Design and Training for Combinational Neural-Logic Systems

H. K. Lam, *Member, IEEE*, and Frank H. F. Leung, *Senior Member, IEEE*

***Abstract*—This paper presents the combinational neural-logic system. The basic components, i.e., the neural-logic-AND, -OR, and -NOT gates, will be proposed. As different applications have different characteristics, a traditional neural network with a common structure might not handle every application well if some network connections are redundant and cause internal disturbances, which may downgrade the training and network performance. In this paper, the proposed neural-logic gates are the basic building blocks for the applications. Based on the knowledge of the application and the neural-logic design methodology, a combinational neural-logic system can be designed systematically to incorporate the characteristics of the application into the structure of the combinational neural-logic system. It will enhance the training and network performance. The parameters of the combinational neural-logic system will be trained by the genetic algorithm. To illustrate the merits of the proposed approach, the combinational neural-logic system will be realized practically to recognize Cantonese speech commands for an electronic book.**

***Index Terms*—Cantonese speech recognition, combinational neural-logic system, neural network.**

I. Introduction

NEURAL networks [1], [2] are useful tools for many applications due to its powerful approximation ability. A neural network was proved to be able to serve as a universal approximator for any continuous nonlinear function in a compact domain [1], [2]. A three-layer fully connected feedforward neural network can approximate any nonlinear continuous function to an arbitrary accuracy in a compact domain. To obtain the network structure automatically, constructive and destructive algorithms can be used [3]. A constructive algorithm starts with a small network. Hidden layers, nodes, and connections are added to expand the network dynamically [4]. A destructive algorithm starts with a large network. Hidden layers, nodes, and connections are then removed to contract the network dynamically [3]. In [6], by introducing switches into the links of the neural network, the network structure can be tuned by the genetic algorithm (GA) [5]. One of the implications behind these algorithms is that different structures of neural networks are needed for different tasks of different characteristics. Hence, the neural network should be designed based on the characteristic of the task to provide good performance.

Manuscript received June 1, 2004; revised April 4, 2006. Abstract published on the Internet September 15, 2006. This work was supported by a grant from the Hong Kong Polytechnic University (Project G-YX31).

H. K. Lam is with the Department of Electronic Engineering, Division of Engineering, King's College London, WC2R 2LS London, U.K.

F. H. F. Leung is with the Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong.

Digital Object Identifier 10.1109/TIE.2006.885446

To implement a given binary logic function, a combinational logic circuit consisting of three basic logic gates (AND, OR, and NOT gates) can be designed in a very systematic and efficient way. The designed combinational logic circuit is guaranteed to have the simplest form. By extending this idea to a neural system, the binary logic function is analogous to the input–output relation of the system. Similar ideas using fuzzy gates have been proposed in [7]. In [8], a fuzzy microprocessor was presented and constructed. In [9] and [10], some analogical gates that are obtained from fuzzy generalization of Boolean logical gates were proposed. These analogical gates have been employed in the control of nonholonomic mobile robots [9], [10] and the synthesis of multicomponent separation processes [11]. However, a systematic design procedure was not provided. The concept of fuzzy J-K flip-flops was proposed in [12]. The binary flip-flops were extended to fuzzy flip-flops using t-norms, s-norms, and fuzzy negation defined by the fuzzy set theory. A family of fuzzy J-K flip-flops using different t-norms, s-norms, and fuzzy negation was investigated in [13]. However, no industrial applications were mentioned using these fuzzy flip-flops. In [14] and [15], a weight-added fuzzy flip-flop with learning ability was proposed. A learning algorithm like the backpropagation algorithm was also proposed for this flip-flop.

Traditionally, neural networks with the same structure were applied to handle different applications. Owing to the different characteristics of applications, neural networks with the same structure might not give the optimal solution. Some of the connections and weights could be redundant, which makes the number of the network parameters unnecessarily large. Furthermore, these redundant connections may cause internal disturbances [3], [16] that downgrade the training and network performance. The traditional neural network approach is good in handling applications without too much information. Practically, engineers should be able to collect some information from the real-world applications. This information provides some important knowledge to tell the characteristics of the application. In this paper, some novel basic building blocks, namely neural-logic-AND, -OR, and -NOT gates, are proposed. By extending the binary logic design algorithm, a combinational neural-logic system can be designed systematically to incorporate the characteristics of the application to the network structure so as to enhance the training and network performance. Furthermore, the structure should be simpler as compared with that of the traditional neural network because some redundant connections are removed. A combinational neural-logic system thus designed is dedicated to the application.

GA is a directed random search technique [5] that is widely applied in optimization problems to find out the optimal

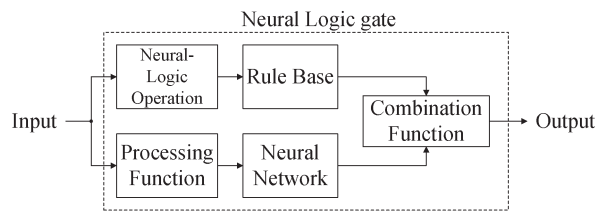

Fig. 1. Block diagram of a neural-logic gate.

solution globally over a domain. It is especially useful for complex optimization problems where the number of parameters is large and the analytical solutions are difficult to obtain. In this paper, GA is employed to train the parameter values of the combinational neural-logic systems. As the training of GA is based on the input–output data only, the derivative information of the cost function need not be known. This feature makes GA suitable for the combinational neural-logic systems, which have different structures for different applications. It is because if the training algorithm (e.g., gradient-descent algorithms) depends on some derivative information of the cost function, the updating rule is needed to be derived each time for each different network structure. Nevertheless, it should be noted that other training algorithms can possibly be used to carry out the training.

This paper is organized as follows. The neural-logic gates will be presented in Section II. The design and learning of the combination neural-logic system will be presented in Section III. As an example, Cantonese speech commands recognized by the proposed combinational neural-logic system will be given in Section IV. A conclusion will be drawn in Section V.

II. Neural-Logic Gates

In this paper, the proposed neural-logic gates consist of five parts, namely: 1) neural-logic operation; 2) rule base; 3) processing function; 4) neural network; and 5) combination function, as shown in Fig. 1. First, the inputs of the neural-logic gate will undergo a neural-logic operation, which will be discussed later. Its output will be passed to the rule base, which stores the boundary conditions and the properties of the neural-logic gates. The processing function and the neural network introduce the nonlinearity to the neural-logic gates for error correction. The output of the neural-logic gate is the combination of the outputs of the rule base and the neural network.

A. *Neural-Logic-*AND *Gate*

A two-input–single-output neural-logic-AND gate is proposed. The inputs and output lie between 0 and 1 inclusively. Referring to Fig. 1, the neural-logic-AND operation is defined as

$$x_1(t) \otimes x_2(t) = 0 \vee (x_1(t) + x_2(t) - 1) \in [0 \quad 1] \qquad (1)$$

which is actually the bounded product [12] of the inputs $x_1(t)$ and $x_2(t)$, where t denotes the current number of input vector, which is a nonzero integer, and $\vee$ denotes the maximum operator. The output of the neural-logic-AND operation will be fed to the rule base, which guarantees the boundary conditions and properties of a binary AND gate as shown in Table I. Throughout this paper, the neural-logic-AND operator is denoted by a "$\circ$," e.g., $x_1(t)$ AND $x_2(t)$ is written as $x_1(t) \circ x_2(t)$.

TABLE I
Boundary Conditions and Properties of the Neural-Logic-AND Gates

$A \circ 0 = 0$	$A \circ 1 = A$	$A \circ \bar{A} = 0$	$A \circ A = A$
$0 \circ 0 = 0$	$0 \circ 1 = 0$	$1 \circ 0 = 0$	$1 \circ 1 = 1$

For the neural network in Fig. 1, a three-layer fully connected feedforward neural network is employed. Before feeding the input signals to the neural network, the input signals $x_1(t)$ and $x_2(t)$ will be processed by a processing function $\mathbf{h}(\cdot)$, which is to be designed. The purpose of introducing the processing function $\mathbf{h}(\cdot)$ is to transform the inputs to a desirable domain to facilitate the subsequent processes. For the neural network, $\mathbf{x}(t) = [x_1(t) \quad x_2(t) \quad \cdots \quad x_{n_{\text{in}}}(t)]$ denotes the input vector; n_{in} denotes the number of input nodes ($n_{\text{in}} = 2$ for a two-input–single-output neural-logic-AND gate); w_{ij}^1, $i = 1, 2, \ldots, n_h$, $j = 1, 2, \ldots, n_{\text{in}}$, denote the connection weights between the input layer and the hidden layer; n_h denotes the number of hidden nodes; w_{ki}^2, $k = 1, 2, \ldots, n_{\text{out}}$, $i = 1, 2, \ldots, n_h$, denote the connection weights between the hidden layer and the output layer; n_{out} denotes the number of output nodes ($n_{\text{out}} = 1$ for a two-input–single-output neural-logic-AND gate); b_i^1, $i = 1, 2, \ldots, n_h$, and b_k^2, $k = 1, 2, \ldots, n_{\text{out}}$, denote the bias weights to the ith hidden and kth output nodes, respectively; $t_f^1(\cdot)$ and $t_f^2(\cdot)$ denote the activation functions of the hidden and output layers, respectively; and $\mathbf{g}(t) = [g_1(t)] \quad g_2(t) \quad \cdots \quad g_{n_{\text{out}}}(t)]$ denotes the output vector. The output of the neural network is defined as

$$g_k(t) = t_f^2\left(\sum_{i=1}^{n_h} t_f^1\left(\mathbf{h}(\mathbf{x}(t))\,\mathbf{w}_i + b_i^1\right) w_{ki}^2\right) + b_k^2, \qquad k = 1, 2, \ldots, n_{\text{out}} \qquad (2)$$

where $\mathbf{w}_i = [w_{i1}^1 \quad w_{i2}^1 \quad \cdots \quad w_{in_{\text{in}}}^1]^T$. The parameters w_{ij}^1, w_{ki}^2, b_i^1, and b_k^2, which form genes of the chromosome, will be tuned by the real-coded GA with arithmetic crossover and nonuniform mutation [5]. The total number of parameters for each neural network is $(n_{\text{in}} + 1)n_h + (n_h + 1)n_{\text{out}}$. As the neural-logic-AND gate has two inputs and one output, from (2), the input–output relationship of the neural network inside the neural-logic-AND gate is defined as

$$g_1(t) = t_f^2\left(\sum_{i=1}^{n_h} t_f^1\left(\mathbf{h}(\mathbf{x}(t))\,\mathbf{w}_i + b_i^1\right) w_{1i}^2\right) + b_1^2. \qquad (3)$$

Consequently, the output of the neural-logic-AND gate is defined as

$$y_{\text{AND}}(t) = (0 \vee f_{\text{AND}}(x_1(t) \otimes x_2(t), g_1(t))) \wedge 1 \qquad (4)$$

subject to the properties of Table I. The symbol $\wedge$ denotes the minimum operator. $f_{\text{AND}}(\cdot)$ is the combination function of the neural-logic-AND gate to be designed. The proposed neural-logic-AND gate satisfies the boundary conditions and exhibits the properties of the binary AND gate.

TABLE II
BOUNDARY CONDITIONS AND PROPERTIES OF THE NEURAL-LOGIC-OR GATES

$A \bullet 0 = A$	$A \bullet 1 = 1$	$A \bullet A = A$	$A \bullet \overline{A} = 1$
$0 \bullet 0 = 0$	$0 \bullet 1 = 1$	$1 \bullet 0 = 1$	$1 \bullet 1 = 1$

B. *Neural-Logic-*OR *Gate*

A two-input–single-output neural-logic-OR gate is proposed. The inputs and output lie between 0 and 1 inclusively. Referring to Fig. 1, the neural-logic-OR operation is defined as

$$x_1(t) \oplus x_2(t) = (x_1(t) + x_2(t)) \wedge 1 \in [0 \quad 1] \tag{5}$$

which is actually the bounded sum [12] of the inputs $x_1(t)$ and $x_2(t)$. The output of the neural-logic-OR operation will be fed to the rule base, which guarantees the boundary conditions and the properties of a binary OR gate as shown in Table II. Throughout this paper, the neural-logic-OR operator is denoted by a "•," e.g., $x_1(t)$ OR $x_2(t)$ is written as $x_1(t) \bullet x_2(t)$. The neural network of the neural-logic-OR gate is the same as that of the neural-logic-AND gate. It should be noted that the parameters of the neural network for each individual logic gate are not shared. The output of the neural-logic-OR gate is defined as

$$y_{\mathrm{OR}}(t) = (0 \vee f_{\mathrm{OR}}(x_1(t) \oplus x_2(t), g_1(t))) \wedge 1 \tag{6}$$

subject to the properties of Table II. $f_{\mathrm{OR}}(\cdot)$ is the combination function of the neural-logic-OR gate to be designed. The proposed neural-logic-OR gate satisfies the boundary conditions and exhibits the properties of the binary OR gate.

C. *Neural-Logic-*NOT *Gate*

The neural-logic-NOT gate has one input and one output. The output of the neural-logic-NOT gate is defined as

$$y_{\mathrm{NOT}}(t) \in [0 \quad 1] = 1 - x(t). \tag{7}$$

It can be seen from (7) that the characteristics of a binary logic NOT gate are retained, i.e., NOT $0 = 1$ and NOT $1 = 0$. The neural-logic-NOT operator is denoted by a "bar," e.g., NOT $x_1(t)$ is written as $\overline{x}_1(t)$.

III. DESIGN AND LEARNING OF COMBINATIONAL NEURAL-LOGIC SYSTEMS

In this section, the design and the learning of the combinational neural-logic system will be presented.

A. *Design*

The design of the combinational neural-logic system, which is formed by some neural-logic-AND, -OR, and -NOT gates, will be considered. Fig. 2 shows the block diagram of a multiple-input–multiple-output combinational neural-logic system. The inputs are denoted by $z_1, z_2, \ldots, z_{n_i}$, where n_i denotes the number of inputs. The outputs are denoted by $y_1, y_2, \ldots, y_{n_o}$, where n_o denotes the number of outputs. The multiple-input–multiple-output combinational neural-logic system consists of three parts, namely 1) normalizer, 2) combinational neural-logic circuit, and 3) denormalizer. As the inputs of the defined neural-logic gates take values between 0 and 1, the normalizer is used to normalize the values of the actual inputs to values inside this range. Similarly, as the outputs of the defined neural-logic gates take values between 0 and 1, the denormalizer is used to map the values of the outputs of the combinational neural-logic circuit to the actual output ranges. The combinational neural-logic circuit is to implement a given neural-logic function (i.e., input–output relation). The (normalized) inputs and outputs of the combinational neural-logic circuit are denoted by $\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_{n_i}$ and $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{n_o}$, respectively.

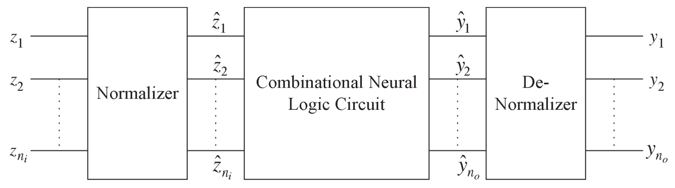

Fig. 2. Block diagram of a combinational neural-logic system.

TABLE III
TRUTH TABLE

$\hat{z}_1$	$\hat{z}_2$	$\hat{y}_1$
L	L	H
L	H	H
H	L	H
H	H	L

We illustrate the design of the combinational neural-logic circuit using an example. We learn the input–output relationship of an unknown function based on its input–output data pairs. First, we have to set up a truth table based on some expert knowledge. The truth table governs the relations among some linguistic states [e.g., high (H) and low (L)]. It should be noted that H and L are linguistic variables, which are different from those in binary logic systems. H and L in binary logic systems refer to crisps 1 and 0, respectively. In the combinational neural-logic system, H and L refer to a state, which is about 1 and 0, respectively. If a combinational neural-logic circuit have two inputs ($\hat{z}_1$ and $\hat{z}_2$,) and one output ($\hat{y}_1$), and the two linguistic states for each variable are H and L, a truth table for the circuit can be given by Table III. These two linguistic states divide the input region (0 to 1) into two subregions. Taking the first rule (row 2 of Table III) as an example, it can be interpreted that if $\hat{z}_1$ is about L and $\hat{z}_2$ is about L, then $\hat{y}_1$ is about H. These rules are determined based on human knowledge about the problem to be handled.

Based on the truth table, a combinational neural-logic circuit can be designed. With reference to the state H, taking $\hat{z}_1$ for example, $\hat{z}_1$ and $\overline{\hat{z}}_1$($\bar{\cdot}$ denotes the neural-logic NOT of an argument) mean that $\hat{z}_1$ is about H and $\hat{z}_1$ is about L in the neural-logic function, respectively (the same notations as those in a binary logic system). Referring to the properties of neural-logic gates tabulated in Tables I and II, the output $\hat{y}_1$ can be obtained through a few steps of linear algebra manipulations as follows:

$$\hat{y}_1 = \left(\overline{\hat{z}}_1 \circ \overline{\hat{z}}_2\right) \bullet \left(\overline{\hat{z}}_1 \circ \hat{z}_2\right) \bullet \left(\hat{z}_1 \circ \overline{\hat{z}}_2\right). \tag{8}$$

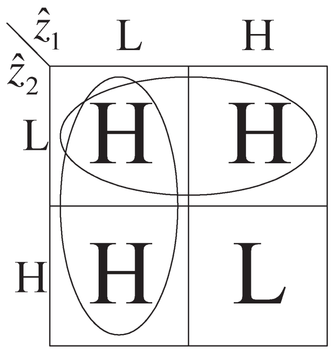

Fig. 3. K-map.

Similar to the binary logic systems, $\bar{\hat{z}}_1 \circ \bar{\hat{z}}_2$, $\bar{\hat{z}}_1 \circ \hat{z}_2$, $\hat{z}_1 \circ \bar{\hat{z}}_2$, and $\hat{z}_1 \circ \hat{z}_2$ are called the minterms of the neural-logic function. Based on the properties in Tables I and II, (8) can be rewritten as

$$\begin{aligned}\hat{y}_1 &= \left(\bar{\hat{z}}_1 \circ \bar{\hat{z}}_2\right) \bullet \left(\bar{\hat{z}}_1 \circ \bar{\hat{z}}_2\right) \bullet \left(\bar{\hat{z}}_1 \circ \hat{z}_2\right) \bullet \left(\hat{z}_1 \circ \bar{\hat{z}}_2\right) \\ &\quad \text{(by making use of the property that } A \bullet A = A) \\ &= \left(\bar{\hat{z}}_1 \circ \left(\bar{\hat{z}}_2 \bullet \hat{z}_2\right)\right) \bullet \left(\left(\bar{\hat{z}}_1 \bullet \hat{z}_1\right) \circ \bar{\hat{z}}_2\right) \\ &= \bar{\hat{z}}_1 \bullet \bar{\hat{z}}_2. \end{aligned} \tag{9}$$

It should be noted that the neural-logic function is in minimal form and can also be obtained using Karnaugh map (K-map) [17] or other methods. The K-map for obtaining (9) is shown in Fig. 3. The above example illustrates the idea of designing a two-input–single-output combinational neural-logic system. The idea can readily be extended to design a multiple-input–multiple-output combinational neural-logic system. In summary, we have illustrated an algorithm to simplify the structure of a combinational neural-logic system based on some expert knowledge.

B. Learning

In this paper, the real-coded GA with arithmetic crossover and nonuniform mutation [5] will be employed to find a set of parameters of the combinational neural-logic system to achieve a given task. To do so, a fitness function that reveals the system performance is needed to be defined, i.e.,

$$\text{fitness} = f(\beta) \tag{10}$$

where $f(\cdot)$ is a function measuring the system performance (which is to be designed according to the task), β is a vector (chromosome) containing all the parameters (genes) of the combinational neural-logic system, e.g., the parameters w^1_{ij}, w^2_{ki}, b^1_i, and b^2_k of each neural-logic gate. The objective is to maximize the value of fitness by adjusting β using GA. Usually, the value of fitness is normalized to be between 0 and 1 inclusively. The higher the value of fitness, the better is the system performance.

IV. Application Example

An application example on Cantonese speech command recognition will be given in this section to illustrate the design procedure and merits of the proposed approach. The proposed combinational neural-logic system is employed to perform the recognition. It will be designed based on expert knowledge on the task to be handled. Real-coded GA [5] will be employed to learn the network parameters. The proposed Cantonese speech command recognizer implemented by the combinational neural-logic system has been successfully implemented in an electronic book reader [19], which has a microphone for capturing the speech signals. Five Cantonese words (five classes), namely /soeng5/ (上), /haa5/ (下), /bat1/ (筆), /daai6/ (大), and /zyu3/ (註) [18], are used to activate the "page up," "page down," "pen," "zoom in," and "bookmark" actions in the electronic book reader.

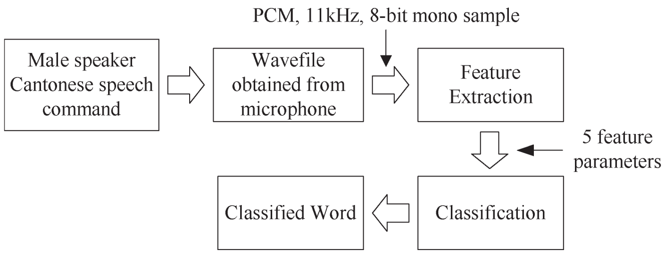
PCM, 11kHz, 8-bit mono sample
Male speaker Cantonese speech command
Wavefile obtained from microphone
Feature Extraction
5 feature parameters
Classification
Classified Word

Fig. 4. Block diagram of the five-Cantonese-speech-command recognition process.

A. Feature Coefficient Extraction

Fig. 4 shows the block diagram of the Cantonese speech recognition process. First, speech signals are recorded from a microphone in mono pulse-code modulation format at an 8-bit, 11-kHz sampling rate. The speech signals in time domain, with windowing by the hamming windows, are transformed into frequency-domain components using fast Fourier transform. A uniform filter bank is applied to model the frequency spectrum of the speech signal. The process of retrieving the feature coefficients from the uniform filter bank is defined as follows:

$$c_{o_\alpha} = \frac{10 \log \sum\limits_{i=1}^{m} s_{\alpha i}}{m}, \qquad \alpha = 1, 2, \ldots, k \tag{11}$$

where c_{o_α} denotes the mean power spectrum of the speech in the αth bandpass filter, k denotes the number of bandpass filters, m denotes the number of the frequency components at each bandpass filter, and $s_{\alpha i}$ denotes the amplitude of ith frequency component at the αth bandpass filter. The difference between c_{o_α} is defined as follows:

$$d_{o_{\alpha+1}} = c_{o_{\alpha+1}} - c_{o_\alpha}, \qquad \alpha = 1, 2, \ldots, k-1 \tag{12}$$

$$d_{o_1} = c_{o_1}. \tag{13}$$

To reduce the number of inputs of the combinational neural-logic system, the following equations [(14) and (15)] are used to reduce the number of feature parameters by half:

$$p_{o_\varsigma} = \frac{d_{o_\varsigma} + d_{o_{\varsigma+1}}}{2}, \qquad \varsigma = 1, 2, \ldots, \rho - 1 \tag{14}$$

where ρ denotes the number of feature coefficients obtained from the filter-bank filter. The feature coefficient used by the

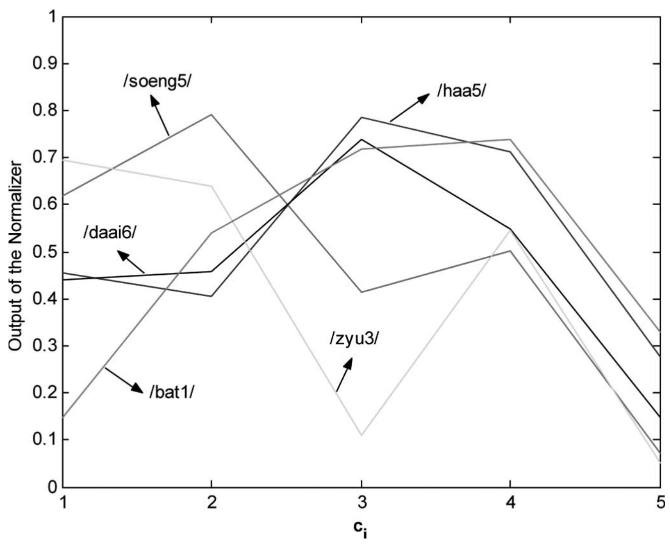

Fig. 5. Averaged values of $\mathbf{c}_i$.

proposed network is defined as

$$\mathbf{c}_i = \frac{\mathbf{P}_{\text{oo}} + \mathbf{P}_{\text{oe}}}{2} \tag{15}$$

where $\mathbf{P}_{\text{oo}}$ and $\mathbf{P}_{\text{oe}}$ denotes the odd and even elements of $\mathbf{p}_o = [p_{o_1} \quad p_{o_2} \quad \cdots \quad p_{o_{\rho-1}}]$, respectively. Referring to Fig. 4, the outputs of the feature extractor will be used as the inputs of the combinational neural-logic system, which is employed to classify the input speech commands. The inputs of the combinational neural-logic system are defined as $\mathbf{z} = \mathbf{c}_i / \|\mathbf{c}_i\|$, where $\mathbf{c}_i = \lfloor c_{i_1} \quad c_{i_2} \quad \cdots \quad c_{i_{(k-1)/2}} \rfloor$, $\|\cdot\|$ denotes the l_2 vector norm. In this paper, the number of bandpass filters is 11. Hence, from (14) and (15), the number of elements of the augmented feature coefficient vector $\mathbf{c}_i$, which are taken as the inputs of the combinational neural-logic system, is five. As there are five commands to be recognized, the combinational neural-logic system will have five outputs. The output with the maximum value indicates the possible input speech command. In the following, the combinational neural-logic system will be designed.

B. Normalizer

Referring to Fig. 2, the normalizer is defined as follows:

$$\hat{\mathbf{z}} = \frac{1}{1 + e^{\frac{-(\mathbf{z}-0.45)}{2(0.025)^2}}} \in [0 \quad 1] \tag{16}$$

where $\mathbf{z}$ denotes the input vector of the normalizer. The function of (16) spans the inputs all over the positive domain with the values lying between 0 and 1.

C. Combinational Neural-Logic Circuit

To design the combinational neural-logic system, a truth table should be constructed first. Fig. 5 shows the plot of the averaged $\mathbf{c}_i$ values of 100 patterns for each speech commands. For a value greater than 0.6, it takes the logic H condition. For a value smaller than 0.4, it takes the logic L condition. For a value around 0.5 (arbitrarily set between 0.4 and 0.6), it takes a don't-care condition (X). The ith output $\hat{y}_i$ will take the logic H when the inputs are the corresponding ith Cantonese speech command (the order of the Cantonese speech commands are /soeng5/ (上), /haa5/ (下), /bat1/ (筆), /daai6/ (大), and /zyu3/ (註), characterized by $i = 1, 2, 3, 4,$ and 5, respectively.) For instance, referring to Fig. 5, the five coefficients $\mathbf{c}_i$ of the Cantonese speech command /soeng5/ (上) take the logic values of HHXXL. Its corresponding output $\hat{y}_1$ should take the logic condition H, whereas the logic conditions of other outputs should be L except those Cantonese speech commands that share the same input logic conditions. For the exceptional cases, the outputs will take the logic condition of "don't care." The truth table for the Cantonese speech commands is shown in Table IV. Taking the positive logic and with the help of Table IV and the K-map, the following output logic expressions can be obtained:

$$\hat{y}_1 = \hat{z}_1 \circ \hat{z}_2 \circ \bar{\hat{z}}_5 \tag{17}$$

$$\hat{y}_2 = \hat{z}_3 \circ \hat{z}_4 \circ \bar{\hat{z}}_5 \tag{18}$$

$$\hat{y}_3 = \hat{z}_1 \circ \hat{z}_4 \circ \bar{\hat{z}}_5 \tag{19}$$

$$\hat{y}_4 = \hat{z}_3 \circ \bar{\hat{z}}_5 \tag{20}$$

$$\hat{y}_5 = \bar{\hat{z}}_3 \circ \bar{\hat{z}}_5. \tag{21}$$

The combinational neural-logic circuit is shown in Fig. 6. The processing functions and the combination functions for all neural-logic gates are defined as follows:

$$\mathbf{h}(\mathbf{x}) = \frac{2\mathbf{x} - 1}{2} \in [-1 \quad 1] \tag{22}$$

$$f_{\text{AND}}(v_1, v_2) = f_{\text{OR}}(v_1, v_2) = \text{logsig}(v_1 + v_2) \tag{23}$$

where $\mathbf{x} = [x_1 \quad x_2]$ denotes the inputs of the neural-logic gates. Referring to (22) and (23), the defined functions are to map their inputs to be in the range of -1 to 1. Other functions could be used to serve the same purpose. In this application, they are chosen experimentally to produce a satisfactory result. Referring to Fig. 1, v_1 and v_2 denote the outputs of the rule base and neural network of the neural-logic gates, respectively; $\text{logsig}(u) = 1/1 + e^{-u}$, where u is a scalar, denotes the logarithmic sigmoid function. In this paper, the hidden node and output node transfer functions of the neural network of the neural-logic gates are the logarithmic sigmoid function and linear function, respectively. The linear function is defined as a function of which the inputs equal the outputs.

D. Denormalizer

The denormalizer is defined as follows:

$$\mathbf{y} = \frac{\bar{\mathbf{y}}}{\|\bar{\mathbf{y}}\|} \in [0 \quad 1]. \tag{24}$$

The function of (24) is to emphasize the largest value and suppress the small values of $\bar{\mathbf{y}}$. The index of the element of $\mathbf{y}$ having the maximum value indicates the possible input Cantonese speech.

TABLE IV
TRUTH TABLE FOR CANTONESE SPEECH COMMANDS

Inputs					Outputs					Inputs					Outputs				
$\hat{z}_1$	$\hat{z}_2$	$\hat{z}_3$	$\hat{z}_4$	$\hat{z}_5$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$	$\hat{y}_4$	$\hat{y}_5$	$\hat{z}_1$	$\hat{z}_2$	$\hat{z}_3$	$\hat{z}_4$	$\hat{z}_5$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$	$\hat{y}_4$	$\hat{y}_5$
L	L	L	L	L	X	X	X	X	X	H	L	L	L	L	X	X	X	X	X
L	L	L	L	H	X	X	X	X	X	H	L	L	L	H	X	X	X	X	X
L	L	L	H	L	X	X	X	X	X	H	L	L	H	L	X	X	X	X	X
L	L	L	H	H	X	X	X	X	X	H	L	L	H	H	X	X	X	X	X
L	L	H	L	L	L	L	L	H	L	H	L	H	L	L	L	L	L	H	L
L	L	H	L	H	X	X	X	X	X	H	L	H	L	H	X	X	X	X	X
L	L	H	H	L	L	H	H	H	L	H	L	H	H	L	L	H	L	H	L
L	L	H	H	H	X	X	X	X	X	H	L	H	H	H	X	X	X	X	X
L	H	L	L	L	X	X	X	X	X	H	H	L	L	L	H	L	L	L	H
L	H	L	L	H	X	X	X	X	X	H	H	L	L	H	X	X	X	X	X
L	H	L	H	L	X	X	X	X	X	H	H	L	H	L	H	L	L	L	H
L	H	L	H	H	X	X	X	X	X	H	H	L	H	H	X	X	X	X	X
L	H	H	L	L	L	L	L	H	L	H	H	H	L	L	H	L	L	H	L
L	H	H	L	H	X	X	X	X	X	H	H	H	L	H	X	X	X	X	X
L	H	H	H	L	L	H	H	H	L	H	H	H	H	L	H	H	L	H	L
L	H	H	H	H	X	X	X	X	X	H	H	H	H	H	X	X	X	X	X

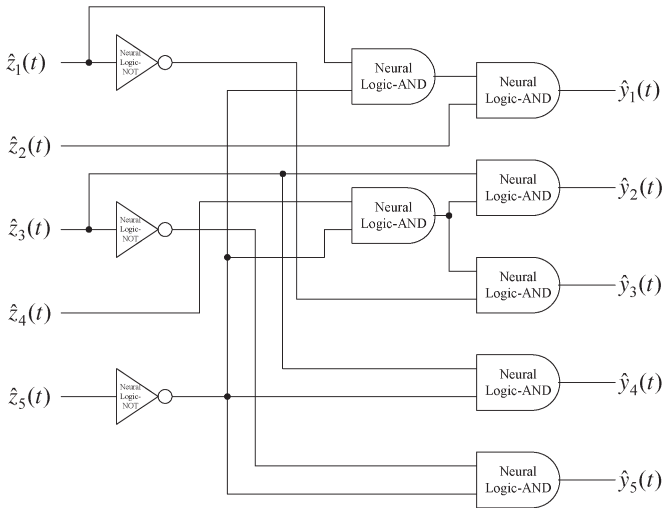

Fig. 6. Combinational neural-logic circuit for Cantonese speech command recognition.

E. Training of Combinational Neural-Logic System Using GA

After the design of the combinational neural-logic system for Cantonese speech command recognition, its parameters will be optimized by the real-coded GA with arithmetic crossover and nonuniform mutation [5]. The details of the training are as follows: For the GA training process, the following fitness function is defined:

$$\text{fitness} = \frac{1}{1 + \text{err}} \tag{25}$$

where err denotes the mean square error, i.e.,

$$\text{err} = \frac{\sum_{t=1}^{100} \left\| \mathbf{y}^d(t) - \mathbf{y}(t) \right\|^2}{5 \times 100} \tag{26}$$

with $\mathbf{y}^d(t) = [y_1^d(t) \quad y_2^d(t) \quad y_3^d(t) \quad y_4^d(t) \quad y_5^d(t)]$ denoting the desired output vector for the tth input vector. The objective is to maximize the fitness value. It should be noted that only one y_i is equal to 1 and the rest are all zero. For instance, when the input vector belongs to class 1, the corresponding output $\mathbf{y}^d(t) = [1 \quad 0 \quad 0 \quad 0 \quad 0]$; when the input vector belongs to class 2, the corresponding output $\mathbf{y}^d(t) = [0 \quad 1 \quad 0 \quad 0 \quad 0]$; and so on. As there are 500 training patterns (100 training patterns for each class) for training, the desired output vector for the first 100 training patterns will be $\mathbf{y}^d(t) = [1 \quad 0 \quad 0 \quad 0 \quad 0]$. For the second to fifth 100 training patterns, $\mathbf{y}^d(t) = [0 \quad 1 \quad 0 \quad 0 \quad 0], [0 \quad 0 \quad 1 \quad 0 \quad 0], [0 \quad 0 \quad 0 \quad 1 \quad 0]$, and $[0 \quad 0 \quad 0 \quad 0 \quad 1]$, respectively. All the parameters of the neural networks of the neural-logic gates will be taken as the genes to form the chromosome for the GA process.

The control parameters of the real-coded GA with arithmetic crossover and nonuniform mutation [5] are as follows: The probability of crossover is 0.8, the probability of mutation is 0.025, the shape parameter is 1, the population size is 40, and the number of training iteration is 2000. The GA training process will run for 30 times. Different numbers of hidden nodes for the neural network of the neural-logic gates are tried (three, five, eight, and 12 hidden nodes.) The lower and upper bounds of all parameters are -10 and 10, respectively. The best set of parameters among the 30 runs will be employed to implement the Cantonese speech command recognizer. The statistical results of the training are shown in Table V. It should be noted that the best set of parameters refers to the parameters that give the maximum fitness value among the 30 runs. In the following, the best trained combinational neural-logic system/neural network refers to that employing the best set of parameters.

F. Testing of the Trained Combinational Neural Network Logic System

One hundred fifty testing patterns (30 patterns for each Cantonese speech commands) are employed to test the recognition ability of the best trained combinational neural-logic system among the 30 runs. The testing fitness of the best trained combinational neural-logic system and the recognition accuracy under different numbers of hidden nodes are tabulated in Tables V and VI, respectively. From these two tables, it can be seen that the training and testing fitness values are higher

TABLE V
STATISTICAL TRAINING AND TESTING RESULTS OF THE COMBINATIONAL NEURAL-LOGIC SYSTEM. THE LEFT AND THE RIGHT VALUES ARE FOR THE TRAINING AND TESTING PATTERNS, RESPECTIVELY

	Training Patterns/Testing Patterns					Testing Patterns
n_h	Number of parameters	Maximum *fitness*	Minimum *fitness*	Mean *fitness*	Standard deviation	*Fitness* of the best-trained system
3	91	0.9892/0.9853	0.9837/0.9516	0.9873/0.9724	0.0014/0.0099	0.9801
5	147	0.9901/0.9861	0.9800/0.9834	0.9881/0.9797	0.0022/0.0062	0.9795
8	231	0.9912/0.9865	0.9787/0.9818	0.9893/0.9826	0.0025/0.0045	0.9834
12	343	0.9920/0.9874	0.9875/0.9669	0.9902/0.9824	0.0013/0.0042	0.9834

TABLE VI
RECOGNITION ACCURACY (IN PERCENT) OF THE BEST TRAINED COMBINATIONAL NEURAL-LOGIC SYSTEM

	Training Patterns					Testing Patterns				
n_h	/soeng5/ (上)	/haa5/ (下)	/bat1/ (筆)	/daai6/ (大)	/zyu3/ (註)	/soeng5/ (上)	/haa5/ (下)	/bat1/ (筆)	/daai6/ (大)	/zyu3/ (註)
3	100	97	96	95	100	100	93.3333	93.3333	93.3333	100
5	99	98	97	96	100	100	93.3333	93.3333	93.3333	100
8	100	99	98	96	100	100	96.6667	96.6667	93.3333	100
12	100	98	98	97	100	100	96.6667	96.6667	96.6667	100

TABLE VII
STATISTICAL TRAINING AND TESTING RESULTS OF THE TRADITIONAL THREE-LAYER FULLY CONNECTED FEEDFORWARD NEURAL NETWORK. THE LEFT AND THE RIGHT VALUES ARE FOR THE TRAINING AND TESTING PATTERNS, RESPECTIVELY

	Training Patterns/Testing Patterns					Testing Patterns
n_h	Number of parameters	Maximum *fitness*	Minimum *fitness*	Mean *fitness*	Standard deviation	*Fitness* of the best-trained system
4	106	0.9907/0.9843	0.9597/0.9112	0.9790/0.9591	0.0090/0.0231	0.9799
6	154	0.9913/0.9807	0.9663/0.9168	0.9869/0.9558	0.0059/0.0201	0.9803
10	250	0.9925/0.9867	0.9886/0.8977	0.9909/0.9694	0.0010/0.0166	0.9834
14	346	0.9936/0.9838	0.9885/0.9466	0.9914/0.9695	0.0013/0.0102	0.9788

TABLE VIII
RECOGNITION ACCURACY (IN PERCENT) OF THE BEST TRAINED TRADITIONAL THREE-LAYER FULLY CONNECTED FEEDFORWARD NEURAL NETWORK

	Training Patterns					Testing Patterns				
n_h	/soeng5/ (上)	/haa5/ (下)	/bat1/ (筆)	/daai6/ (大)	/zyu3/ (註)	/soeng5/ (上)	/haa5/ (下)	/bat1/ (筆)	/daai6/ (大)	/zyu3/ (註)
4	100	98	96	94	100	96.6667	93.3333	90	83.3333	96.6667
6	100	97	96	95	100	96.6667	96.6667	90	86.6667	100
10	100	98	97	94	100	100	96.6667	90	86.6667	100
14	100	98	98	95	100	100	96.6667	86.6667	90	100

for a larger number of hidden nodes. The training fitness values under different numbers of hidden nodes are all slightly higher than the testing fitness values. Similarly, the recognition is more accurate for a larger number of hidden nodes. On average, the recognition rate for each Cantonese speech command is over 90% for both training and testing under different numbers of hidden nodes.

G. Comparison

For comparison, a traditional three-layer fully connected feedforward neural network of (2) with five inputs and five outputs will be employed to implement the Cantonese speech recognizer. The input training and testing patterns of the traditional neural network are defined as $(2\mathbf{z} - 1)/2$ where $\mathbf{z}$ is the input patterns of the combinational neural-logic system. The desired training and testing output patterns are the same as those of the combinational neural-logic system. The real-coded GA with arithmetic crossover and nonuniform mutation is employed to train the parameters of the neural networks. For the GA process, the training environment is the same as that of the combinational neural-logic system. The training will also go through 30 times. To have a similar number of parameters to that of the combinational neural-logic system, the numbers of hidden nodes of the traditional neural network are chosen to be four, six, ten, and 14. Tables VII and VIII summarize the statistical training and testing results, as well as the recognition accuracy of the best trained traditional three-layer feedforward neural network.

From Tables V and VII, it can be seen that the fitness values of both approaches for the training and testing patterns are more or less the same. However, the proposed combinational neural-logic system exhibits more consistent training results as revealed by the standard deviation values. From Tables VI and VIII, it can be seen that the proposed combinational neural-logic system gives more accurate recognition results with the lowest recognition rate of 93.3333% for /haa5/ (下), /bat1/ (筆), and /daai6/ (大) and the highest recognition rate of 100% for /soeng5/ (上) and /zyu3/ (註). The traditional neural network approach gives recognition results with the lowest recognition rate of 83.3333% for /daai6/ (大). Referring to these tables, the combinational neural-logic system with the least number of network parameters performs better than that of the traditional neural network with different numbers of parameters in terms of recognition accuracy. These results illustrate the effective training and network performance provided by the proposed approach. Furthermore, a smaller network implies a simpler network structure.

V. Conclusion

The neural-logic-AND, -OR, and -NOT gates, which form the basic components of the proposed combinational neural-logic system, have been proposed. By constructing a truth table based on the knowledge of an application and with the help of K-map, the combinational neural-logic system formed by the neural-logic gates can be designed systematically to incorporate the characteristics of the application into the network structure to enhance the training and network performance. To show the merits of the proposed approach, a Cantonese speech recognizer has been implemented using the combinational neural-logic system.

References

[1] J. M. Zurada, *Introduction to Artificial Neural Systems*. Singapore: West Info Access, 1992.

[2] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[3] X. Yao, "Evolving artificial networks," *Proc. IEEE*, vol. 87, no. 7, pp. 1423–1447, 1999.

[4] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 694–713, May 1997.

[5] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*, 2nd ed. New York: Springer-Verlag, 1994.

[6] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 79–88, Jan. 2003.

[7] T. Yamakawa, T. Miki, and F. Ueno, "The design and fabrication of the current mode fuzzy logic semi-custom IC in the standard CMOS IC technology," in *Proc. IEEE ISMVL*, 1985, pp. 76–88.

[8] M. Togi and H. Watanabe, "A VLSI implementation of a fuzzy-inference engine. Toward an expert system on a chip," in *Proc. 2nd IEEE Int. Conf. AI and Appl.*, Washington, DC, 1985, pp. 192–197.

[9] E. Badreddin, "Analogical gates: A fuzzy operator approach for locomotion control of a non-holonomic mobile robot," in *Proc. Int. Conf. Fuzzy Syst. FUZZ-IEEE/IFES*, Yokohama, Japan, Mar. 20–24, 1995, pp. 875–882.

[10] ——, "Analogical gates as supervisory and direct controllers for a non-holonomic mobile robot," in *Proc. 5th IEEE Int. Conf. Fuzzy Syst.*, Sep. 8–11, 1996, vol. 3, pp. 1763–1769.

[11] S. Aly, "Fuzzy analogical gates for separation sequence synthesis," *Chem. Eng. Process.*, vol. 36, no. 3, pp. 209–217, Jun. 1997.

[12] K. Hirota and K. Ozawa, "The concept of fuzzy flip-flop," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 5, pp. 980–997, Sep./Oct. 1989.

[13] L. Gniewek and J. Kluska, "Family of fuzzy J-K flip-flops based on bounded product, bounded sum and complementation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 28, no. 6, pp. 861–868, Dec. 1998.

[14] J. Diamond, W. Pedrycz, and D. McLeod, "Fuzzy JK flip-flops as computational structures: Design and implementation," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 41, no. 3, pp. 215–226, Mar. 1994.

[15] K. Hirota and W. Pedrycz, "Design of fuzzy systems with fuzzy flip-flops," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 1, pp. 169–176, Jan. 1995.

[16] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 694–713, May 1997.

[17] R. S. Sandige, *Modern Digital Design*. New York: McGraw-Hill, 1990.

[18] [Online]. A Chinese syllabary pronounced according to the dialect of canton. Available: http://www.humanum.arts.cuhk.edu.hk/Lexis/Canton/

[19] K. F. Leung, F. H. F. Leung, H. K. Lam, and S. H. Ling, "On interpretation of graffiti digits and characters for eBooks: Neural-fuzzy network and genetic algorithm approach," *IEEE Trans. Ind. Electron.*, vol. 51, no. 2, pp. 464–471, Apr. 2004.

H. K. Lam (M'95) received the B.Eng. (Hons.) and Ph.D. degrees from the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong, in 1995 and 2000, respectively.

In 2000 and 2005, he was with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, as a Postdoctoral Fellow and a Research Fellow, respectively. In 2005, he joined as a Lecturer King's College London, London, U.K. His current research interests include intelligent control systems and computational intelligence.

Frank H. F. Leung (M'92–SM'03) was born in Hong Kong in 1964. He received the B.Eng. and Ph.D. degrees in electronic engineering from the Hong Kong Polytechnic University, Kowloon, Hong Kong, in 1988 and 1992, respectively.

In 1992, he joined the Hong Kong Polytechnic University, where he is currently an Associate Professor with the Department of Electronic and Information Engineering. He is active in research and has published more than 130 research papers on computational intelligence, control, and power electronics. At present, he is involved in R&D on intelligent multimedia home and powerline communications.

Prof. Leung is a Corporate Member of the Institution of Electrical Engineers, U.K. He has been serving as a Reviewer for many international journals and has been helping organize many international conferences. He is currently an Executive Committee Member of the IEEE Hong Kong Chapter of Signal Processing. He is also a Chartered Engineer.