A Genetic Algorithm Based Neural-Tuned 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-tuned neural network, which is trained by genetic algorithm (GA). The neural-tuned neural network consists of a neural network and a modified neural network. In the modified neural network, a neuron model with two activation functions is introduced. Some parameters of these activation functions will be tuned by neural network. The proposed network structure can increase the search space of the network and gives better performance than traditional feed-forward neural networks. Some application examples are given to illustrate the merits of the proposed network.**

***Index Terms*—Genetic Algorithm, Neural Network, Pattern Recognition, Sunspot Forecasting.**

I. Introduction

Neural network was proved to be a universal approximator [1]. 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, 8], system modeling and control [1]. Owing to its particular structure, a neural network is good at learning [9] using some algorithms such as the genetic algorithm (GA) [5, 10] and back propagation [9]. Traditionally, a feed-forward neural network [2] has 3 layers (input, hidden and output layers) of nodes.

GA is a directed random search technique [10] that is widely applied in optimization problems [9-11]. GA can help to find out the globally optimal solution over a domain [9-11]. It has been applied in different areas such as fuzzy control [7, 12-13], path planning [14], greenhouse climate control [15], modeling and classification [6] etc.

In this paper, a neural-tuned neural network (NTNN), which consists of a traditional neural network (TNN) [2] and a modified neural network (MNN) [4], is proposed. For the modified neural network, a new neuron structure is used in the hidden layer. Two different activation functions are used in the proposed neurons. Some parameters of these activation functions will be tuned by the TNN. By introducing the proposed neuron, the degree of freedom of the network parameters will be increased. Comparing with the traditional feed-forward neural network [2], the proposed neural-tuned neural network can give a better performance in terms of accuracy and convergence rate. In this paper, all parameters of the neural network are trained by GA with arithmetic crossover and non-uniform mutation [10, 16-18]. Two application examples are used to test the proposed network and good results are obtained.

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

II. Neural-Tuned Neural Network

The block diagram of the proposed NTNN is shown in Fig. 1. It consists of a modified neural network (MNN) [4] and a traditional neural network (TNN) [2]. In the MNN, each neuron in the hidden layer has two activation functions called static activation function (SAF) and dynamic activation function (DAF).

A. Modified Neural Network

Fig. 2 shows the proposed neuron with an SAF and a DAF inside. The parameters of the SAF are fixed, and its output depends on the inputs of the neuron. For the DAF, the parameters depend on the outputs of the TNN. Its input is the output of the SAF. With this proposed neuron, the connection of the MNN is shown in Fig. 3, which is a three-layer neural network. In this figure, $\mathbf{p}^U = [p_1^U, \ p_2^U, \ \ldots \ p_{n_h}^U]$ and $\mathbf{p}^L = [p_1^L, \ p_2^L, \ \ldots \ p_{n_h}^L]$ are provided by the TNN, where n_h denotes the number of hidden nodes.

1) Proposed neuron model: For the SAF, let v_{ik} be the synaptic connection weight from the i-th input node z_i to the k-th neuron. The output κ_k of the k-th neuron's SAF is defined as,

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

where n_{in} denotes the number of inputs, n_h denotes the number of hidden nodes and $net_s^k(\cdot)$ is an SAF defined as:

0-7803-7906-3/03/$17.00 ©2003 IEEE.

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

where m_s^k and σ_s^k are the static mean and static standard deviation for the k-th SAF respectively. The parameters m_s^k and σ_s^k are fixed after the training process. By using the proposed activation function in (2), the output value is ranged from -1 to 1. The shapes of the SAFs are shown in Fig. 4. The static mean is used to control the bias as shown in Fig. 4a and the static standard deviation influences the sharpness as shown in Fig. 4b.

For the DAF, the neuron output ζ_k of the k-th neuron is defined as,

$$\zeta_k = net_d^k(\kappa_k, p_k^U, p_k^L),\ k = 1, 2, \ldots, n_h, \quad (3)$$

and

$$net_d^k(\kappa_k, p_k^U, p_k^L) = \begin{cases} e^{\frac{-(\kappa_k - p_k^U)^2}{2p_k^{L\,2}}} - 1 & \text{if } \kappa_k \le p_k^U \\ 1 - e^{\frac{-(\kappa_k - p_k^U)^2}{2p_k^{L\,2}}} & \text{otherwise} \end{cases}. \quad (4)$$

p_k^U and p_k^L are the parameters of the DAF, which are effectively the dynamic mean and the dynamic standard deviation respectively of the k-th DAF. From (1) to (4), the input-output relationship of the proposed neuron is given by,

$$\zeta_k = net_d^k(net_s^k(\sum_{i=1}^{n_{in}} z_i v_{ik}), p_k^U, p_k^L). \quad (5)$$

2) Connection of the modified neural network: The MNN has n_{in} nodes in the input layer, n_h nodes in the hidden layer, and n_{out} nodes in the output layer. For the hidden layer neurons, the above neuron model is employed. For the output layer neurons, a static activation function is used. Considering an input-output pair $(\mathbf{z}, \mathbf{y})$ for the neural network, the output of the k-th node of the hidden layer is given by (5). The output of the MNN (Fig. 5) is given by,

$$y_l = net_o^l(\sum_{k=1}^{n_h} \zeta_k w_{kl}) \quad (6)$$

$$= net_o^l(\sum_{k=1}^{n_h} net_d^k(net_s^k(\sum_{i=1}^{n_{in}} z_i v_{ik}), p_k^U, p_k^L) w_{kl}), \quad (7)$$

where w_{kl}, $k = 1, 2, \ldots, n_h$; $l = 1, 2, \ldots n_{out}$ denotes the weight of the link between the k-th hidden and the i-th output nodes; $net_o^l(\cdot)$ denotes the activation function of the output neuron:

$$net_o^l(\sum_{k=1}^{n_h} \zeta_k w_{kl}) = \begin{cases} e^{\frac{-\left(\sum_{k=1}^{n_h} \zeta_k w_{kl} - m_o^l\right)^2}{2\sigma_o^{l\,2}}} - 1 & \text{if } \sum_{k=1}^{n_h} \zeta_k w_{kl} \le m_o^l \\ 1 - e^{\frac{-\left(\sum_{k=1}^{n_h} \zeta_k w_{kl} - m_o^l\right)^2}{2\sigma_o^{l\,2}}} & \text{otherwise} \end{cases}, \quad (8)$$

where m_o^l and σ_o^l are the mean and the standard deviation of the output node activation function respectively.

From Fig. 3, we can see that the first layer simply distributes the input variables. In the hidden layer, the SAF determines hyper planes as switching surfaces. Owing to the DAF, the input $\mathbf{p^U}$ concerns the bias term and the input $\mathbf{p^L}$ influences the sharpness of the edges of these hyper-planes. They are eventually combined into convex regions by the output layer. The static parameters of the MNN are trained by GA and the dynamic parameters are provided by the TNN.

B. Traditional Neural Network

The TNN is shown in Fig. 6. The input and output variables of the TNN are z_i and p_h respectively, where $i = 1, 2, \ldots, n_{in}$; $h = 1, 2, \ldots, q$, and $q = 2n_h$ is the number of output variables. The input-output relationship of the TNN is given by,

$$p_h = \text{tansig}\left(\sum_{g=1}^{n_{th}} u_{gh}\,\text{tansig}\left(\sum_{i=1}^{n_{in}} t_{ig} z_i - b_g^1\right) - b_h^2\right),\ h = 1, 2, \ldots, n_{tout}. \quad (9)$$

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

$$\text{tansig}(\alpha) = \frac{2}{1 + e^{-2\alpha}} - 1,\ \alpha \in \Re \quad (10)$$

p_h, $h = 1, 2, \ldots, q$ are the outputs of the TNN, which affect the parameters of the DAF. The number of outputs of the TNN doubles the number of hidden nodes of the MNN. In the MNN, $p_k^U \in [-0.5 \quad 0.5]$ and $p_k^L \in [0.01 \quad 0.5]$. Thus,

transformations from p_h to p_k^U and p_k^L are needed. Referring to Fig. 3 and Fig. 6, we can set $p_1^U = \mathrm{T_m}(p_1)$, $p_1^L = \mathrm{T}_\sigma(p_2)$, ..., $p_{n_h}^U = \mathrm{T_m}(p_{q-1})$, $p_{n_h}^L = \mathrm{T_s}(p_q)$, where $\mathrm{T_m}(\phi) = 2\phi$ and $\mathrm{T_s}(\phi) = -0.245\phi + 0.255$ are the transformation functions. The weights of the TNN are tuned by GA.

III. Training of Parameters of Neural-Tuned Neural Network Using Genetic Algorithm

In this section, the proposed neural-tuned neural network is employed to learn the input-output relationship of an application using GA with arithmetic crossover and non-uniform mutation [11]. 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 [11]. 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. Let the input-output relationship be described by,

$$\mathbf{y}^d(t) = g\left(\mathbf{z}^d(t)\right), \mathrm{t} = 1, 2, \ldots, n_d, \tag{11}$$

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

$$fitness = \frac{1}{1+err}, \tag{12}$$

$$err = \frac{\sum_{t=1}^{n_d}\sum_{k=1}^{n_{out}}\left|y_k^d(t) - y_k(t)\right|}{n_d n_{out}}. \tag{13}$$

The objective is to maximize the fitness value of (12) (minimize *err* of (13)) using GA by setting the chromosome to be $\left[v_{ik} \quad m_s^k \quad \sigma_s^k \quad w_{kl} \quad m_o^l \quad \sigma_o^l \quad t_{ig} \quad u_{gh} \quad b_g^1 \quad b_h^2\right]$ for all g, *h*, *i*, *k*, *l*. In this paper, $v_{ik}, w_{kl}, t_{ig}, u_{gh}, b_g^1, b_h^2 \in [-1 \quad 1]$, $m_s^k, m_o^l \in [-0.5 \quad 0.5]$ and $\sigma_s^k, \sigma_o^l \in [0.01 \quad 0.5]$. The range of the fitness value of (12) is [0, 1].

IV. Application Examples

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

A. Forecasting of the Sunspot Number

An application example on forecasting the sunspot number [3, 8] will be given in this session. The cycles formed by the sunspot numbers are non-linear, non-stationary, and non-Gaussian which are difficult to model and predict. We use the proposed neural-tuned neural network for the sunspot number forecasting. The inputs, z_i, of the proposed network are defined as $z_1(t) = y^d(t-1)$, $z_2(t) = y^d(t-2)$ and $z_3(t) = y^d(t-3)$ where t denotes the year and $y^d(t)$ is the sunspot numbers at the year t. The sunspot numbers of the first 180 years (i.e. $1705 \le t \le 1884$) are used to train the proposed neural network. Referring to (7), the proposed network used for the sunspot forecasting is governed by,

$$y(t) = net_o\left(\sum_{k=1}^{3} net_d^k\left(net_s^k\left(\sum_{i=1}^{3} z_i v_{ik}\right), p_k^U, p_k^L\right) w_k\right). \tag{13}$$

The fitness function is defined as follows,

$$fitness = \frac{1}{1+err}, \tag{14}$$

$$err = \sum_{t=1705}^{1884} \frac{(y^d(t))^2 - (y(t))^2}{180}. \tag{15}$$

In the proposed network, the number of hidden nodes (n_h) in the MNN is set at 3 and the number of hidden nodes (n_{th}) in the TNN is set at 2. These values are obtained by trial and error through experiments with good performance. GA is employed to tune the parameters of the proposed neural-tuned neural network of (13). The objective is to maximize the fitness function of (14). The population size used for the GA is 10. The chromosome is $\left[v_{ik} \quad m_s^k \quad \sigma_s^k \quad w_{kl} \quad m_o \quad \sigma_o \quad t_{ig} \quad u_{gh} \quad b_g^1 \quad b_h^2\right]$ for all g, *h*, *i*, *k*. The initial values of the parameters of the neural network are randomly generated. For comparison purpose, a traditional feed-forward neural network trained by the same GA, is also applied to forecast the sunspot number. The number of hidden nodes of the traditional neural network is 9 so that the numbers of training parameters in the two networks are the same. For all approaches, the number of iterations to train the neural network is 1000. 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.

The proposed network is used to forecast the sunspot number during the years 1885-1979. The fitness value, the training error (governed by (47)) and the forecasting error (governed by $\sum_{t=1885}^{1980} \frac{(y^d(t))^2 - (y(t))^2}{96}$) are tabulated in Table I. It can be observed from Table I that our approach performs better than the traditional approaches.

B. 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: rectangle, triangle and straight line (Fig. 7). We use 300 sets of 10 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 (7), the proposed network used for the pattern recognition is governed by,

$$y_l(t) = net_o^l\left(\sum_{j=1}^{6} net_d^j\left(net_s^j\left(\sum_{i=1}^{10} z_i v_{ij}\right), p_k^U, p_k^L\right) w_{jl}\right), \; l = 1, 2, 3. \tag{16}$$

GA is employed to tune the parameters of the proposed network of (16). The fitness function is defined as follows,

$$fitness = \frac{1}{1+err}, \tag{17}$$

$$err = \frac{\sum_{t=1}^{300}\sum_{k=1}^{3}\left(\left(\frac{y_k(t)}{\|\mathbf{y}(t)\|}\right)^2 - \left(\frac{y_k^d(t)}{\|\mathbf{y}^d(t)\|}\right)^2\right)}{300\times 3}. \tag{18}$$

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 MNN is set at 6 and the number of hidden node (n_{th}) in TNN is set at 5, which are chosen by trial and error through experiments for good performance. GA is employed to tune the parameters of the proposed NTNN of (16). The population size used for the GA is 10. The chromosomes used for the GA are $\left[v_{ik} \quad m_s^k \quad \sigma_s^k \quad w_{kl} \quad m_o^l \quad \sigma_o^l \quad t_{ig} \quad u_{gh} \quad b_g^1 \quad b_h^2\right]$ for all g, h, i, k, l. The initial values of the parameters of the neural network are randomly generated. For comparison, a traditional network trained by the GA is also used to recognize the patterns. The number of hidden node of the traditional neural network is 18 so that the number of training parameters is even larger. For all approaches, the number of iterations to train the neural network is 2000. For the GA, the probabilities of crossover and mutation are set at 0.8 and 0.1 respectively. The shape parameter of mutation is set at 1.

After training, we use 600 (200×3) sets of data for testing. The results are tabulated in Table II. From this Table, it can be seen that the training error (governed by (18)) and forecasting error (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

$$\text{Accuracy} = \left(\frac{n_{recognized}}{n_{testing}}\right)\times 100\%. \tag{19}$$

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 higher.

V. Conclusion

A neural-tuned neural network has been proposed. It consists of a modified neural network and a traditional neural network. A modified neuron model with two activation functions has been introduced in the hidden layer of the modified neural network. The parameters of the dynamic activation function in the modified neuron are tuned by the traditional neural network. By employing this neuron model and the network structure, the performance of the proposed network is found to be better than that of the traditional feed-forward neural networks. The parameters of the proposed network can be tuned by GA with arithmetic crossover and non-uniform mutation. Examples of sunspot forecasting 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, *Neurofuzzy 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, P.K.S. Tam, and Y.S. Lee, "A novel GA-based neural network for short-term load forecasting," *IEEE Trans. Ind. Electron*, vol.. 50, no. 4, pp. 793-799, Aug. 2003.

[5] S.H. Ling, H.K. Lam, F.H.F. Leung, and P.K.S. Tam, "Parameter learning of neural network using fuzzy genetic algorithm," in *Proc.2002 Congress on Evolutionary Computation (CEC 2002), World Congress on Computational Intelligence (WCCI 2002)*, Honolulu, Hawaii, May 12-17, 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. Ind. 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. Ind. Electron.*, to be published.

[8] M. Li, K. Mechrotra, C. Mohan, and S. Ranka, "Sunspot numbers forecasting using neural network," in *Proc. 5th IEEE Int. Symp. Intelligent Control*, 1990, pp. 524-528.

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

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

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

[12] B.D. Liu, C.Y. Chen, and J.Y. Tsao, "Design of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms," *IEEE Trans. Systems, Man and Cybernetics*, Part B, vol. 31 no. 1, pp. 32-53, Feb. 2001.

[13] K. Belarbi and F. Titel "Genetic algorithm for the design of a class of fuzzy controllers: an alternative approach," *IEEE Trans. Fuzzy Systems*, vol. 8, no. 4, pp. 398-405, Aug. 2000.

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

[15] R. Caponetto, L. Fortuna, G. Nunnari, L. Occhipinti, and M.G. Xibilia, "Soft computing for greenhouse climate control," *IEEE Trans. Fuzzy Systems*, vol. 8, no. 6, pp. 753-760, Dec. 2000.

[16] M. Srinivas and L.M. Patnaik, "Genetic algorithms: a survey," *IEEE Computer*, vol. 27, issue 6, pp. 17-26, Jun. 1994.

[17] L. Davis, *Handbook of Genetic Algorithms*. NY: Van Nostrand Reinhold, 1991.

[18] J.D. Schaffer, D. Whitley, and L.J. Eshelman "Combinations of genetic algorithms and neural networks: a survey of the state of the art," in *Proc. International Workshop on Combinations of Genetic Algorithms and Neural Networks, COGANN-92*, 1992, pp 1-37.

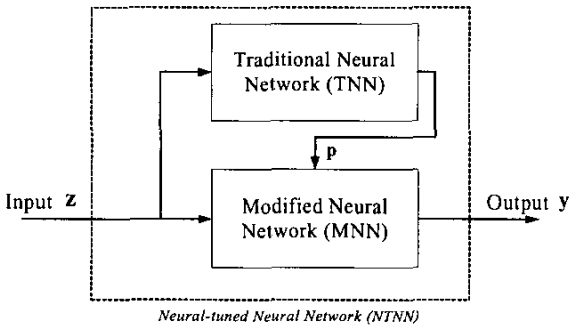


Fig. 1. Block diagram of the proposed neural-tuned neural network.

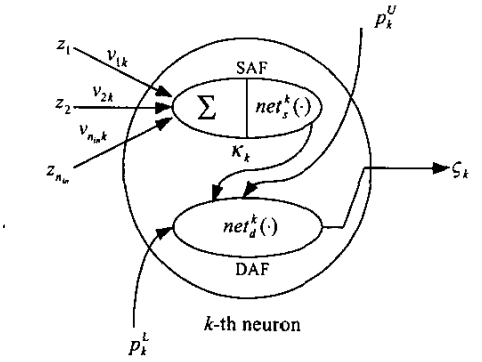


Fig. 2. Proposed modified neuron.

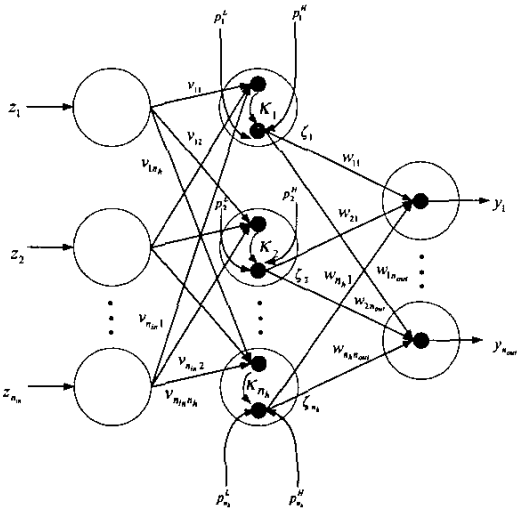
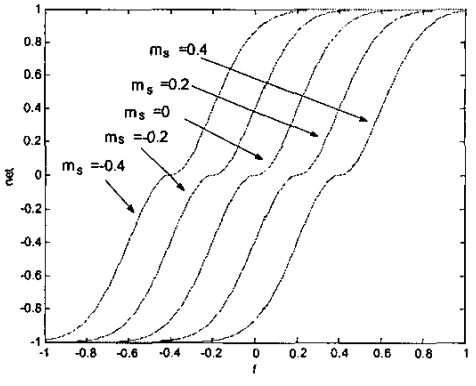
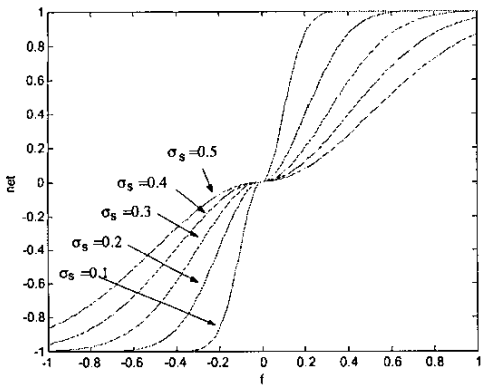


Fig. 3. Connection of the modified neural network (MNN).



(a)



(b)

Fig. 4. Sample activation functions of the proposed neuron: (a) $\sigma_s = 0.2$, (b) $m_s = 0$.

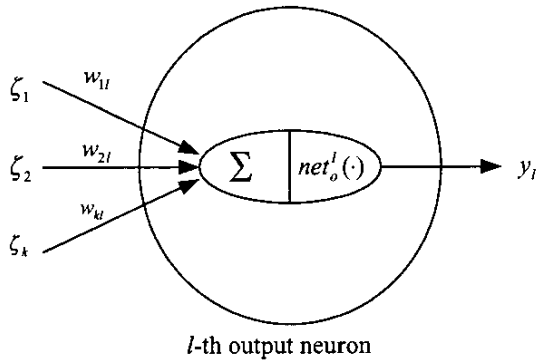


Fig. 5. Model of the proposed output neuron.

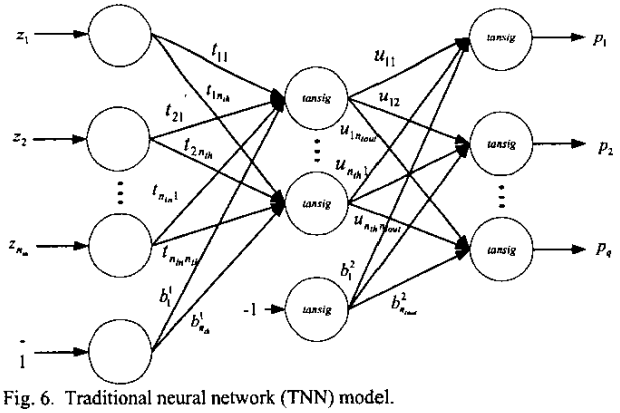


Fig. 6. Traditional neural network (TNN) model.

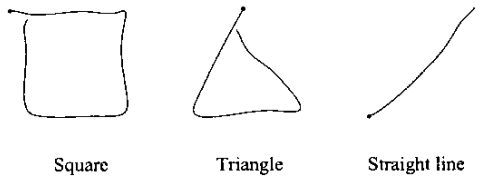


Fig. 7. Samples of the hand-written patterns.

TABLE I

RESULTS OF THE PROPOSED AND TRADITIONAL NEURAL NETWORKS FOR THE FORECASTING OF SUNSPOT NUMBER

	Proposed Network	Traditional Network
Fitness value	0.9944	0.9915
Training error (MSE)	1.1263	1.7147
Forecasting error (MSE)	1.7851	1.9265
Number of parameters	46	46

TABLE II

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

	Proposed Network	Traditional Network
Fitness value	0.9833	0.9709
Training error (MSE)	0.0169	0.030
Forecasting error (MSE)	0.0204	0.0415
Number of parameters	223	255
Recognition accuracy	95.67%	92.33%