN in the second layer is just like a lens so that each FN focuses on one pixel of an input pattern but it also can see the surrounding pixels. How many pixels a FN can actually see is determined by the value of β, which is to be decided by the learning algorithm. We also call $w[m, n]$ the fuzzification function.

Each MAX-FN in this layer has M different outputs (M is the number of FN's in the third layer), one for each FN in the third layer. The outputs of the (p, q)th MAX-FN in this layer are:

$$y_{pqm}^{[2]} = g_{pqm}[s_{pq}^{[2]}]$$
$$\text{for } p = 1 \text{ to } N_1, q = 1 \text{ to } N_2, m = 1 \text{ to } M \tag{14}$$

where $y_{pqm}^{[2]}$ is the mth output of the (p, q)th MAX-FN which is to be connected to the mth MIN-FN in the third layer. The output function $g_{pqm}[s_{pq}^{[2]}]$ is to be determined by the learning algorithm. For simplicity, we have chosen isosceles triangles with heights equal to 1 and base lengths equal to α (shown in Fig. 4) as the output functions of the MAX-FN's in the second layer. Hence

$$y_{pqm}^{[2]} = g_{pqm}[s_{pq}^{[2]}]$$
$$= \begin{bmatrix} 1 - 2|s_{pq}^{[2]} - \Theta_{pqm}|/\alpha & \text{if } \alpha/2 \geq |s_{pq}^{[2]} - \Theta_{pqm}| \geq 0 \\ 0 & \text{if otherwise} \end{bmatrix}$$
$$\text{for } \alpha \geq 0, p = 1 \text{ to } N_1, q = 1 \text{ to } N_2, m = 1 \text{ to } M \tag{15}$$

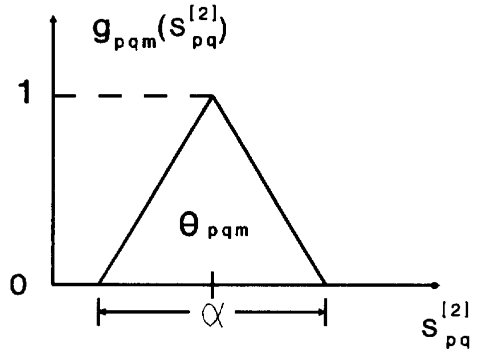


Fig. 4. Output function of a MAX-FN in the second layer.

where Θ_{pqm} is the central point of the base of function $g_{pqm}[s^{[2]}_{pq}]$. What have to be determined by the learning algorithm are the corresponding α and Θ_{pqm} for every set of p, q and m.

We use MIN-FN's in the third layer. Each MIN-FN in the third layer represents one learned pattern. Hence, the number of MIN-FN's in the third layer, M, could be determined only after the learning procedure is finished. The output of the mth MIN-FN in the third layer is:

$$y^{[3]}_m = s^{[3]}_m = \min_{p=1}^{N_1}(\min_{q=1}^{N_2}(y^{[2]}_{pqm})) \text{ for } m = 1 \text{ to } M \quad (16)$$

where $s^{[3]}_m$ represents the state of the mth MIN-FN in the third layer.

The fourth layer is the output layer. We use COMP-FN's in this layer, one for each of the M learned patterns, to provide nonfuzzy outputs. If an input pattern is most similar to the mth learned pattern, then the output of the mth COMP-FN in the fourth layer is 1 while other outputs are 0. The number of COMP-FN's in the output layer is equal to M. The algorithm of the mth COMP-FN in the fourth layer is:

$$s^{[4]}_m = z^{[4]}_m = y^{[3]}_m \qquad \text{for } m = 1 \text{ to } M \quad (17)$$

$$y^{[4]}_m = g[s^{[4]}_m - T] = \begin{bmatrix} 0 & \text{if } s^{[4]}_m < T \\ 1 & \text{if } s^{[4]}_m = T \end{bmatrix} \text{ for } m = 1 \text{ to } M \quad (18)$$

$$T = \max_{m=1}^{M}(y^{[3]}_m) \text{ for } m = 1 \text{ to } M \quad (19)$$

where T is the activation threshold of all the COMP-FN's in the fourth layer.

IV. Self-Organizing Learning Algorithm of the FNN

The following parameters must be determined by the learning procedure: The parameters of the output functions of the MAX-FN's in the second layer, α and Θ_{pqm} (for each set of p, q and m); the parameter of the fuzzification function, β; and the number of FN's in each of the third and fourth layers, M. We define T_f as the fault tolerance of the FNN ($0 \le T_f \le 1$) and K as the total number of training patterns (for $k = 1$ to K). The steps of the learning algorithm are:

Step 1. Create $N_1 \times N_2$ INPUT-FN's in the first layer and $N_1 \times N_2$ MAX-FN's in the second layer. Choose a value for α ($\alpha \ge 0$) and a value for β (See Section VI for appropriate choices).

Step 2. Set $M = 0$ and $k = 1$.

Step 3. Set $M = M + 1$. Create the Mth MIN-FN in the third layer and the Mth COMP-FN in the fourth layer. Set:

$$\Theta_{pqM} = s^{[2]}_{pqM} = \max_{i=1}^{N_1}(\max_{j=1}^{N_2}(w[p-i, q-j]x_{ijk}))$$
$$\text{for } p = 1 \text{ to } N_1, q = 1 \text{ to } N_2. \quad (20)$$

where Θ_{pqM} is the central point of the Mth output function of the (p, q)th MAX-FN in the second layer. $\boldsymbol{X}_k = \{x_{ijk}\}$ is the kth training pattern.

Step 3. Set $k = k + 1$. If $k > K$, then the learning procedure is finished. Otherwise, input the kth training pattern to the network and compute the output of the current FNN (with M FN's in the third and fourth layers). Set:

$$\sigma = 1 - \max_{j=1}^{M}(y^{[3]}_{jk}) \quad (21)$$

where $y^{[3]}_{jk}$ is the output of the jth MIN-FN in the third layer for the kth training pattern $\boldsymbol{X}_k$. If $\sigma \le T_f$, go to Step 4. If $\sigma > T_f$, go to Step 3.

V. Analysis of the FNN

In Sections III and IV, we have developed the structure and the learning algorithm of a four-layer feedforward fuzzy neural network for pattern recognition. Like neural networks, this FNN is a parallel system which processes all the pixels of an input pattern simultaneously. The proposed FNN is adaptively organized or constructed during the learning procedure. The proposed FNN is consisted of four types of fuzzy neurons, which can express and process fuzzy information and uses fuzzy algorithm to solve pattern recognition problems.

The first layer of the network accepts the data of an input pattern into the network. The INPUT-FN's in this layer transform the pixel values of an input pattern into normalized values within interval [0, 1].

The second layer of the network fuzzifies the input pattern. Each MAX-FN in the second layer is connected to all the INPUT-FN's of the first layer by the weight function $w[m, n]$ and takes the maximum value of all the weighted inputs as its state. The result of using MAX-FN's and using fuzzification weights is that one dark pixel in the input pattern will affect the states of several FN's in the second layer. Consequently, the dark pixels of the input pattern are fuzzified by the second layer of the FNN. The degree of an input pattern being fuzzified by the second layer depends on β. The smaller the value of β is, the more will be the numbers of FN's in the second layer affected by a dark pixel of an input pattern. This means that β controls the extent of fuzzification. If β is too small, the FNN can not separate some distinct training patterns.

If β is too large, the FNN may lose its ability to recognize some shifted or distorted patterns. β should be so chosen such that all the distinct training patterns can be separated by the FNN and the FNN has an acceptable recognition rate.

The mth output of the (p, q)th MAX-FN in the second layer, $y_{pqm}^{[2]}$, expresses the fuzzy concept with regard to the extent in which the pixel values around the (p, q)th pixel of the input pattern are similar to the pixel values around the (p, q)th pixel of the mth learned pattern. The output function $g_{pqm}[s_{pq}^{[2]}]$ is in fact a membership function of this fuzzy set and it should contain information of the pixel values around the (p, q)th pixel of the mth learned pattern. This is why we use Θ_{pqm} to store such information in the Step 3 of the learning algorithm. Consequently, the FNN can recall all of the learned patterns.

The FN's in the third layer give the similarities of the input pattern to all of the learned patterns. As we use MIN-FN's in the third layer, the similarity of the input pattern $\boldsymbol{X} = \{x_{ij}\}$ to the mth learned pattern is computed by the FNN as the output of the mth MIN-FN in the third layer:

$$y_m^{[3]} = \begin{cases} \min\limits_{p,q}(1 - \frac{2}{\alpha}|s_{pq}^{[2]} - \Theta_{pqm}|) & \text{if } \max\limits_{p,q}(|s_{pq}^{[2]} - \Theta_{pqm}|) \le \frac{\alpha}{2} \\ 0 & \text{if otherwise} \end{cases} \quad \text{for } m = 1 \text{ to } M \tag{22}$$

where $s_{pq}^{[2]}$ is the state of the (p, q)th MAX-FN in the second layer when input pattern is $\boldsymbol{X}$. We can see from (22) that α is a scope parameter and the value of α affects the computation of similarities. In order to cover all the possible pixel values, an appropriate choice of α is needed (see Section VI). When an input pattern $\boldsymbol{X}$ is one of the learned patterns, there will be one similarity $y_m^{[3]}$ (where $1 \le m \le M$) which equals to 1. When the input pattern is not any of the learned patterns, all of the M similarities are less than 1.

The output layer of the FNN is used to do defuzzification and give nonfuzzy outputs. It chooses the maximum similarity as the activation threshold of all the COMP-FN's in the fourth layer. If $y_m^{[3]}$ is the maximum among all the outputs of the FN's in the third layer, then the output of the mth COMP-FN in the fourth layer is 1 and the outputs of the other COMP-FN's in the fourth layer are 0. The recognition procedure of the proposed FNN is finished in four steps: input data (layer 1); fuzzification (layer 2); fuzzy deduction (layer 3); and defuzzification (layer 4).

Using the self-organizing learning algorithm proposed in this paper, the third and fourth layers of the FNN are constructed during the learning procedure. Additional training patterns could be learned at any time by restarting the learning algorithm from the Step 2 with $M = M_0$ (M_0 is the number of previous learned patterns). As a result, new additional FN's in the third and fourth layers will be added when distinct patterns are used in the additional training patterns. If a learned pattern or a pattern similar to one of the learned patterns is fed to the FNN, the FNN will treat this pattern as a previous learned pattern without relearning it. Whether an additional pattern will be treated as a distinct pattern or as a learned pattern is determined by the similarities of the additional pattern with all the learned patterns, and by the learning parameters α, β and T_f. α and β affect the computation of similarities. T_f is the fault tolerance of the FNN. If one of the similarities of an input pattern is larger than or equal to $1 - T_f$, then this input pattern is treated as a previously learned pattern. Otherwise, it is treated as a new distinct pattern. We can see that the learning of this FNN can be a continual process. In other words, the FNN needs not to be completely trained before being used. This is somewhat like the learning ability of the human brain.

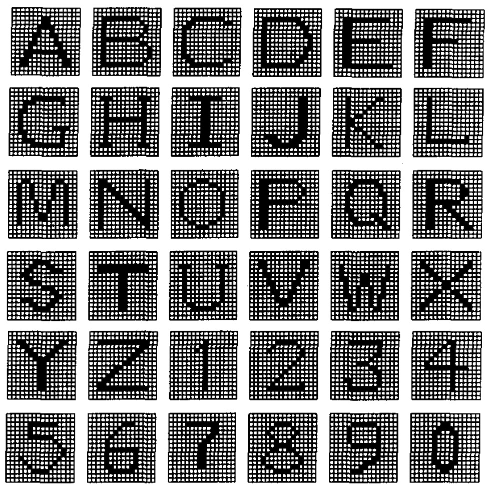


Fig. 5. 36 exemplar patterns.

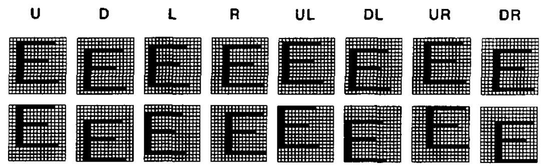


Fig. 6. Shifts of exemplar E in eight directions by 1 pixel and 2 pixels (U: upward, D: downward, L: left, R: right, UL: up-left, DL: down-left, UR: up-right, DR: down-right).

The proposed FNN is used to process 2-D patterns. The FN's in its first two layers are displayed and indexed in two-dimension. Since the 2-D structure of the first two layers only affect the indexes of the connection weights between the first and the second layers, the FNN can be implemented in hardware as if it were a 1-D structure by using the corresponding 1-D indexes of the connection weights. Besides being a fuzzy system, this FNN also possesses the advantages of high-speed parallelism, and the learning ability of neural networks.

VI. Simulation Results

We have simulated the proposed four-layer feedforward FNN on 486-PC (33 MHz) using C language. 36 exemplar patterns (as shown in Fig. 5), consisting of the 26 English alphabets and the 10 Arabic numerals (0 to 9), have been constructed. They are represented in 16×16 format and have 0, 1 pixel values. Each of the 36 exemplar patterns was shifted in eight directions by 1 pixel and 2 pixels. The eight

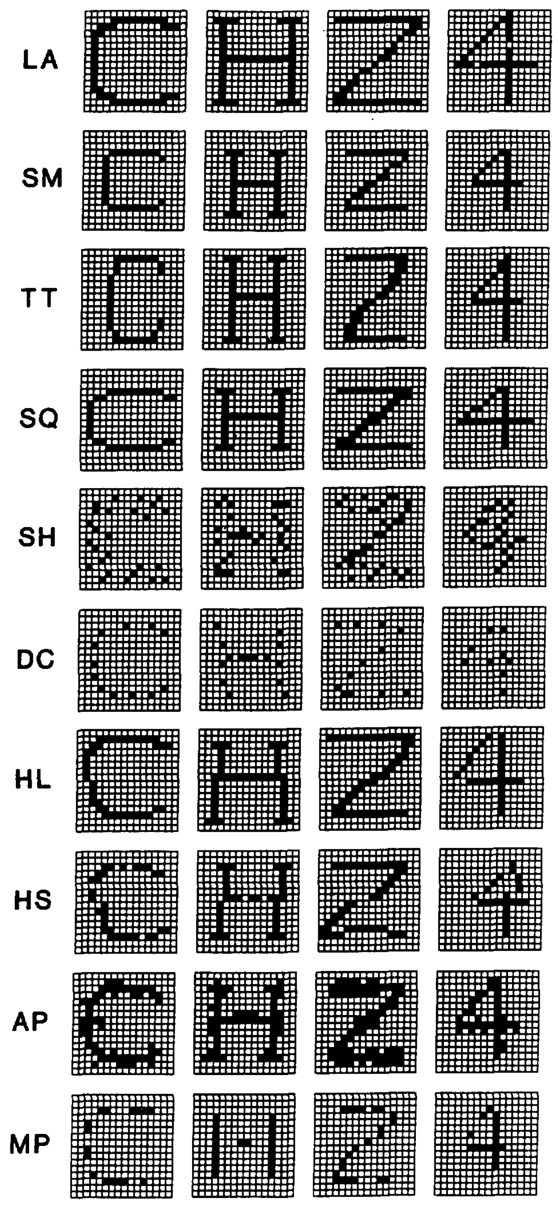


Fig. 7. 10 distorted versions of exemplars C, H, Z, 4 (LA: larger, SM: smaller, TT: taller and thinner, SQ: squished, SH: shaking, DC: disconnected, HL: half-part larger, HS: half-part shifted, AP: added small parts, MP: missed small parts).

directions are: upward (U), downward (D), left (L), right (R), up-left (UL), up-right (UR), down-left (DL) and down-right (DR). Fig. 6 gives all the 16 shifts of the exemplar pattern E. The 36 exemplars and a variety of their shifted versions are used as training patterns. We have also used 10 kinds of distorted patterns for each of the 36 exemplars to test the performances of the FNN. The 10 kinds of distortions are: larger (LA), smaller (SM), taller and thinner (TT), squished (SQ), shaking (SH), disconnected (DC), half-part larger (HL), half-part shifted (HS), added small parts (AP), missed small parts (MP). Fig. 7 gives all the 10 kinds of distortions of four of the 36 exemplar patterns, exemplar C, exemplar H, exemplar Z, and exemplar 4.

TABLE I
FOUR SETS OF TRAINING PATTERNS OF FNN

Training Set	Patterns included in Training Set
Set 1	36 exemplars
Set 2	36 exemplars & 36 exemplars shifted left by 1 pixel & 36 exemplars
Set 3	36 exemplars & 36 exemplars shifted upward by 1 pixel & 36 exemplars shifted downward by 1 pixel
Set 4	36 exemplars & 36 exemplars shifted left by 1 pixel & 36 exemplars shifted right by 2 pixel

TABLE II
FNN STRUCTURE AND TRAINING TIME TRAINED BY DIFFERENT TRAINING SETS WITH DIFFERENT T_f ($\alpha = 2.0 \& \beta = 0.3$)

Training Set	T_f	Total Number of Training Patterns	Number of FNs in 3rd & 4th Layers	Training Time (secs)
Set 1	0.10	36	36	64.0
	0.25	36	36	64.0
	0.35	36	35	63.9
Set 2	0.10	108	72	211.7
	0.25	108	36	199.9
	0.35	108	36	199.9
Set 3	0.10	108	108	215.9
	0.25	108	36	199.9
	0.35	108	36	199.9
Set 4	0.10	108	108	215.9
	0.25	108	72	203.8
	0.35	108	71	203.5

TABLE III
FNN STRUCTURE AND TRAINING TIME TRAINED BY TRAINING SET 1 WITH DIFFERENT α AND T_f ($\beta = 0.3$)

α	T_f	Total Number of Training Patterns	Number of FNs in 3rd & 4th Layers	Training Time (secs)
1.5	0.10	36	36	64.0
1.5	0.25	36	36	64.0
1.5	0.35	36	36	64.0
2.0	0.10	36	36	64.0
2.0	0.25	36	36	64.0
2.0	0.35	36	35	63.9
3.5	0.10	36	36	64.0
3.5	0.25	36	33	58.2
3.5	0.35	36	22	57.4

In the simulation experiments, 4 sets of patterns which contain the 36 exemplars and a variety of their shifted versions were used as training pattern sets. The patterns chosen in each set are listed in Table I. By using these 4 sets of training patterns, the FNN was consisted of 16×16 INPUT-FN's in the first layer and 16×16 MAX-FN's in the second layer. The number of FN's in the third and fourth layers was determined by the learning procedure. In the simulation experiments, different values of α, β and T_f were used for training the FNN. Tables II, III, and IV give the resultant FNN structures and the corresponding training time in CPU seconds (s) when the Training Set 1 was used with different combinations of α, β and T_f.

From Table II, we note that if the same pattern is fed to the FNN several times, the FNN will store its features only once. If a pattern that is similar to one of the learned patterns (the similarity is larger than or equal to $1 - T_f$) is fed to the FNN, the FNN will treat this pattern as a previously learned pattern without relearning it. If a pattern that is not similar to any of the learned patterns (all the similarities are less than

TABLE IV
FNN STRUCTURE AND TRAINING TIME TRAINED BY TRAINING SET 1 WITH DIFFERENT β AND T_f ($\alpha = 2.0$)

ß	T_f	Total Number of Training Patterns	Number of FNs in 3rd & 4th Layers	Training Time (secs)
0.1	0.10	36	32	58.9
0.1	0.25	36	9	56.8
0.1	0.35	36	6	56.3
0.3	0.10	36	36	64.0
0.3	0.25	36	36	64.0
0.3	0.35	36	35	63.9
1.6	0.10	36	36	64.0
1.6	0.25	36	36	64.0
1.6	0.35	36	36	64.0

TABLE V
RECOGNITION RATE OF FNN TRAINED BY TRAINING SET 1 FOR 1 PIXEL SHIFTED PATTERNS WITH DIFFERENT α AND T_f ($\beta = 0.3$).

α	T_f	Recognition Rate (%)								
		U	D	L	R	UL	UR	DL	DR	Average
1.5	0.10	100	100	100	100	100	100	100	100	100
1.5	0.25	100	100	100	100	100	100	100	100	100
1.5	0.35	100	100	97.22	97.22	94.44	100	97.22	100	98.26
2.0	0.10	100	100	100	100	100	100	100	100	100
2.0	0.25	100	100	100	100	100	100	100	100	100
2.0	0.35	97.22	97.22	97.22	97.22	97.22	97.22	97.22	97.22	97.22
3.5	0.10	100	100	100	100	100	100	100	100	100
3.5	0.25	91.67	91.67	91.67	91.67	91.67	91.67	91.67	91.67	91.67
3.5	0.35	61.11	61.11	61.11	61.11	61.11	61.11	61.11	61.11	61.11

TABLE VI
RECOGNITION RATE OF FNN TRAINED BY TRAINING SET 1 FOR 1 PIXEL SHIFTED PATTERNS WITH DIFFERENT β ($\alpha = 2.0 \& T_f = 0.1$)

ß	Recognition Rate (%)								
	U	D	L	R	UL	UR	DL	DR	Average
1.6	100	100	100	100	0.0	0.0	0.0	0.0	50.00
0.3	100	100	100	100	100	100	100	100	100
0.1	100	100	100	100	80.55	86.11	77.78	83.33	90.97

$1 - T_f$) is fed to the FNN, the FNN will treat this pattern as a distinct pattern and store its features. Table III indicates that if α value is large, T_f should be small in order to obtain a FNN that can separate all the distinct training patterns. If the differences between distinct training patterns are small, both α and T_f should be small. From Table IV, we note that the selection of β value will also affect the ability of the FNN in separating distinct training patterns. If the differences between distinct training patterns are small and T_f is large, β should be large. From Tables II, III, and IV, we can see that whether a pattern will be treated as a distinct pattern or as a learned pattern is determined by the similarities of the pattern to the learned patterns, and also by the learning parameters α, β and T_f. Hence, the structure of the FNN is determined by the training parameters and the training patterns. The FNN can recall all of the learned distinct patterns.

After being trained, the FNN was then tested using shifted and distorted patterns. The recognition time for each pattern is 1.9 CPU s. In order to analyze the effects of the learning parameters α, β and T_f on recognition rates, several experiments have been done. Shifted patterns have been provided to different FNN's trained by the Training Set 1 with different values of α, β and T_f. The recognition rates of these FNN's for all of the patterns shifted by 1 pixel are summarized in Tables V and VI.

From the results of Tables II, III, IV, V, and VI, we note that if the FNN cannot separate all the distinct training patterns, the recognition rates will be low. So α and T_f should be small enough, and β should be large enough to enable the FNN to separate all the distinct training patterns. However, too small α and T_f values or too large β value will also result in low recognition rates. Very small T_f will result in a FNN that treats all the different training patterns as distinct patterns and therefore more FN's are needed in the third and fourth layers. Consequently, the value of α, β and T_f can not be too small or too large. In the experiments, if $\alpha = 2(P_{v\max} - P_{v\min})$, where $P_{v\max}$ and $P_{v\min}$ are, respectively, the maximum and minimum pixel values among all the input patterns, and β is chosen such that $0.5 = \exp[-\beta^2(\delta_x^2 + \delta_y^2)]$, where $\delta_x = 2$ and $\delta_y = 2$ are, respectively, the largest shifted pixels in x and y directions, the resultant FNN will have a high recognition rate.

TABLE VII
RECOGNITION RATE OF FNN TRAINED BY TRAINING SET 1 FOR SHIFTED PATTERNS ($\alpha = 2.0, \beta = 0.3 \& T_f = 0.1$)

No. of Shifted Pixels	Recognition Rate (%)								
	U	D	L	R	UL	UR	DL	DR	Average
1	100	100	100	100	100	100	100	100	100
2	91.67	91.67	100	100	91.67	88.88	94.44	77.77	92.01

TABLE VIII
RECOGNITION RATE OF FNN TRAINED BYTRAINING SET 3 FOR SHIFTED PATTERNS ($\alpha = 2.0, \beta = 0.3$ AND $T_f = 0.1$)

No. of Shifted Pixels	Recognition Rate (%)								
	U	D	L	R	UL	UR	DL	DR	Average
1	100	100	100	100	100	100	100	100	100
2	100	100	100	100	97.22	100	100	91.67	98.61

Tables VII and VIII give the recognition rates of the FNN with shifted patterns. When the FNN was trained by the Training Set 1 with $\alpha = 2.0$, $\beta = 0.3$ and $T_f = 0.1$, it can recognize patterns shifted by 1 pixel with 100% recognition rate and patterns shifted by 2 pixels with an average recognition rate of 92.01%. When the FNN was trained by the Training Set 3 with $\alpha = 2.0$, $\beta = 0.3$ and $T_f = 0.1$, it can recognize patterns shifted by 1 pixel with 100% recognition rate and patterns shifted by 2 pixels with an average recognition rate of 98.61%. From these results, we can see that the recognition rate of the FNN becomes lower when the shiftings of the patterns are larger. However, the recognition rates can be improved by training the FNN with some more shifted patterns.

We have also tested the FNN with 10 kinds of distorted patterns for each of the 36 exemplars. The FNN can recognize all of the distorted patterns with 100% recognition rates when trained by the Training Set 1 with $\alpha = 2.0$, $\beta = 0.3$ and $T_f = 0.1$, and by the Training Set 3 with $\alpha = 2.0$, $\beta = 0.3$ and $T_f = 0.1$. The recognition results of the FNN for these distorted patterns are summarized in Tables IX and X.

For illustration, the performances of the FNN are compared to that of the Hamming network [21], which classifies an input pattern by computing the Hamming distances between the input pattern and all the training patterns. When trained by the Training Set 1, the Hamming network can reach only 26.7%

TABLE IX
RECOGNITION RATE OF FNN TRAINED BY TRAINING SET 1 FOR DISTORTED PATTERNS ($\alpha = 2.0, \beta = 0.3$ AND $T_f = 0.1$).

Recognition Rate (%)										
LA	SM	TT	SQ	SH	DC	HL	HS	AP	MP	Average
100	100	100	100	100	100	100	100	100	100	100

TABLE X
RECOGNITION RATE OF FNN TRAINED BY TRAINING SET 3 FOR DISTORTED PATTERNS ($\alpha = 2.0, \beta = 0.3$ AND $T = 0.1$)

Recognition Rate (%)										
LA	SM	TT	SQ	SH	DC	HL	HS	AP	MP	Average
100	100	100	100	100	100	100	100	100	100	100

TABLE XI
RECOGNITION RATE OF HAMMING NETWORK TRAINED BY TRAINING SET 1 FOR SHIFTED PATTERNS

No. of Shifted Pixels	Recognition Rate (%)								
	U	D	L	R	UL	UR	DL	DR	Average
1	44.44	47.22	44.44	47.22	8.33	11.11	5.56	5.56	26.74
2	22.22	27.78	11.11	5.56	2.78	2.78	2.78	5.56	10.07

TABLE XII
RECOGNITION RATE OF HAMMING NETWORK TRAINED BY TRAINING SET 1 FOR DISTORTED PATTERNS

Recognition Rate (%)										
LA	SM	TT	SQ	SH	DC	HL	HS	AP	MP	Average
19.44	13.88	22.22	11.11	38.89	50.00	41.67	52.77	97.22	91.67	43.89

and 10.1% recognition rates, respectively, for patterns shifted by 1 pixel and 2 pixels. For distorted patterns, the average recognition rate of the Hamming network was 43.89%. The results are summarized in Tables XI and XII. From Tables VII to XII, we can see that the performances of the FNN are better than those of the Hamming network for shifted patterns and distorted patterns. The FNN can recall all of the distorted patterns correctly while the Hamming network can not. Nevertheless, the Hamming network works well for two kinds of distorted patterns (AP and MP) as shown in Table XII.

VII. CONCLUSION

In this paper, we have defined a fuzzy neuron, introduced four types of fuzzy neurons, and proposed a four-layer feedforward FNN with its associated learning algorithm which can be used in pattern recognition. The proposed FNN possesses the structure and learning ability of a neural network while using fuzzy algorithms to process patterns. The proposed FNN can learn additional new patterns at any time without relearning the learned patterns or patterns similar to the learned patterns. Only those new patterns with distinct features will be learned by the FNN. The learning speed of the proposed FNN is faster than that of a neural network using the backpropagation algorithm. The structure of this FNN is simple and its recognition time is fast. The proposed FNN performs well as demonstrated by the results obtained in the alphanumeric character recognition problem, even when the input patterns are shifted or distorted. The recognition rates of this FNN are higher than those of the Hamming network. The results of the proposed FNN suggest that the formulation of a fuzzy neural network by combining the strengths of fuzzy logic and neural networks is fruitful. The proposed FNN can also be adapted for applications in some other pattern recognition problems.

REFERENCES

[1] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA: MIT Press, 1986.

[2] C. Lau, Ed., *Neural Networks, Theoretical Foundations and Analysis*. Piscataway, NJ: IEEE Press, 1992.

[3] E. Sanchez-Sinencio and C. Lau, Eds., *Artificial Neural Networks, Paradigms, and Hardware Implementations*. Piscataway, NJ: IEEE Press, 1992.

[4] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE Trans. Syst., Man, and Cybern.*, vol. SMC-13, no. 5, pp. 826–834, Sept./Oct. 1983.

[5] K. Fukushima and N. Wake, "Handwritten alphanumeric character recognition by neocognitron," *IEEE Trans. Neural Networks*, vol. 2, no. 3, pp. 355–365, May 1991.

[6] G. A. Carpenter and S. Grossberg, "The ART of adaptive pattern recognition by a self-organizing neural network," *IEEE Computer Mag.*, vol. 21, no. 3, pp. 77–88, March 1988.

[7] G. L. Martin and J. A. Pittman, "Recognizing hand-printed letters and digits using backpropagation learning," *Neural Computation*, vol. 3, no. 2, pp. 258–267, Summer 1991.

[8] I. Guyon, P. Albrecht, Y. L. Cun, J. Denker, and W. Hubbard, "Design of a neural network character recognizer for a touch terminal," *Pattern Recognition*, vol. 24, no. 2, pp. 105–117, Feb. 1991.

[9] M. Fukumi, S. Omatu, F. Takeda, and T. Kosaka, "Rotation-invariant neural pattern recognition system with application to coin recognition," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 272–279, Mar. 1992.

[10] S. J. Perantonis and P. J. G. Lisboa, "Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 241–251, Mar. 1992.

[11] L. A. Zadeh, "The role of fuzzy logic in the management of uncertainty in expert systems," in *Approximate Reasoning in Expert System*, Gupta, Kandel, Bandler, and Kiszka, Eds. New York: Elsevier, 1985.

[12] S. K. Pal and D. K. D. Majumder, *Fuzzy Mathematical Approach to Pattern Recognition*. New Delhi, India: Wiley Eastern Ltd., 1986.

[13] J. C. Bezdek and S. K. Pal, Eds., *Fuzzy Models for Pattern Recognition*. Piscataway, NJ: IEEE Press, 1992.

[14] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[15] A. Kandel, *Fuzzy Techniques in Pattern Recognition*. New York: Wiley, 1982.

[16] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[17] T. Yamakawa and S. Tomoda, "A fuzzy neuron and its application to pattern recognition," in *Proc. Third Int. Fuzzy System Associat. Congress*, Japan, 1989, pp. 30–38.

[18] H. Takagi, T. Kouda, and Y. Kojima, "Neural-network design on approximate reasoning and its application to the pattern recognition," in *Proc. Int. Conf. of Fuzzy Logic and Neural Networks*, Iizuka, Japan, July 1990, pp. 671–674.

[19] L. Wang and J. M. Mendel, "A fuzzy approach to hand-written rotation-invariant character recognition," in *Proc. 1992 Int. Conf. on Acoustics, Speech, and Signal Processing*, San Francisco, CA, Mar. 23–26, 1992, pp. III-145–III-148.

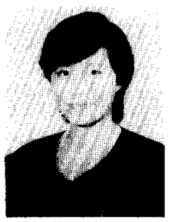
[20] R. J. Machado and A. F. Rocha, "A hybrid architecture for fuzzy connectionist expert systems," in A. Kandel and G. Langholz, Eds., *Hybrid Architectures for Intelligent Systems*. Boca Raton, FL: CRC Press, 1992.

[21] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, no. 2, pp. 4–22, Apr. 1987.

Hon Keung Kwan (M'81–SM'86) received the B.Sc. degree from the the University of London, United Kingdom and the M.Phil. degree from the Chinese University of Hong Kong, Hong Kong, in 1976 and 1977, respectively. He received the D.I.C. and Ph.D. degree in 1981 from Imperial College, University of London.

During 1977–1978, he was with Interquartz Ltd., Hong Kong, as an Assistant Design Engineer and later Tek-Devices Ltd., Hong Kong as a Design Engineer. In 1981, he joined the Department of Electronic Engineering, Hong Kong Polytechnic as a Lecturer. Since December 1981, he has been with the Department of Electrical and Electronic Engineering, University of Hong Kong as a Lecturer. In 1988, he joind the Department of Electrical Engineering, University of Windsor, Canada, as an Associated Professor, where he is currently a Professor.

Dr. Kwan is a member of the Digial Sygnal Processing Technical Committee, the Neural Systems and Applications Technical Committee of the IEEE Circuits and Systems Society, and the Institution of Electrical Engineers (UK) and a Chartered Electrical Engineer (UK). His current research interests include high-speed digital filters, adaptive digital filters, systolic architectures, and fuzzy neural systems.



Yaling Cai (S'93) received the B.E. degree in electrical engineering from Tsinghua University, P.R. China, in 1985, and the M.E. degree in electrical engineering from the Second Institute of Aerospace Ministry, P.R. China, in 1988.

She is now working towards the Ph.D. degree in electrical engineering at the University of Windsor, Canada. Her research interests include fuzzy systems, neural networks, and signal processing.