A Genetic Algorithm Based Variable Structure Neural Network

S.H. Ling, H.K. Lam, F.H.F. Leung, and Y.S. Lee

Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, the Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Abstract – **This paper presents a neural network model with a variable structure, which is trained by genetic algorithm (GA). The proposed neural network consists of a Neural Network with a Node-to-Node Relationship (N^4R) and a Network Switch Controller (NSC). In the N^4R, a modified neuron model with two activation functions in the hidden layer, and switches in its links are introduced. The NSC controls the switches in the N^4R. The proposed neural network can model different input patterns with variable network structures. The proposed neural network provides better result and learning ability than traditional feed forward neural networks. Two application examples on XOR problem and hand-written pattern recognition are given to illustrate the merits of the proposed network.**

Keywords -- **Genetic algorithm, neural network, hand-written pattern recognition.**

I. INTRODUCTION

Neural networks [1] mimic the biological information processing mechanism. They are typically designed to perform a non-linear mapping from a set of inputs to a set of outputs. As a neural network can autonomously develop operational capabilities in response to an information environment, it can learn from experience and generalize from previous examples, and is ideal for cases where the required mapping algorithm is not known and tolerance to faulty input information is required. A 3-layer feed-forward neural network can approximate any nonlinear continuous function to an arbitrary accuracy. Neural networks are widely applied in areas such as prediction [3], system modeling and control [1]. Algorithms such as Genetic Algorithm (GA) [5, 9] and back propagation [8] can be used to train neural networks. Traditionally, a feed-forward neural network [2] has 3 layers (input, hidden and output layers) of nodes connected in a layer-to-layer manner. In many applications, neural networks are fully connected. However, the performance of a fully connected neural network may not be better than that of a partly connected one with the same number of hidden nodes [3]. It is because some links in a neural network could be redundant.

In general, there are two classes of training algorithms, local and global search algorithms. Gradient descent techniques, e.g. the back-propagation algorithm [1], are kinds of local search algorithms for local optima. They are good for simple functions, e.g. convex or concave functions. GA [8, 10], simulated annealing [8] and Tabu search [8] are examples of global search algorithms for global optima over a defined domain. Global search techniques are good for optimizing some complex functions. In this paper, GA will be employed as the algorithm for tuning the parameter values of a proposed variable structure neural network.

GA is a directed random search technique [9] that is widely applied in optimization problems [8-10]. This is especially useful for complex optimization problems where the number of parameters is large and the analytical solutions are difficult to obtain. It can help to find the globally optimal solution over a domain [8-10], and has been applied in different areas such as fuzzy control [7, 12], forecasting [3-4], path planning [11], and classification [6] etc.

In this paper, a GA based variable structure neural network is proposed. The proposed variable structure neural network consists of a Neural Network with a Node-to-Node Relationship (N^4R) and a Network Switch Controller (NSC). In the N^4R, two different activation functions are used in the neurons of the hidden layer, and a node-to-node relationship is introduced. By introducing the proposed neuron in the network, the space of the search domain of a non-linear function will be increased. Switches are introduced in some links of the N^4R to facilitate the tuning of the network structure. The on/off states of the link switches are controlled by the NSC, which is a 3-layer feed-forward neural network. For an application with a large domain of input-output mapping, a variable structure neural network (due to the link switches) should be able to perform the mapping task in a more efficient way in terms of accuracy and network complexity (number of parameters). In practice, a variable structure neural network divides the input-output domain into several sub-domains, and different neural network structures are used to perform the mapping task for different input-output sub-domains. It can handle different input patterns through an adaptive network structure, and exhibit a better performance. Two examples are used to test the proposed GA based variable structure neural network, and good results are obtained.

This paper is organized as follows. In section II, the proposed variable structure neural network is presented. In section III, training of the parameters of the proposed variable structure neural network using GA will be presented. In section IV, two application examples will be given. A conclusion will be drawn in session V.

II. VARIABLE STRUCTURE NEURAL NETWORK MODEL

A block diagram of the proposed variable structure neural network is shown in Fig. 1. It consists of two parts, namely a Neural Network with a Node-to-Node Relationship (N^4R) and a Network Switch Controller (NSC). In the N^4R, each neuron

0-7803-7906-3/03/$17.00 ©2003 IEEE.

in the hidden layer has two activation functions called the static activation function (SAF) and dynamic activation function (DAF). Some switches are introduced in the some or all links between the hidden layer and the output layer. The action of the switches is controlled by the NSC, which depends on the network inputs.

A. Neural Network with a Node-to-Node Relationship (N^4R)

Fig. 2 shows the proposed neuron in the hidden layer of the N^4R. The SAF and DAF govern the input-output relationships of the neuron. With this proposed neuron, the connection of N^4R is shown in Fig. 1, which is a three-layer neural network. The first layer is the input layer, which simply distributes the inputs. The second layer is the hidden layer in which the SAF determines hyper-planes as switching surfaces. It selects convex regions in the input spaces for firing. Owing to the DAF, the upper neighbor's output concerns the bias term while the lower neighbor's output influences the sharpness of the edges of the hyper-planes. They eventually combine into convex regions by the output layer. By introducing, a node-to-node relationship in the hidden layer, the space of the searching domain can be increased. Switches are also introduced in the network links between the hidden layer and the output layer so that the structure of the network is variable. Effectively, this divides the search domain into many sub-domains. Different structures of neural networks can be used to perform the mapping task for different input-output sub-domains.

1) Proposed neuron model: For the SAF, let v_{ij} be the synaptic connection weight from the i-th input node z_i to the j-th neuron. The output κ_j of the j-th neuron's SAF is given by,

$$\kappa_j = net_s^j\left(\sum_{i=1}^{n_{in}} z_i v_{ij}\right),\ i = 1, 2, \ldots, n_{in};\ j = 1, 2, \ldots, n_h, \quad (1)$$

where n_{in} denotes the number of input and $net_s^j(\cdot)$ is a static activation function, which is given by,

$$net_s^j\left(\sum_{i=1}^{n_{in}} z_i v_{ij}\right) = \begin{cases} e^{\frac{-\left(\sum_{i=1}^{n_{in}} z_i v_{ij} - m_s^j\right)^2}{2\sigma_s^{j\,2}}} - 1 & \text{if } \sum_{i=1}^{n_{in}} z_i v_{ij} \le m_s^j \\ 1 - e^{\frac{-\left(\sum_{i=1}^{n_{in}} z_i v_{ij} - m_s^j\right)^2}{2\sigma_s^{j\,2}}} & \text{otherwise} \end{cases},$$

$$j = 1, 2, \ldots, n_h, \quad (2)$$

where m_s^j and σ_s^j are the static mean and static standard deviation for the j-th SAF respectively. These parameters (m_s^j and σ_s^j) are fixed after the training process. The parameter m_s^j affects the bias of the activation function and σ_s^j influences the sharpness of the activation function.

For the DAF, the neuron output ζ_j of the j-th neuron is given by,

$$\zeta_j = net_d^j(\kappa_j, m_d^j, \sigma_d^j),\ j = 1, 2, \ldots, n_h \text{ and} \quad (3)$$

$$net_d^j(\kappa_j, m_d^j, \sigma_d^j) = \begin{cases} e^{\frac{-(\kappa_j - m_d^j)^2}{2\sigma_d^{j\,2}}} - 1 & \text{if } \kappa_j \le m_d^j \\ 1 - e^{\frac{-(\kappa_j - m_d^j)^2}{2\sigma_d^{j\,2}}} & \text{otherwise} \end{cases}, \quad (4)$$

where

$$m_d^j = p_{j+1,j} \times \kappa_{j+1}, \quad (5)$$

$$\sigma_d^j = p_{j-1,j} \times \kappa_{j-1}. \quad (6)$$

m_d^j and σ_d^j are the dynamic mean and dynamic standard deviation of the j-th DAF respectively, κ_{j-1} and κ_{j+1} represent the SAF's outputs of the $j-1$ -th and $j+1$ -th neurons respectively, $p_{j+1,j}$ denotes the weight of the link between the j+1-th node and the j-th node, and $p_{j-1,j}$ denotes the weight of the link between the $j-1$ -th node and the j-th node. It should be noted from Fig. 1 that if $j = 1$, $p_{j-1,j}$ is equal to $p_{n_h,j}$; and if $j = n_h$, $p_{j+1,j}$ is equal to $p_{1,j}$.

Unlike the SAF, the DAF is dynamic as its parameters depend on the outputs of the $j-1$ -th and $j+1$ -th neurons. Referring to (3) – (6), the input-output relationship of the proposed neuron is as follows:

$$\zeta_j = net_d^j\left(net_s^j\left(\sum_{i=1}^{n_{in}} z_i v_{ij}\right), m_d^j, \sigma_d^j\right),\ j = 1, 2, \ldots, n_h. \quad (7)$$

2) Connection of the N^4R: The proposed neural network has three layers with n_{in} nodes in the input layer, n_h nodes in the hidden layer, and n_{out} nodes in the output layer. In the hidden layer, the neuron model presented in the previous sub-section is employed. A node-to-node relationship is introduced in the hidden layer. In the output layer, a static activation function is employed. The link switches between the hidden layer and the output layer is realized, as a unit step function is follows,

$$\delta(\alpha) = \begin{cases} 0 \text{ if } \alpha < 0 \\ 1 \text{ if } \alpha \ge 0 \end{cases}, \alpha \in \Re. \quad (8)$$

In this proposed variable structure neural network, the number of switches (n_{sw}) is variable, and is given by,

$$n_{sw} = k_f \times n_h \times n_{out}, \quad (9)$$

where $k_f \in (0\ \ 1]$ is a constant governing the number of switches used in the network, e.g. $k_f = 0.5$ means 50% of the links have switches, $k_f = 1$ means all links have switches. Owing to the link switches in the network, the output of the N^4R (Fig. 3) is given by,

$$y_l = net_o^l\left(\sum_{j=1}^{n_h} \zeta_j'(\zeta_j, \beta_{jl}, w_{jl})\right),\ l = 1, 2, \ldots, n_{out} \quad (10)$$

$$= net_o^l\left(\sum_{j=1}^{n_h}\zeta_j'\left(net_d^j\left(net_s^j\left(\sum_{i=1}^{n_{in}} z_i v_{ij}\right), m_d^j, \sigma_d^j\right), \beta_{jl}, w_{jl}\right)\right), \quad (11)$$

where $w_{jl}, j = 1, 2, \ldots, n_h$; $l = 1, 2, \ldots n_{out}$ denotes the weight of the link between the j-th hidden and the l-th output nodes; β_{jl} denotes the on/off states of the switches. Referring to Fig. 1, let

$$\begin{bmatrix} s_{11} & s_{21} & \cdots & s_{n_h 1} \\ s_{12} & s_{22} & \cdots & s_{n_h 2} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1n_{out}} & s_{2n_{out}} & \cdots & s_{n_h n_{out}} \end{bmatrix} = \begin{bmatrix} \varphi_1 & \varphi_2 & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \varphi_{n_{sout}-1} & \varphi_{n_{sout}} \end{bmatrix},$$

$$\beta_{jl} = \begin{bmatrix} \delta(s_{11}) & \delta(s_{21}) & \cdots & \delta(s_{n_h 1}) \\ \delta(s_{12}) & \delta(s_{22}) & \cdots & \delta(s_{n_h 2}) \\ \vdots & \vdots & \ddots & \vdots \\ \delta(s_{1n_{out}}) & \delta(s_{2n_{out}}) & \cdots & \delta(s_{n_h n_{out}}) \end{bmatrix}, \quad (12)$$

where s_{jl} denotes the parameter of the link switch between the j-th hidden and the l-th output nodes; φ_h, h = 1, 2, ..., n_{sout}, denotes the outputs of the NSC; n_{sout} denotes the number of output of the NSC. If a link has no switch, the corresponding element $\delta(\cdot)$ in (12) is equal to 1. For instance, when $k_f = 0.5$ (half of the links have switches), n_h =4 and n_{out} =3, $n_{sw} = 0.5\times4\times3 = 6$ and β_{jl} can be given by:

$$\beta_{jl} = \begin{bmatrix} \delta(s_{11}) & \delta(s_{21}) & 1 & 1 \\ \delta(s_{12}) & \delta(s_{22}) & 1 & 1 \\ \delta(s_{13}) & \delta(s_{23}) & 1 & 1 \end{bmatrix}. \quad (13)$$

$net_o^l(\cdot)$ denotes the activation function of the output neuron:

$$net_o^l\left(\sum_{j=1}^{n_h}\zeta_j'(\zeta_j, w_{jl}, \beta_{jl})\right) =$$

$$\begin{cases} e^{\frac{-\left(\sum_{j=1}^{n_h}\zeta_j(w_{jl}\beta_{jl})-m_o^l\right)^2}{2\sigma_o^{l\,2}}} - 1 & \text{if } \sum_{j=1}^{n_h}\zeta_j(w_{jl}\beta_{jl}) \le m_o^l \\ 1 - e^{\frac{-\left(\sum_{j=1}^{n_h}\zeta_j(w_{jl}\beta_{jl})-m_o^l\right)^2}{2\sigma_o^{l\,2}}} & \text{otherwise} \end{cases} \quad (14)$$

where m_o^l and σ_o^l are the mean and the standard deviation of the output node activation function respectively. The parameters of this N^{4}R can be trained by GA, and the values of the link switches are obtained by the NSC.

B. Network Switch Controller (NSC)

In the N^{4}R, the switch parameters φ_{jl} are provided by a Network Switch Controller (NSC), which is a 3-layer feed-forward neural network. As shown in Fig. 1, the input-output relationship of the NSC is given by,

$$\varphi_h = \text{tansig}\left(\sum_{g=1}^{n_{sh}} u_{gh}\,\text{tansig}\left(\sum_{i=1}^{n_{in}} t_{ig} z_i - b_g^1\right) - b_h^2\right), \; h = 1, 2, \ldots, n_{sout}, \quad (15)$$

where z_i, i = 1, 2, ..., n_{in} are the input variables; n_{in} denotes the number of inputs; n_{sh} denotes the number of hidden nodes; t_{ig}, g = 1, 2, ..., n_{sh}, denotes the weight of the link between the i-th input and the g-th hidden node; u_{gh}, denotes the weight of the link between the g-th hidden and the h-th output nodes; b_g^1 and b_h^2 denote the biases for the hidden nodes and output nodes respectively; tansig(·) denotes the hyperbolic tangent sigmoid function:

$$\text{tansig}(\alpha) = \frac{2}{1+e^{-2\alpha}} - 1, \; \alpha \in \Re, \quad (16)$$

φ_h is the h-th output of the NSC. All the parameters of the NSC are tuned by GA.

III. Training of Parameters

The proposed variable structure neural network is employed to learn the input-output relationship of an application using GA with arithmetic crossover and non-uniform mutation [10]. A population of chromosomes P is initialized and then evolves. First, two parents are selected from P by the method of spinning the roulette wheel [10]. Then a new offspring is generated from these parents using the crossover and mutation operations, which are governed by the probabilities of crossover and mutation respectively. These probabilities are chosen by trial and error through experiments for good performance. The new population thus generated replaces the current population. These procedures are repeated until a certain termination condition is satisfied, e.g. a predefined number of generations has been reached. The input-output relationship can be described by,

$$\mathbf{y}^d(t) = \mathbf{g}\left(\mathbf{z}^d(t)\right), \; t = 1, 2, \ldots, n_d, \quad (17)$$

where $\mathbf{z}^d(t) = \begin{bmatrix} z_1^d(t) & z_2^d(t) & \cdots & z_{n_{in}}^d(t) \end{bmatrix}$ and $\mathbf{y}^d(t) = \begin{bmatrix} y_1^d(t) & y_2^d(t) & \cdots & y_{n_{out}}^d(t) \end{bmatrix}$ are the given inputs and the desired outputs of an unknown nonlinear function $\mathbf{g}(\cdot)$ respectively; n_d denotes the number of input-output data pairs. The fitness function is defined as,

$$fitness = \frac{1}{1+err}, \quad (18)$$

$$err = \frac{\sum_{t=1}^{n_d}\sum_{k=1}^{n_{out}}\left|y_k^d(t) - y_k(t)\right|}{n_d n_{out}}. \quad (19)$$

The objective is to maximize the fitness value of (18) (minimize *err* of (19)) by setting the chromosome to be $\begin{bmatrix} v_{ij} & w_{jl} & m_s^j & \sigma_s^j & p_{j+1,j} & p_{j-1,j} & m_o^l & \sigma_o^l & t_{ig} & u_{gh} & b_g^1 & b_h^2 \end{bmatrix}$ for all g, h, i, j, l. In this paper, $v_{ij}, w_{jl}, p_{j+1,j}, p_{j-1,j}, t_{ig}, u_{gh}, b_g^1, b_h^2 \in \begin{bmatrix} -1 & 1 \end{bmatrix}$,

$m_s^j, m_o^l \in [-0.5 \quad 0.5]$ and $\sigma_s^j, \sigma_o^l \in [0.01 \quad 0.5]$. The range of the fitness value of (45) is [0, 1].

IV. Application Examples

Two application examples will be given in this section to illustrate the merits of the proposed GA-based neural network.

A. XOR Problem

A three-input XOR function, which is not linearly separable, has the following input-output relationship:

$$\begin{cases}(-1,-1,-1),(-1,+1,+1),(+1,-1,+1),(+1,+1,-1) \rightarrow -1 \\ (-1,-1,+1),(-1,+1,-1),(+1,-1,-1),(+1,+1,+1) \rightarrow +1\end{cases} \quad (20)$$

The three inputs of the proposed neural network are defined as $z_i(t)$, i = 1, 2, 3 and $y(t)$ is the network output. The number of hidden nodes (n_h) in the N^{4}R is set at 3, the number of hidden nodes in the NSC (n_{sh}) is set at 1 and the constant factor (k_f) is set at 1/3. The total number of training parameters is 32. This network structure is chosen by trial and error through experiments for good performance. Referring to (11), the proposed variable structure neural network is governed by,

$$y = net_o^l\left(\sum_{j=1}^{4}\zeta_j'\left(net_d^j\left(net_s^j\left(\sum_{i=1}^{3} z_i v_{ij}\right), m_d^j, \sigma_d^j\right), \beta_{jl}, w_{jl}\right)\right) \quad (21)$$

The fitness function is defined as follows,

$$fitness = \frac{1}{1+err} \quad (22)$$

$$err = \frac{\sum_{t=1}^{8}\left|y^d(t) - y(t)\right|}{8} \quad (23)$$

Referring to (9), with k_f = 1/3, the number of switches (n_{sw}) is equal to 1. We set β_1 to be provided by the NSC, and β_2 and β_3 to be equal to 1. GA is employed to tune the parameters of the proposed variable structure neural network of (21). The objective is to maximize the fitness function of (22). The best fitness value is 1 and the worst one is 0. The population size used for the GA is 10. The chromosome used is $\left[v_{ij} \quad w_{jl} \quad m_s^j \quad \sigma_s^j \quad p_{j+1,j} \quad p_{j-1,j} \quad m_o^l \quad \sigma_o^l \quad t_{il} \quad u_{il} \quad b_i^1 \quad b_l^2\right]$ for all i, j. The initial values of the parameters of the proposed network are randomly generated. For comparison purpose, a traditional feed-forward neural network trained by the same GA is applied to the XOR problem. The number of hidden node of the traditional neural network is 6 so that the total number of parameter is 31, which is very near to the number of parameters of the proposed network. The number of iteration for training the two neural networks is 500, and all results are averaged ones out of 30 runs. For the GA, the probabilities of crossover and mutation are set at 0.6 and 0.5 respectively. The shape parameter of mutation is set at 0.1. The results in terms of fitness value are tabulated in Table I. It can be seen that the performance of the proposed network is better than that of the traditional one. The on/off state of the switch for each pattern is tabulated in Table II. In this table, it can be seen that the structure of the proposed neural network is adaptively changed with respect to the input pattern. The best network outputs of the XOR problem using the proposed approach and the traditional approach are tabulated in Table III.

B. Hand-written Pattern Recognition

In this section, an application on hand-written graffiti pattern recognition will be presented. Numbers are assigned to pixels on a two-dimensional plane, and 10 normalized numbers are used to characterize the 10 uniformly sampled pixels of a given graffiti. A 10-input-3-output network is used. The ten inputs nodes, z_i, i = 1, 2, ..., 10 are the numbers representing the graffiti pattern. Three standard patterns are to be recognized: rectangles, triangles and straight lines (Fig. 4). We use 300 sets of 10 normalized sample-points for each pattern to train the neural network. Hence, we have 900 sets of data for training. The three outputs, $y_l(t)$, l = 1, 2, 3 indicate the similarity between the input pattern and the three standard patterns respectively. The desired outputs of the pattern recognition system are $\mathbf{y}(t) = [1 \quad 0 \quad 0]$, $\mathbf{y}(t) = [0 \quad 1 \quad 0]$ and $\mathbf{y}(t) = [0 \quad 0 \quad 1]$ for rectangles, triangles and straight lines respectively. After training, a larger value of $y_l(t)$ implies that the input pattern matches more closely to the corresponding graffiti pattern. For instance, a large value of $y_1(t)$ implies that the input pattern is near to a rectangle. Referring to (11), the proposed network used for the pattern recognition is governed by,

$$y_l = net_o^l\left(\sum_{j=1}^{8}\zeta_j'\left(net_d^j\left(net_s^j\left(\sum_{i=1}^{10} z_i v_{ij}\right), m_d^j, \sigma_d^j\right), \beta_{jl} w_{jl}\right)\right), \quad l = 1, 2, 3. \quad (24)$$

GA is employed to tune the parameters of the proposed network of (24). The fitness function is defined as follows,

$$fitness = \frac{1}{1+err}, \quad (25)$$

$$err = \frac{\sum_{t=1}^{300}\sum_{k=1}^{3}\left(\left(\frac{y_k(t)}{\|\mathbf{y}(t)\|}\right)^2 - \left(\frac{y_k^d(t)}{\|\mathbf{y}^d(t)\|}\right)^2\right)}{300\times 3}. \quad (26)$$

The value of *err* indicates the mean square error (MSE) of the recognition system. In the proposed network, the number of hidden nodes (n_h) in the N^{4}R is set at 8, the number of hidden node in the NSC (n_{sh}) is set at 3 (the constant factor k_f is set at 0.125.) The total number of parameters is therefore 187. The network structure is chosen by trial and error through experiments for good performance. GA is employed to tune the parameters of the proposed neural network of (24). The population size used for the GA is 10. The chromosome used is $\left[v_{ij} \quad w_{jl} \quad m_s^j \quad \sigma_s^j \quad p_{j+1,j} \quad p_{j-1,j} \quad m_o^l \quad \sigma_o^l \quad sv_{ig} \quad sw_{gh} \quad b_g^1 \quad b_h^2\right]$ for all g, h, i, j, l. The initial values of the parameters of the neural network are randomly generated. For comparison, a

traditional 3-layer neural network trained by the GA is also used to recognize the patterns. The number of hidden nodes of the traditional neural network is 16 (so that the total number of parameters is 227, which is even larger than that of the proposed network.) The number of iteration to train the two neural networks is 2000, and all results are averaged ones of 25 runs. For the GA, the probabilities of crossover and mutation are set at 0.8 and 0.2 respectively. The shape parameter of mutation is set at 1 for both networks.

After training, we use 600 (200×3) sets of data for testing. The results are tabulated in Table IV. From this Table, it can be seen that the average training errors (governed by (26)) and the average forecasting errors (governed by $err = \frac{\sum_{t=1}^{200}\sum_{k=1}^{3}\left(\left(\frac{y_k(t)}{\|\mathbf{y}(t)\|}\right)^2 - \left(\frac{y_k^d(t)}{\|\mathbf{y}^d(t)\|}\right)^2\right)}{200\times 3}$) of the proposed network are smaller. The recognition accuracy is governed by

$$\text{Accuracy} = \left(\frac{n_{recognized}}{n_{testing}}\right)\times 100\%, \qquad (27)$$

where $n_{testing}$ and $n_{recognized}$ are the total number of testing patterns and the number of successfully recognized patterns respectively. The accuracy of the proposed network is also better.

V. Conclusion

A GA based variable structure neural network has been proposed. It consists of a Neural Network with a Node-to-Node Relationship (N^4R) and a Network Switch Controller (NSC). In the N^4R, a modified neuron model with two activation functions in the hidden layer, and link switches between the hidden and output layers have been introduced. With the NSC controlling the switches in the N^4R, the proposed neural network can model different input patterns with different network structures. By employing this neuron model and the variable structure, the performance of the proposed network is found to be better than that of the traditional neural network. All parameters of the proposed neural network are tuned by GA. Examples of an XOR problem and pattern recognition have been given. The performance of the proposed network in these examples is good.

VI. Acknowledgement

The work described in this paper was substantially supported by a grant from the Research Grants Council of the Hong Kong Special Administration Region, China (Project Nos. PolyU 5098/01E).

VII. References

[1] M. Brown and C. Harris, *Neural Fuzzy Adaptive Modeling and Control.* Prentice Hall, 1994.

[2] D.E. Rumelhrt and J.L. McClelland, *Parallel Distributed Processing: Explorations in Microstructure of Cognition,* vol. 1. Cambridge, MA:MIT Press, 1986.

[3] 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 Networks,* vol.14, no. 1, pp.79-88, Jan. 2003.

[4] S.H. Ling, F.H.F. Leung, H.K. Lam, Y.S. Lee, and P.K.S. Tam " A novel GA-based neural network for short-term load forecasting," *IEEE Trans. Industrial Electron.*, to be published.

[5] S.H. Ling, H.K. Lam, F.H.F. Leung, and P.K.S. Tam," Learning of neural network parameters using fuzzy genetic algorithm," in *Proc. Congress on Evolutionary Computation (CEC2002), Hawaii*, May 2002, pp. 1928-1933.

[6] K.F. Leung, F.H.F. Leung, H.K. Lam, and S.H. Ling, "On interpretation of graffiti digits and commands for eBooks: neural-fuzzy network and genetic algorithm Approach," *IEEE Trans. Industrial Electron.*, to be published.

[7] F.H.F Leung, H.K. Lam, S.H. Ling, and P.K.S. Tam, "Optimal and stable fuzzy controllers for nonlinear systems using an improved genetic algorithm" *IEEE Trans. Industrial Electron.*, to be published.

[8] D.T. Pham and D. Karaboga, *Intelligent Optimization Techniques, Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks.* Springer, 2000.

[9] J.H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press, MI, 1975.

[10] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs,* 2nd extended ed. Springer-Verlag, 1994.

[11] H. Juidette and H. Youlal, "Fuzzy dynamic path planning using genetic algorithms," *Electronics Letters*, vol. 36, no. 4, pp. 374-376, Feb. 2000.

[12] H.K. Lam, F.H.F. Leung, and P.K.S. Tam, "Design and stability analysis of fuzzy model based nonlinear controller for nonlinear systems using genetic algorithm," *IEEE Trans. Syst., Man and Cybern, Part B: Cybernetics,* to be published.

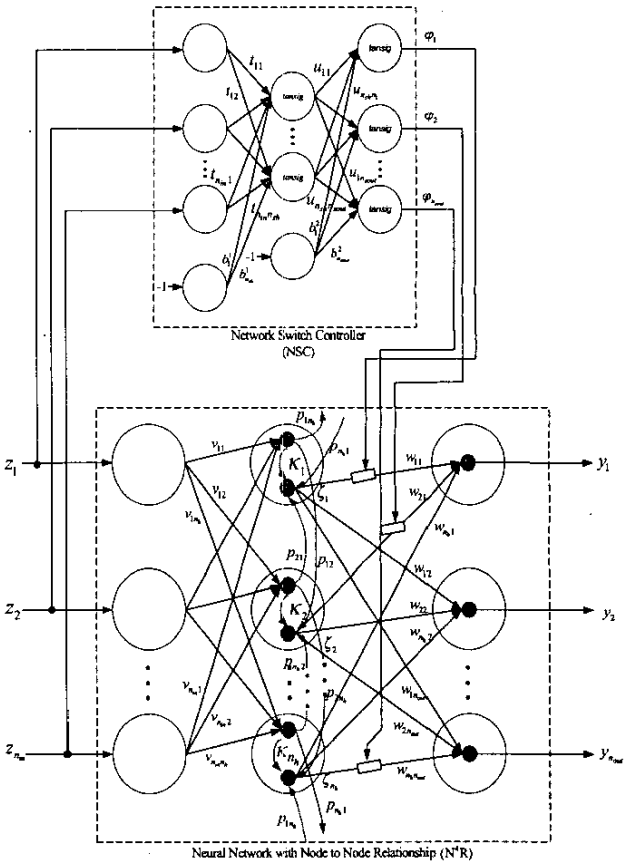

Fig. 1. Variable Structure Neural Network Model

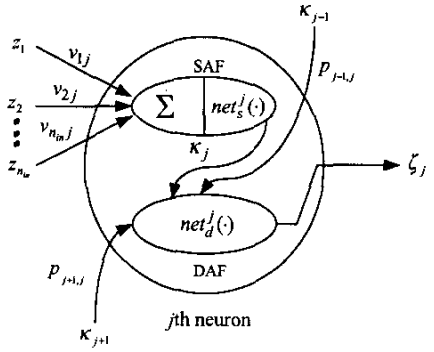

Fig. 2. Proposed modified neuron.

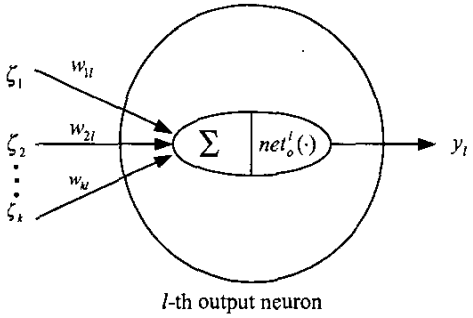

Fig. 3. Model of the proposed output neuron.

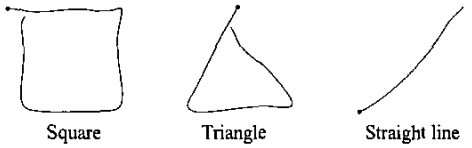

Fig. 4. Samples of the hand-written patterns.

TABLE I
RESULTS OF THE PROPOSED NEURAL NETWORK AND THE TRADITIONAL NEURAL NETWORK FOR THE XOR PROBLEM

	Proposed Network	Traditional Network
Average fitness value:	0.999	0.9888
Maximum fitness value:	1.0000	0.9933
Minimum fitness value:	0.9980	0.9609
Standard deviation:	0.0003	0.0063
Number of parameter:	32	31

TABLE II
THE ON/OFF STATE OF THE SWITCH FOR DIFFERENT INPUT PATTERNS

Input pattern			On/off state (β)
x_1	x_2	x_3	β_1
-1	-1	-1	0
-1	-1	+1	0
-1	+1	-1	0
-1	+1	+1	0
+1	-1	-1	0
+1	-1	+1	1
+1	+1	-1	0
+1	+1	+1	1

TABLE III
THE BEST OUTPUT OF THE PROPOSED AND TRADITIONAL NETWORKS FOR THE XOR PROBLEM

Input			Output		
x_1	x_2	x_3	y^d	y (Proposed algorithm)	$y_{(\cdot)}$ (Traditional algorithm)
-1	-1	-1	-1	-1.0000	-0.9949
-1	-1	+1	+1	+1.0000	+0.9792
-1	+1	-1	+1	+1.0000	-0.9921
-1	+1	+1	-1	-1.0000	-0.9938
+1	-1	-1	+1	+1.0000	+0.9978
+1	-1	+1	-1	-1.0000	-0.9921
+1	+1	-1	-1	-1.0000	-0.9969
+1	+1	+1	+1	+1.0000	+0.9987

TABLE IV
RESULTS OF THE PROPOSED NEURAL NETWORK AND THE TRADITIONAL NEURAL NETWORK FOR HAND-WRITTEN PATTERN RECOGNITION

	Proposed Network	Traditional Network
Average fitness value:	0.9845	0.9689
Average training error (MSE):	0.0157	0.0321
Average forecasting error (MSE):	0.0237	0.0501
Number of parameter:	187	227
Recognition accuracy:	95.50%	92.33%