International Journal of Intelligence Science, **2013, 3, 145-161**
http://dx.doi.org/10.4236/ijis.2013.34016 Published Online October 2013 (http://www.scirp.org/journal/ijis)

Hybrid Designing of a Neural System by Combining Fuzzy Logical Framework and PSVM for Visual Haze-Free Task

Hong Hu, Liang Pang, Dongping Tian, Zhongzhi Shi
Key Laboratory of Intelligent Information Processing, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
Email: huhong@ict.ac.cn, pangl@ics.ict.ac.cn, tiandp@ics.ict.ac.cn, shizz@ics.ict.ac.cn

Received June 28, 2013; revised August 25, 2013; accepted September 5, 2013

ABSTRACT

Brain-like computer research and development have been growing rapidly in recent years. It is necessary to design large scale dynamical neural networks (more than 10^6 neurons) to simulate complex process of our brain. But such a kind of task is not easy to achieve only based on the analysis of partial differential equations, especially for those complex neural models, e.g. Rose-Hindmarsh (RH) model. So in this paper, we develop a novel approach by combining fuzzy logical designing with Proximal Support Vector Machine Classifiers (PSVM) learning in the designing of large scale neural networks. Particularly, our approach can effectively simplify the designing process, which is crucial for both cognition science and neural science. At last, we conduct our approach on an artificial neural system with more than 10^8 neurons for haze-free task, and the experimental results show that texture features extracted by fuzzy logic can effectively increase the texture information entropy and improve the effect of haze-removing in some degree.

Keywords: Artificial Brain Research; Brain-Like Computer; Fuzzy Logic; Neural Network; Machine Learning; Hopfield Neural Network; Bounded Fuzzy Operator

1. Introduction

Driven by rapid ongoing advances in computer hardware, neuroscience and computer science, artificial brain research and development are blossoming [1]. The representative work is the Blue Brain Project, which has simulated about 1 million neurons in cortical columns and included considerable biological detail to reflect spatial structure, connectivity statistics and other neural properties [2]. The more recent work of a large-scale model for the functioning brain is reported in the famous journal Science, which is done by the group of Chris Eliasmith's group [3]. In order to bridge the gap between neural activity and biological function, Chris Eliasmith's group presented a 2.5-million-neuron model of the brain (called "Spaun") to exhibit many different behaviors. Among these large scale visual cortex simulations, the visual cortex simulations are most concerned. The two simulations aforementioned are all about the visual cortex. The number of neurons in cortex is enormous. According to [4], the total number in area 17 of the visual cortex of one hemisphere is close to 160,000,000. For the total cortical thickness the numerical density of synapses is 276,000,000 per mm^3 of tissue. It is almost impossible to design or analyze a neural network with more than 10^8 neurons only based on partial differential equations. The nonlinear complexity of our brain prevents our progress from simulating useful and versatile functions of our cortex system. Many studies only deal with simple neural networks with simple functions, and the connection matrices should be simplified. The visual functions simulated by "Blue Brain Project" and "Spaun" are so simple that they are nothing in the traditional pattern recognition.

On the other hand, logic inference plays a very important role in our cognition. With the help of logical design, the things become simple, and this is the reason why computer science has made great progress. There are more than 10^8 transistors in a CPU today. Why don't we use similar techniques to build complex neural networks? The answer is yes. As our brains work in the non Turing computable way, fuzzy logic rather than Boolean logic should be used. For this purpose, we introduce a new concept-fuzzy logical framework of a neural network. Fuzzy logic is not a new topic in science, but it is really very fundamental and useful. If the function of a dynamical neural network can be described by fuzzy logical formulas, it can greatly help us to understand behavior of

this neural network and design it easily.

For neural systems, the basic logic processing module to be used as a building module in the logic architectures of the neural network comes from OR/AND neuron [3,5], also referred by [6]. The ideal of hybrid design neural networks and fuzzy logical system is firstly proposed by [7]. While neural networks and fuzzy logic have added a new dimension to many engineering fields of study, their weaknesses have not been overlooked, in many applications the training of a neural network requires a large amount of iterative calculations. Sometimes the network cannot adequately learn the desired function. Fuzzy systems, on the other hand, are easy to understand because they mimic human thinking and acquire their knowledge from an expert who encodes his knowledge in a series of if/then rules [7].

Neural networks can work either in dynamical way or static way. The former can be described by partial differential equations and denoted as "dynamical neural networks". Static points or stable states are very important for dynamical analysis of a neural network. Many artificial neural networks are just abstract of static points or stable states of dynamical neural networks, e.g. perception neural networks, such a kind of artificial neural networks work in a static way and are denoted as "static neural networks". There is a natural relation between a static neural network and a fuzzy logical system, but for dynamical neural networks, we should extend the static fuzzy logic to dynamic fuzzy logic. A novel concept denoted as "fuzzy logical framework" is defined for this purpose.

At last, we give out an application of our hybrid designing approach for the visual task about image matting-haze removing from a single input image. Image matting refers to the problem of softly extracting the foreground object from a single image. The system designed by our novel hybrid approach has a comparable ability with ordinary approach proposed by [8]. Texture information entropy (TIE) is introduced for roughly evaluating the effect of haze removing. Experiments show texture features extracted by fuzzy logic can effectively increase TIE.

The main contributions of this paper include: 1) we develop a novel hybrid designing approach of neural networks based on fuzzy Logic and Proximal Support Vector Machine Classifiers (PSVM) learning in the artificial brain designing, which greatly simplifies the designing of large scale artificial brain; 2) a novel concept about fuzzy logical framework of neural network is firstly proposed; 3) instead of the linear mapping in [8], a novel nonlinear neural fuzzy logical texture feature extracting, which can effectively increase TIE, is introduced in the task of haze free application. The experiments show that our approach is effective.

2. Hopfield Model

There are many neuron models, e.g. Fitz Hugh (1961), Morris, Lecar (1981), Chay (1985) and Hindmarsh, Rose (1984) [9-11]. Whether the fuzzy logical approach can be used in all kinds of neural networks for different neuron models? In order to answer this question, we consider a simple neuron model- Hopfield model [12] (see Equation (1.1)) as a standard neuron model, which has a good character of fuzzy logic. We have proved that Hopfield model has universal meaning, such that almost all neural models described by first order differential equations can be simulated by them with arbitrary small error in an arbitrary finite time interval [13], these neural models include all the models summarized by H D I [14].

$$\begin{aligned} \dot{U}_i &= -a_i \cdot U_i + \sum_j w'_{ij} V_j + \beta \sum_k w_{ik} I_k ; \\ V_i &= S\left(U_i - T_i\right) \end{aligned} \tag{1.1}$$

where sigmoid function $S(\,)$ can be a piecewise linear function or logistic function. Hopfield neuron model has a notable biological characteristic and has been widely used in visual cortex simulation. One example of them is described in [7,10,15-17]), (see Equation (1.2)). Such cellos membrane potential is transferred to output by a sigmoid-like function. Only the amplitude of output pluses carries meaningful information. The rising or dropping time (Δt) of output pluses conveys no useful information and is always neglected. According to [15], the neural networks described by Equation (1.2) are based on biological data [18-23].

In such kind neural networks, cells are arranged on a regular 2-dimensional array with image coordinates $i = (n_i, m_i)$ and divided into two categories: excitatory cells $x_{i\theta}$ and inhibitory cells $y_{i\theta}$. At every position $i = (n_i, m_i)$, there are M cells with subscript t_θ that are sensitive to a bar of the angle θ. Equation (1.2) is the dynamical equation of these cells. Only excitatory cells receive inputs from the outputs of edge or bar detectors. The direction information of edges or bars is used for segmentation of the optical image.

$$\begin{aligned} \dot{x}_{i\theta} &= -a_x x_{i\theta} - g_y\left(y_{i\theta}\right) + J_c g_x\left(x_{i\theta}\right) + I_c \\ &\quad - \sum_{\nabla\theta \neq 0} \psi\left(\nabla\theta\right) g_y\left(y_{i,\theta+\nabla\theta}\right) + \sum_{j\neq i,\theta'} J_{i\theta,j\theta'} g_x\left(x_{j\theta'}\right) + I_{i\theta}; \\ \dot{y}_{i\theta} &= -a_y y_{i\theta} - g_x\left(x_{i\theta}\right) + g_x\left(x_{i\theta}\right) + I_c + \sum_{j\neq i,\theta'} W_{i\theta,j\theta'} g_x\left(x_{j\theta'}\right). \end{aligned} \tag{1.2}$$

where $g_x(x)$ and $g_y(x)$ are sigmoid-like activation functions, and Ψ is the local inhibition connection in the location i, and $J_{i\theta,j\theta'}$ and $W_{i\theta,j\theta'}$ are the synaptic connections between the excitatory cells and from the excitatory cells to inhibition cells, respectively. If we represent the excitatory cells and inhibitory cells with

same symbol U_i and summarize all connections (local ψ, global exciting $W_{i\theta,j\theta'}$ and global inhibiting $J_{i\theta,j\theta'}$) as w'_{ij}, the Equation (1.2) can be simplified as Hopfield model Equation (1.1).

3. Fuzzy Logical Framework of Neural Network

Same fuzzy logical function can have several equivalent formats; these formats can be viewed as the structure of a fuzzy function. When we discuss the relationship between the fuzzy logic and neural network, we should not only probe the input-output relationship but also their corresponding structure. Section 3.1 discusses this problem. Section 3.2 discusses the problem about what is the suitable fuzzy operator, and in Section 3.3, we prove three theorems about the relationship between the fuzzy logic and neural network.

3.1. The Structure of a Fuzzy Logical Function and Neural Network Framework

In order to easily map a fuzzy formula to a dynamical neural network, we should define the concept about the structure of a fuzzy logical function.

Definition 1. [The structure of a fuzzy logical function] If S^1 is a set of fuzzy logical functions(FLF), and a FLF $f(x_1, x_2, \cdots, x_n)$ can be represented by the combination of all FLFs in S^1 with fuzzy operators "$\wedge$" and "$\vee$", but with no parentheses, then the FLFs in S^1 is denoted as the 1^{st} layer sub fuzzy logical functions (1^{st} FLF) of $f(x_1, x_2, \cdots, x_n)$; similarly, if a variable x_i in a FLF in S^1 is not just a variable, *i.e.* $x_i = f_1(y_1, y_2, \cdots, y_n)$, then $f_1(y_1, y_2, \cdots, y_n)$ has its own 1^{st} layer sub FLFs which are denoted as the 2^{nd} layer sub FLFs of $f(x_1, x_2, \cdots, x_n)\cdots$, and every k^{th} layer non variable sub FLF can have its sub fuzzy logical functions. In this way, $f(x_1, x_2, \cdots, x_n)$ has a layered structure of sub fuzzy logical functions, we denote such layered structure as the structure of $f(x_1, x_2, \cdots, x_n)$.

For example, $f(x_1, x_2, x_3, x_4)$ in **Figure 1** can be represented by $f_1(x_1, x_2) = x_1 \wedge x_2$ and $f_2(x_2, x_3, x_4) = x_2 \wedge x_3 \vee x_4$ as $f_1(x_1, x_2) \vee f_2(x_2, x_3, x_4)$, so $f_1(x_1, x_2)$ and $f_2(x_2, x_3, x_4)$ are the 1^{st} layer FLFs in S^1.

$f(x_1, x_2, \cdots, x_n)$ may have several equivalent formats, so the structure of $f(x_1, x_2, \cdots, x_n)$ is not unique, for example,

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= x_1 \wedge x_2 \vee x_2 \wedge x_3 \vee x_4 \\ &= x_1 \wedge x_2 \vee (x_2 \wedge x_3 \vee x_4) \\ &= ((x_1 \wedge x_2) \vee (x_2 \wedge x_3)) \vee x_4 \cdots \end{aligned} ,$$

can be represented as trees (a) and (b) in **Figure 1**. If a sub fuzzy logical function is just $f(x_1, x_2, \cdots, x_n)$ itself, then $f(x_1, x_2, \cdots, x_n)$ has a recurrent structure, otherwise, $f(x_1, x_2, \cdots, x_n)$ has a tree kind structure.

If $f(x_1, x_2, \cdots, x_n)$ has a recurrent structure, then it can be represented as Equation (1.3), and the time needed for output is Δt and $f^t(x_1, x_2, \cdots, x_n)$ is linearly changed to $f^{t+\Delta t}(x_1, x_2, \cdots, x_n)$ then $f(x_1, x_2, \cdots, x_n)$ can create a time serial output and can be written in partial differential form as Equation (1.4).

$$f(x_1, x_2, \cdots, x_n) = g(x_1, x_2, \cdots, x_n, f(x_1, x_2, \cdots, x_n)) \quad (1.3)$$

$$f^{(t+dt)}(x_1, x_2, \cdots, x_n) = f^t(x_1, x_2, \cdots, x_n) + \left[g(f^t(x_1, x_2, \cdots, x_n), x_1, x_2, \cdots, x_n) - f^t(x_1, x_2, \cdots, x_n) \right] dt / \Delta t \quad (1.4)$$

where $(x_1, x_2, \cdots, x_n)$ is a stable input vector which can be viewed as an initial condition of a dynamical system and $0 \le dt \le \Delta t$. Suppose the dynamical behavior of a neural network can be described as

$$\vec{y}(t) = F(\vec{x}(t))$$

where $\vec{x}(t)$ is the input at time t and $\vec{y}(t)$ is the output at t. If $G(\vec{x}(t))$ is a FLF with same dynamical variables in $\vec{x}(t)$, we again define $\vec{y}_G(t) = G(\vec{x}(t))$ as the dynamical behavior of a fuzzy logical function, where $\vec{x}(t)$ and $\vec{y}_G(t)$ are input and output at t respectively, and the input domain for $\vec{x}(t)$ is D.

The definition 2 is the measure about the difference between a fuzzy logical function and a neural network.

Definition 2. [Difference error between a fuzzy logical function and a neural network] If $\vec{X}(t)$ is input, and the region of input $\vec{X}(t)$ is D. The difference error between a fuzzy logical function G and a neural network F is defined as:

Dynamical case:

$$(a)\, err_G = \int_0^T \left| G(\vec{x}(t)) - F(\vec{x}(t)) \right| \mathrm{d}t \quad (1.5)$$

Static case:

$$err_G = \sum_{\vec{x} \in D} \left| G(\vec{x}) - F(\vec{x}) \right|, \quad (1.6)$$

where $F(\vec{x})$ is the fixed point of F.

Usually, a neural model can only approximately simulate a fuzzy operator, so it is necessary to find the most similar fuzzy function for a neural network, which is denoted as the fuzzy logical framework of a neural network, the definition 3 gives out the concept of the fuzzy logical framework.

Definition 3. [The fuzzy logical framework] Suppose

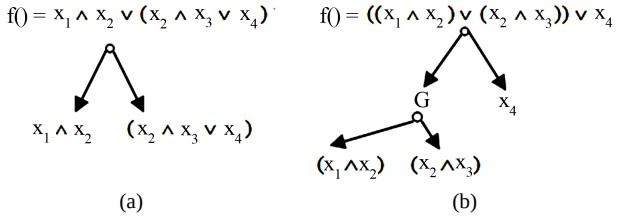

Figure 1. $f(x_1, x_2, \cdots, x_n)$ **may have several equivalent formats, so the structure of** $f(x_1, x_2, \cdots, x_n)$ **is not unique.**

Ω is a set of fuzzy logical functions, a fuzzy logical framework for a neural network F is the best fuzzy logical function G in Ω satisfied the constrain that G has smallest err_G *i.e.* $err_G = \min_{G' \in \Omega} err_{G'}$. If there is a one to one onto mapping from neurons in a neural network F to the all layers sub fuzzy logical functions in a G's structure, such kind fuzzy logical framework is denoted as structure keeping fuzzy logical framework.

3.2. The Suitable Fuzzy Operator

After the theory of fuzzy logic was conceived by [24], many fuzzy logical systems have been presented, for example, the Zadeh system, the probability system, the algebraic system, and Bounded operator system, etc. According to universal approximation theorem [25], it is not difficult to prove that q-value weighted fuzzy logical functions (5) can precisely simulate Hopfield neural networks with arbitrary small error, or vice versa, *i.e.* every layered Hopfield neural network has a fuzzy logical framework of the q-value weighted bounded operator with arbitrary small error. This means that if the sigmoid function used by Hopfield neurons is a piecewise linear function, such kind fuzzy logical framework is structure keeping. Unfortunately, if the sigmoid function is logistic function, such kind fuzzy logical framework is usually not structure keeping. Only in an approximate case(see Appendix A), a layered Hopfield neural network may have a structure keeping fuzzy logical framework.

Definition 4. [Bounded Operator $F(\oplus_f, \otimes_f)$] Bounded product:

$$x \otimes_f y = \max(0, x + y - 1),$$

and Bounded sum:

$$x \oplus_f y = \min(1, x + y),$$

where $0 \le x, y \le 1$.

In order to simulate neural cells, it is necessary to extend the Bounded Operator to Weighted Bounded Operator. The fuzzy formulas defined by q-value weighted bounded operators is denoted as q-value weighted fuzzy logical functions.

Definition 5. [q-value Weighted Bounded operator $F(\oplus_f, \otimes_f)$] q-value Weighted Bounded product:

$$\begin{aligned} p_1 \otimes_f p_2 &= F_{\otimes_f}(p_1, p_2, w_1, w_2) \\ &= \max\left(0, w_1 p_1 + w_2 p_2 - (w_1 + w_2 - 1) q\right) \end{aligned} \tag{1.7}$$

q-value Weighted Bounded sum:

$$p_1 \oplus_f p_2 = F_{\oplus_f}(p_1, q_2, w_1, w_2) = \min\left(q, w_1 p_1 + w_2 p_2\right) \tag{1.8}$$

where $0 \le p_1, p_2 \le q$.

For association and distribution rules, we define:

$$\left(p_1 \Delta_f p_2\right) \Theta_f p_3 = F_{\Theta_f}\left(F_{\Delta_f}(p_1, p_2, w_1, w_2), p_3, 1, w_3\right)$$

and

$$p_1 \Delta_f \left(p_2 \Theta_f p_3\right) = F_{\Delta_f}\left(p_1, F_{\Theta_f}(p_2, p_3, w_2, w_3), w_1, 1\right),$$

Here $\Delta_f, \Theta_f = \otimes_f$ or $\oplus_f$. We can prove that $\oplus_f$ and $\otimes_f$ follow the associative condition (see Appendix B) and

$$x_1 \oplus_f x_2 \oplus_f x_3 \oplus_f \cdots \oplus_f x_n = \min\left(q, \sum_{1 \le i \le n} w_i x_i\right) \tag{1.9}$$

$$\begin{aligned} &x_1 \otimes_f x_2 \otimes_f x_3 \otimes_f \cdots \otimes_f x_n \\ &= \max\left(0, \sum_{1 \le i \le n} w_i x_i - \left(\sum_{1 \le i \le n} w_i - 1\right) q\right) \end{aligned} \tag{1.10}$$

For more above q-value weighted bounded operator $F(\oplus_f, \otimes_f)$ follows the Demorgan Law, *i.e.*

$$\begin{aligned} &N\left(x_1 \oplus_f x_2 \oplus_f x_3 \oplus_f \cdots \oplus_f x_n\right) \\ &= q - \min\left(q, \sum_{1 \le i \le n} w_i x_i\right) \\ &= \max\left(0, q - \sum_{1 \le i \le n} w_i x_i\right) \\ &= \max\left(0, \sum_{1 \le i \le n} w_i (q - x_i) - \left(\sum_{1 \le i \le n} w_i - 1\right) q\right) \\ &= N(x_1) \otimes_f N(x_2) \otimes_f N(x_3) \otimes_f \cdots \otimes_f N(x_n). \end{aligned} \tag{1.11}$$

But for the q-value weighted bounded operator $F(\oplus_f, \otimes_f)$, the distribution condition is usually not hold, and the boundary condition is hold only all weights equal to 1, for

$$p_1 \otimes_f q = F_{\otimes_f}(p_1, q, w_1, w_2) = \max\left(0, w_1 p_1 + (1 - w_1) q\right)$$

and $p_1 \oplus_f q = F_{\oplus_f}(p_1, q, w_1, w_2) = \min\left(q, w_1 p_1 + w_2 q\right)$.

3.3. Three Important Theorems about the Relationship between the Fuzzy Logic and Neural Network

H D I [14] studied the synchronization of 13 neuron models. These models include [9,11,26-35] and integrate-and-fire model [36] [see Equation (1.14), the rest

11 neuron models are all the special cases of the generalized model described by the ordinary differential Equation (1.16).

$$\begin{aligned}&\ddot{x}_i+\mu\left(x_{ai}^2-p^2\right)\dot{x}_i+g^2x_{ai}=f\left(t\right).\\&x_{ai}=x_i+\sum_j\lambda_{ji}x_j\end{aligned}\tag{1.12}$$

$$\begin{aligned}&\dot{x}_i=y_i,\\&\dot{y}_i+\mu\left(x_{ai}^2-p^2\right)y_i+g^2x_{ai}=f\left(t\right).\\&x_{ai}=x_i+\sum_j\lambda_{ji}x_j\end{aligned}\tag{1.13}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t}=-\frac{v}{\tau_0}+I_{ext}+I_{syn}\left(t\right)\tag{1.14}$$

where $0<v<\Theta$ and $v\left(t_0^+\right)=0$ if $v\left(t_0^-\right)=\Theta$. Usually $I_{syn}\left(t\right)$ is defined by Equation (1.15).

$$\begin{aligned}&I_{syn}\left(t\right)=g\sum_{spikes}f\left(t-t_{spike}\right)\\&f\left(t\right)=A\left[\exp\left(-t\tau_1\right)-\exp\left(-t\tau_2\right)\right]\end{aligned}\tag{1.15}$$

In fact, if we introduce a new variable $y_i=\dot{x}_i$, the Van-der-Pol generator [28] model can be changed to Equation (1.13) which is just a special case of Equation (1.16); for the integrate-and-fire model, if we use logistic function $\frac{1}{1+\exp\left(-\lambda\left(x-T\right)\right)}$ to replace the step function, the integrate-and-fire model can also has the form of the Equation (1.16). So the Equation (1.16) can be viewed as a general representation of almost all neuron models, if we can prove the Equation (1.16) can be simulated by a neural network based on the Hopfield neuron model [see Equation (1.1)], then almost all neuron models can be simulated in the same way.

$$\begin{cases}\dot{x}_1=-a_1x_1+w_1f_1\left(x_1,x_2,\cdots,x_n\right)+u_1\\\dot{x}_2=-a_2x_2+w_2f_2\left(x_1,x_2,\cdots,x_n\right)+u_2\\\quad\vdots\\\dot{x}_n=-a_nx_n+w_nf_n\left(x_1,x_2,\cdots,x_n\right)+u_n\end{cases}\tag{1.16}$$

where every $f_i\left(x_1,x_2,\cdots,x_n\right),1\le i\le n$, has the continuous partial differential in the finite hypercubic domain $D=\left[a_1,b_1\right]\times\left[a_2,b_2\right]\times\cdots\times\left[a_n,b_n\right]$ of its trajectory space $TR=\left\{\left(x_1\left(t\right),x_2\left(t\right),\cdots,x_n\left(t\right)\right):0\le t\le T\right\}$.

The Equation (1.17) is the fixed point of a Hopfield neural circuit which only has one cell with input I_k.

$$\bar{U}_i=\sum_k w_{ik}I_k/a_i,\ \bar{V}_i=1\Big/\left[\exp\left(-\lambda\left(U_i-T_i\right)\right)+1\right]\tag{1.17}$$

At the fixed point, every neuron works just like a neuron in a perception neural network. Theorem 1 tries to show the condition of Equation (1.17) to simulate disjunctive normal form (DNF) formula. The fixed point of Equation (1.17) can easily simulate binary logical operators; on the other hand, a layered neural network can be simulated by a q-value weighted fuzzy logical function.

Theorem 1 Suppose in Equation (1.17), $a_i=1$, and every $w_{ik}=\beta_kT_i,\beta_k>0,T_i>0,1\le k\le K$, for more, $C=\left\{S_i\middle|i=1,\cdots,L\right\}$ is a class of index sets, and every index set S_i is a subset of $\left\{1,2,3,\cdots,K\right\}$, then we have:

1. If $f\left(x_1,x_2,\cdots,x_k\right)=\bigvee_{l=1,\cdots,L}\left(\wedge_{j_l\in S_l}x_{j_l}\right)$ is a disjunctive normal form (DNF) formula, and the class $C=\left\{S_i\middle|i=1,\cdots,L\right\}$ is the class which has the following two characters: (1). for every $S_i,S_j\in C$, $S_i\bigcap S_j\ne S_k\in C$ for all k and $i\ne j$ (this condition assures that $f\left(x_1,x_2,\cdots,x_k\right)$ has a simplest form); (2). every S_i has the character $\sum_{j\in S_l}\beta_j>1$, where $S_i\in C$, and any index sets $S'\notin C$ have character $\sum_{j\in S'}\beta_j<1$, or if $\sum_{j\in S'}\beta_j>1$, there must be an index set $S_i\in C$ such that $S'\bigcap S_i=S_i$ (this condition assures C is the largest), then the fixed point of the neural cell described by Equation (1.17) can simulate the DNF formula

$$f\left(x_1,x_2,\cdots,x_k\right)=\bigvee_{l=1,\cdots,L}\left(\wedge_{i\in S_l}x_{j_i}\right)$$

with arbitrary small error, where $x_i=z_i$, if the corresponding input $I_i=z_i$, or $x_i=\bar{z}_i$ if $I_i=1-z_i$.

2. If a neural cell described by Equation (1.17) can simulate the Boolean formula $f\left(x_1,x_2,\cdots,x_k\right)$ with arbitrary small error, and $\left(\wedge_{i\in S_l}x_i\right)$ is an item in the disjunctive normal form of $f\left(x_1,x_2,\cdots,x_k\right)$, *i.e.* $f\left(x_1,x_2,\cdots,x_k\right)=1$ at $x_j=1$ for all $j\in S_l$ and $x_j=0$ for all $j\notin S_l$, then $\sum_{i\in S_l}\beta_i>1$.

3. If a couple of index sets S_{l_1} and S_{l_2} can be found in the formula $f\left(x_1,x_2,\cdots,x_k\right)=\bigvee_{l=1,\cdots,k,}\left(\wedge_{t\in S_l}x_t\right)$, such that $\left(\wedge_{t_1\in S_{l_1}}x_{t_1}\right)\wedge\left(\wedge_{t_2\in S_{l_2}}x_{t_2}\right)=z_i\wedge\bar{z}_i=\text{false}$, then the fixed point of the neural cell described by Equation (1.17) can't simulate the formula $f\left(x_1,x_2,\cdots,x_k\right)$.

Proof

1. If $I_t=1$, for all $t\in S_l$, and $I_t=0$, for all $t\notin S_l$, because $\sum_{i\in S_l}\beta_i>1$, then for the index set S_l is a subset

of $\{1,2,3,\cdots,K\}$, we have

$$\begin{aligned}\bar{V}_i &= 1\Big/\left[\exp\left(-\lambda\left(U_i - T_i\right)\right)+1\right] \\ &= 1\Big/\left[\exp\left(-\lambda\left(\sum_{1\le k\le K} w_{ik}I_k - T_i\right)\right)+1\right] \\ &= 1\Big/\left[\exp\left(-\lambda\left(\sum_{i\in S_l}\beta_i - 1\right)T_i\right)+1\right],\end{aligned}$$

so $\lim_{\lambda\to+\infty}\bar{V}_i = 1 = f\left(x_1,x_2,\cdots,x_k\right)$. If $I_t = 1, \forall t \in S'$; $I_t = 0, \forall t \notin S'$ and $S' \notin C$, then according to the condition of this theorem: if

$$\sum_{i\in S'}\beta_i < 1,\ \lim_{\lambda\to+\infty}\bar{V}_i = 0 = f\left(x_1,x_2,\cdots,x_k\right);$$

if $\sum_{i\in S'}\beta_i > 1$, then there is an index set $S_i \in C$ such that $S' \bigcap S_i = S_i$, then $\lim_{\lambda\to+\infty}\bar{V}_i = 1 = f\left(x_1,x_2,\cdots,x_k\right)$. So when $\lambda \to \infty$, the static error defined by Equation (1.6) trends to 0.

2. If the fixed point of the neural cell described by Eqaution (1.17) can simulate the Boolean formula $f\left(x_1,x_2,\cdots,x_k\right)$ which is not a constant with arbitrary small error, and for a definite binary input $x_1,x_2,\cdots,x_k$, then the arbitrary small error is achieved when λ trends to infinite and $\left(\bar{U}_i - T_i\right) = \sum_{k\in S_l} w_{ik}I_k - T_i \neq 0$ where S_l is the set of the labels and $I_i = 1$, for all $i \in S_l$, and $I_i = 0$, for all $i \notin S_l$. The theorem's condition supposes that every $w_{ik} = \beta_k T_i, \beta_k > 0, T_i > 0, 1 \le k \le K$, and $x_1,x_2,\cdots,x_k$ are binary number 0 or 1, so if $f\left(x_1,x_2,\cdots,x_k\right)$ is not a constant, when $f\left(x_1,x_2,\cdots,x_k\right) = 0$, there must be $\lim_{\lambda\to+\infty} V_i = 0$; and when $f\left(x_1,x_2,\cdots,x_k\right) = 1$, it is necessary for $\lim_{\lambda\to+\infty} V_i = 1$. $\lim_{\lambda\to+\infty} V_i = 0$ needs that $-\lambda\left(\sum_{i\in S_l}\beta_i T_i - T_i\right)$ trends to minus infinite and $\lim_{\lambda\to+\infty} V_i = 1$ needs that $-\lambda\left(\sum_{i\in S_l}\beta_i T_i - T_i\right)$ trends to plus infinite. So if $f\left(x_1,x_2,\cdots,x_k\right) = 1$ at $x_j = 1$ for all $j \in S_l$ and $x_j = 0$ for all $j \notin S_l$, in order to guarantee

$$\lim_{\lambda\to+\infty} err_{f(x_1,x_2,\cdots,x_k)}\left(w_{i,1},w_{i,2},\cdots,w_{i,k},T_i\right) = 0,$$

$\sum_{i\in S_l}\beta_i > 1$ must be hold, here

$err_{f(x_1,x_2,\cdots,x_k)}\left(w_{i,1},w_{i,2},\cdots,w_{i,k},T_i\right)$ is the static error defined by Equation (1.6) between the neural cell described by Equation (1.17) and $f\left(x_1,x_2,\cdots,x_k\right)$.

3. The third part of the theorem is based on the simple fact that for a single neuron V_i is monotone on every input I_i which can be z_i or $1-z_i$.

An example of above theorem is that the *xor* function can't be simulated by the neuron described by Equation (1.17).

If the neural network described by Equation (1.1) has a layered structure, the fixed point of a neuron at the non-input layer l is

$$\begin{aligned}U_{l,i} &= \sum_k w_{l,i,k}V_{l-1,i}/a_i \\ V_{l,i} &= S\left(U_{l,i} - T\right)\end{aligned} \tag{1.18}$$

Equation (1.18) is just a perception neural network, so a perception neural network can be viewed as an abstract of static points or stable states of a real neural network described by Equation (1.18).

Theorem 2 shows the fact that a continuous function can be simulated by a layered Hopfield neural network just like a multi layered perception neural network with arbitrary small error, and a layered neural network can be simulated by a q-value weighted fuzzy logical function. Theorem 2 is directly from the universal approximation theorem [25,37]'s proof.

Theorem 2 If $f\left(x_1,\cdots,x_m\right)$ is a continuous mapping from $[0,1]^m$ to $(0,1)^p$, for any $\epsilon > 0$, we can build a layered neural network Ψ defined by Equation (1.18), and its fixed point can be viewed as a continuous map $F\left(x_1,\cdots,x_m\right) = \left(F_1\left(x_1,\cdots,x_m\right),\cdots,F_q\left(x_1,\cdots,x_m\right)\right)$ from $[0,1]^m$ to $(0,1)^p$, such that $\left|F\left(x_1,\cdots,x_m\right) - f\left(x_1,\cdots,x_m\right)\right| < \varepsilon$, here $x_1,x_2,\cdots,x_m$ are k inputs of the neural network. For more, for an arbitrary layered neural network Ψ defined by Equation (1.17) which has a fixed point function $F\left(x_1,\cdots,x_m\right)$, we can find a q-value fuzzy logical function $F'\left(x_1,\cdots,x_m\right) = \left(F_1'\left(x_1,\cdots,x_m\right),m,F_q'\left(x_1,\cdots,x_m\right)\right)$ of weighted Bounded operator , such that

$$\left|F\left(x_1,\cdots,x_m\right) - F'\left(x_1,\cdots,x_m\right)\right| < \varepsilon.$$

Theorem 3 tries to prove that all kind recurrent neural networks described by the Equation (1.16) can be simulated by Hopfield neural networks described by Equation (1.1). The ordinary differential Equation (1.16) has a strong ability to describe neural phenomena. The neural network described by Equation (1.16) can have feedback. For the sake of the existence of feedback of a recurrent neural network, chaos will occur in such a neural network. As we known, the important characteristics of chaotic dynamics, *i.e.*, aperiodic dynamics in deterministic systems are the apparent irregularity of time traces and the divergence of the trajectories over time (starting from two nearby initial conditions). Any small error in the calculation of a chaotic deterministic system will cause unpredictable divergence of the trajectories over time, *i.e.* such kind neural networks may behave very differently under different precise calculations. So any small difference between two approximations of a trajectory of a chaotic recurrent neural network may create two totally different approximate results of this trajectory. Fortunately, all animals have only limited life and the

domain of trajectories of their neural network are also finite, so for most neuron models, the Lipschitz condition is hold in a real neural system, and in this case, the simulation is possible.

Theorem 3 If $[0,T], +\infty > T > 0$, is an arbitrary finite time interval, and the partial differential for all $f_i(x_1, x_2, \cdots, x_n)$ and x_j in Equation (1.16), $i, j = 1, 2, \cdots, n$ are continuous in the finite domain D of its trajectory space then every neural network NC described by Equation (1.16) can be simulated by a Hopfield neural network described by Equation (1.1) in the time interval $[0,T]$ and the finite domain D with an arbitrary small error $\varepsilon > 0$.

Proof The detail is showed in [13].

Neural networks described by Equation (1.16) can include almost all neural models found nowadays. [13] uses 1251 neurons Hopfield neural network to simulate a Rose-Hindmarsh (RH) neuron according to Theorem 3. Rose-Hindmarsh (RH) neuron is much more complicate than Hopfield neurons. To simulate a complicate neuron by simple neurons is not a difficult task, but the reverse task is almost impossible to complete, *i.e.*, it is almost impossible to simulate a Hopfield neuron by a set of RH neurons.

4. Hybrid Designing Based on the Fuzzy Logic and PSVM

For the sake of pages, in this paper only layered neural networks are discussed. For a layered neural network N, if the number of neurons at every layer is fixed, the number of structure keeping fuzzy logical frameworks of N is fixed, so the number of the fuzzy logical frameworks of a neural network is also fixed. When the coefficients of N are continuously changed, the neural network N is shifted from one structure keeping fuzzy logical framework to another.

There are two different parameters in a dynamical layered neural network. The first kind parameters are weights which represent the connection topology of neural network. The second parameters are time coefficients which control the time of spikes.

Time coefficients should be decided according to the dynamical behavior of the whole neural network.

There are two ways to design weights of a layered neural network: (1) according to the logical relation about this neural network, we can design the weights based on the q-value weighted fuzzy logical functions; (2) According to the input and output relation function $f_i(x_1, x_2, x_3, \cdots, x_n)$, we use machine learning approaches ,e.g. Back Propagation method, to learn weights for $f_i(x_1, x_2, x_3, \cdots, x_n)$. In order to speed up the learning process, for a layered neural network, we combine logical designing with PSVM [15], which is called as "Logical support vector machine (LPSVM)".

LPSVM algorithm:

- Step 1: Except for the last output layer's weights, designing the layers' weights according to the logical relations;
- Step 2: If X is the input train set, computing the last inner layers' output $F(X)$ based on X;
- Step 3: Using PSVM to compute the output layer's weights according to the target set Y;
- Step 4: Back propagate the error to the inner layers by the gradient-based learning and modify the input layers' weights.
- Step 5: Repeat the step 2 to step 4, until the output error is small enough.

5. Hybrid Design of Columnar Organization of a Neural Network Based on Fuzzy Logic and PSVM

In the neural science, the design of nonlinear dynamic neural networks to model bioneural experimental results is an intricate task. But with the help of fuzzy logic, we can design neural models similar to design binary digit networks. In our Hybrid designing approach(LPSVM), we firstly design neural networks with the help of fuzzy logic, and then we use PSVM to accomplish the learning for some concrete visual tasks.

Although there are already many neural model to simulate the functions of the primary visual cortex, they only focus on very limited function. Early works only try to address the problem of boundary detection by combining ideas and approaches from biological and computational vision [39-41], and the most recent works [1,3] are only for very simple pattern recognition tasks. In this experiment, we try to hybrid design a model of a columnar organization in the primary visual cortex which can separate haze from its background.

5.1. The Theory of Image Matting

According to Levin A *et al.* (2008), image matting refers to the problem of softly extracting the foreground object from a single input image. Formally, image matting methods take I as an input, which is assumed to be a composite of a foreground image F and a background B in a linear form and can be written as $I = \alpha F + (1-\alpha)B$. Closed form solution assumes that α is a linear function of the input image I in a small window $w: \alpha_i \approx aI_i + b, \forall i \in w$. Then to solve a spare linear system to get the alpha matte. Our neural fuzzy logical approach gets rid of the linear assumption between α and I. Instead, we try to introduce nonlinear relation between α and I:

$$\alpha_i = F(W_{I_w}) \tag{1.19}$$

here W_{I_w} is the image block included in the small window w. We take color or texture in local window as our

input feature, and the trimap image as the target. "Tripmap" means three kinds of regions, white denotes definite foreground region, black denotes definite background region and gray denotes undefined region. After training, the neural fuzzy logical network will generate the result of alpha matte. In the application of alpha matting, our method can remove the haze using dark channel prior as the trimap.

5.2. Neural System for Haze-Free Task with Columnar Organization

Many functions of the primary visual cortex are still unknown, but the columnar organization is well understood. The lateral geniculate nucleus (LGN) transfers information from eyes to brain stem and primary visual cortex (V1) [42]. Columnar organization of V1 plays an important role in the processing of visual information. V1 is composed of a grid $\left(1\times 1\,\mathrm{mm}^2\right)$ of hypercolumns (hc). Every hypercolumn contains a set of minicolumns (mc). Each hypercolumn analyzes information from one small region of the retina. Adjacent hypercolumns analyze information from adjacent areas of the retina. The recognition of our hypercolumns' system (see **Figure 2**) is started with the recognition orientation or simple structure of local patterns, then the trimap image is computed based on these local patterns. The hypercolumns is designed by LPSVM, the weights of 1st and 2nd layers are designed by fuzzy logic, and the weights of the 3rd layer are designed by PSVM to learn the trimap image.

5.2.1. The 1st Layer

Every minicolumn (**Figure 3**) in the 1st layer tries to change a 3×3 pixels' image block into a binary 3×3 pixels' texture pattern. The input image is normalized. This process needs 3×3 Hopfield neurons to focus a 3×3 small window, every neuron focuses only one pixel, and there are two kinds of fuzzy processing. The 1^{st} processing directly transforms every pixel's value to a fuzzy logical one by a sigmoid function, and the 2nd processing is also completed by a sigmoid function, the difference is that every boundary pixel's value subtracts with the center pixel's value before sending it to a sigmoid function. Such processing emphasizes the contrast of texture, and our experiments support this fact. These two processing scan be viewed as some kind preprocessing of input image. Every neuron in a 1st layer's minicolumn has only one input weight w_{ij} in **Figure 3**, which equals 1; when $\lambda \to +\infty$, the coefficient λ in Equation (1.17) changes the outputs from fuzzy values to binary numbers(see **Figure 4**).

5.2.2. The 2nd Layer

Every minicolumn in the 2nd layer works in the way described by Hopfield neuron equation as Equation (1.18) or Equation (1.17) and can be viewed as a hypercolumn of the 1^{st} layer minicolumns, which focuses on same small 3×3 window, and has some ability to recognize a definite shape (see **Figure 5**). If there are total q local small patterns, a hypercolumn in the 2nd layer contains q (in our system $q=256$ or 512) minicolumns of the 2nd layer, which have same receptive field, and try to recognize q local small patterns from q minicolumns of the 1st layer. For a $m\times n$ image, if adjacent windows are overlapped, then $(m-2)\times(n-2)$ hypercolumns are needed. In the plane of $(m-2)\times(n-2)$ hypercolumns, similar 2nd minicolumns' outputs create q small images sized $(m-2)\times(n-2)$ and defined as "minicolumn-images"; if an image size is 256×256, then 254×254 hypercolumns are used for every color R or G or B.

The input of every 2nd-layer minicolumn comes from the output of a 1^{st}-layer's minicolumn and has three different ways:

1. In a local image pattern recognition way(LIPW): every 2nd layer hypercolumn contains 512 2nd–layer's minicolumns, and inputs of these 2nd–layer's minicolumns come from a 1st–layer's minicolumn which focuses on a 3×3 small window and applies the first kind processing. Every 2nd–layer's minicolumn tries to classify the image block in this window into 512 binary texture patterns(BTP), e.g. eight important BTPs are shown in **Figure 6**. The pixel value is "1" for white and "0" for black. In this mode, 3×3 Hopfield neurons of the 1^{st} layer output a 3×3 vector, *i.e.*, a 3×3 fuzzy logical pattern of a BTP, which is computed by a Sigmoid function. When the coefficient λ in Equation (1.17) is large enough, after enough cycles, the output of the neural model in **Figure 3** changes a gray pattern to a binary pattern of LIPW.

2. In a local Binary Pattern operator simulating way (LBPW). Every 2nd–layer's hypercolumn in LBPW is similar to a 2nd–layer's hypercolumn in above LIPW, except that its focused 1^{st}–layer's minicolumns apply the second kind processing. A 2nd–layer's hypercolumn contains 256 2nd–layer's minicolumns which can be labeled by $LBP(x_C, y_C)$ in Equation (1.20). [43] introduced the Local Binary Pattern operator in 1996 as a mean of summarizing local gray-level structure. The operator takes a local neighborhood around each pixel, thresholds the pixels of the neighborhood at the value of the central pixel and uses the resulting binary-valued image patch as a local image descriptor. It was originally defined for neighborhoods, giving 8 bit codes based on the 8 pixels around the central one. Formally, the LBP operator takes the form:

$$LBP(x_C, y_C) = \sum_{n=0}^{7} 2^n S(i_n - i_C) \qquad (1.20)$$

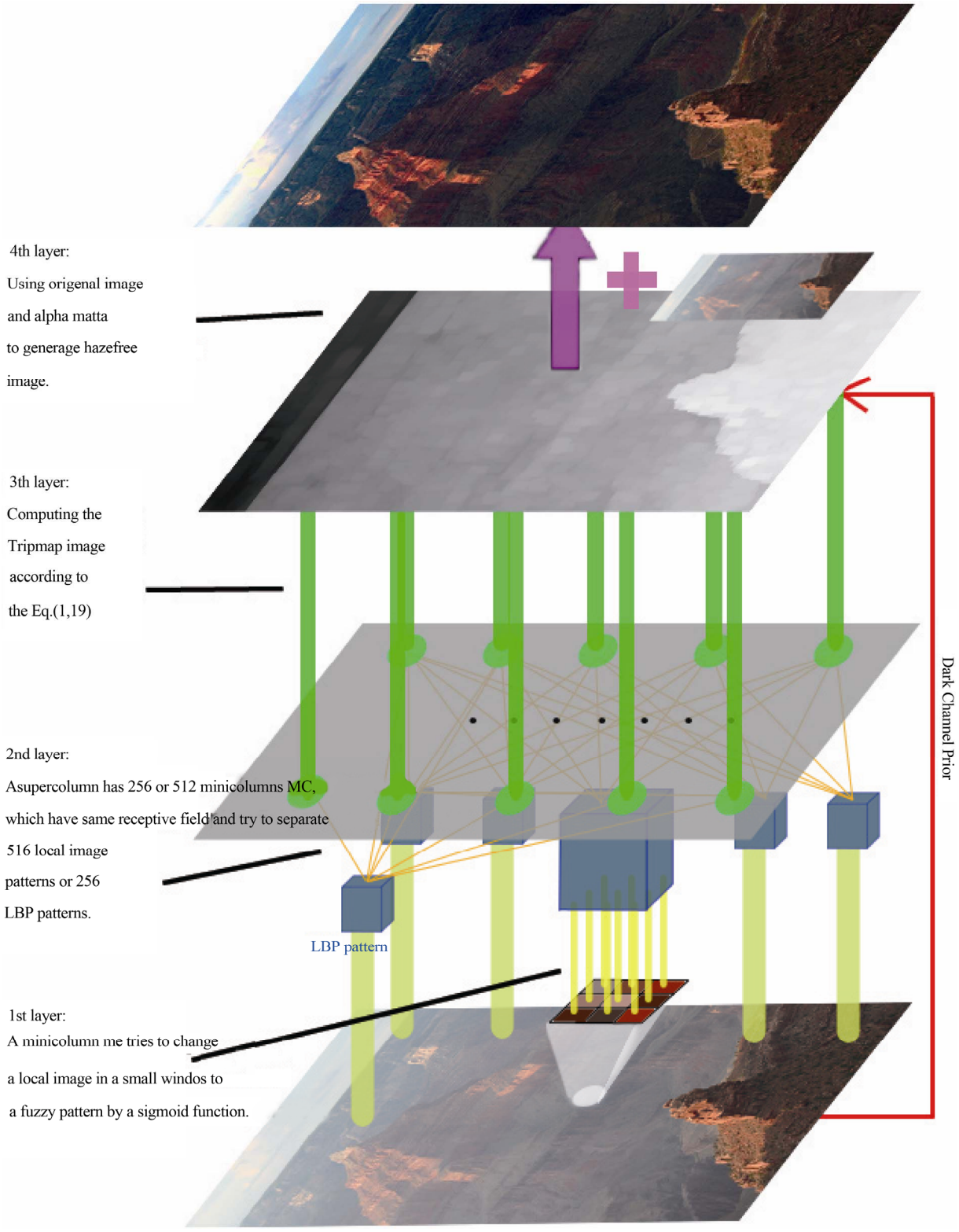

Figure 2. A 4 layers' structure of a columnar organization of V1 for haze-background separation.

where in this case n runs over the 8 neighbors of the central pixel c, $i_{c,k}$ and $i_{n,k}$ in Equation (1.20) are single color value Red $(k=1)$ or Gree $(k=2)$ or Blue $(k=3)$ at c and n, and $S(u)$ is a sigmoid function or step function, *i.e.* 1 if $u>0$ and 0 otherwise. In this mode, a 1^{st} layer's minicolumn outputs a

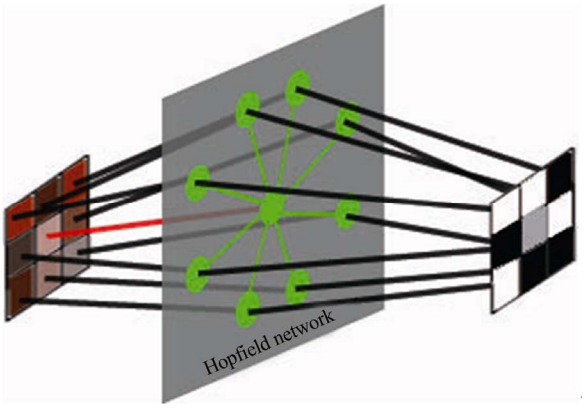

Figure 3. Every 1st layer minicolumn tries to change local images into binary texture patterns, for a 3×3 small window, a minicolumn in the 1st layer contains 9 Hopfield neurons, and every neuron focuses only one pixel.

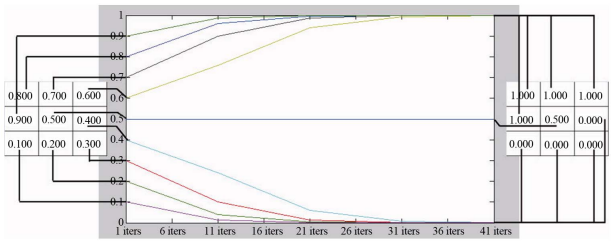

Figure 4. When the coefficient λ in Equation (1.17) is large enough, after enough cycles, the output of the neural model in Figure 3 changes a gray pattern to a binary pattern of LIPW.

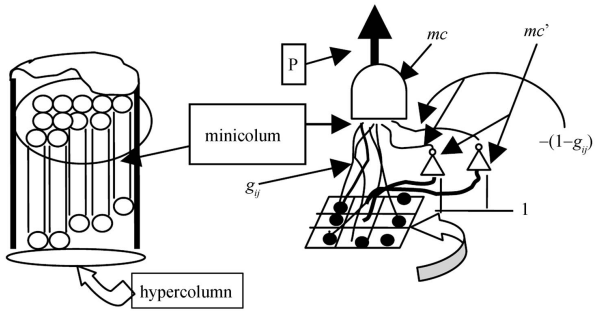

Figure 5. A hypercolumn in the 2nd layer contains q minicolumns which have same receptive field and try to recognize q definite small shapes. A *"and"* neuron is needed for every 2nd layer minicolumn.

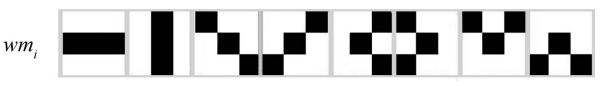

Figure 6. Every the 2nd layer's minicolumn contains 256 or 512 minicolumn which corresponds to 256 or 512 modules in above picture.

24-dimensional vector $O=(V_R,V_G,V_B)$, here

$$V_k=\left(S\left(i_{1,k}-i_{C,k}\right),S\left(i_{2,k}-i_{C,k}\right),\cdots,S\left(i_{8,k}-i_{C,k}\right)\right),$$

$$k=R,G,B$$

and $S\left(i_{l,k}-i_{C,k}\right), l=1,\cdots,8$ are the outputs from the 1st layer's minicolumns.

3. Hybrid LIPW and LBPW (LBIPW). In this approach, the boundary pixels' value are substracted by the center pixel's value in a 3×3 small window similar to LBPW, except that the center pixel's value is also sent to the input of every 2nd layer's minicolumn. So every 2nd layer hypercolumn also contains 512 2nd–layer's minicolumns,

In our system, a hypercolumn in the 2nd -layer contains 512 2nd layer's minicolumns for LIPW and LBIPW way, or 256 2nd layer's minicolumns for LBPW way for every color R,G or B. So a hypercolumn in the 2nd layer has a 512×3 dimensions output or 256×3 dimensions output. To recognize above two patterns is simple, a Hopfield neuron defined by Equation (1.17) is enough to recognize a 3×3 image. For example, the "$\cap$" shape in **Figure 5** can be described by a fuzzy logical formula (Equation (1.21)). The "and" operator for 9 inputs in Equation (1.21) can be created by a neuron mc (see . **Figure 5**). In Equation (1.21), every pixel P_{ij} has two states7 m_{ij} and $\bar{m}_{ij}$. Suppose the unified gray value of P_{ij} is g_{ij}, and an image module needs a high value g_{ij} at the place of m_{ij} and a low value at $\bar{m}_{ij}$. So the input for the neuron mc at m_{ij} is $I_{ij}=g_{ij}$, and at $\bar{m}_{ij}$ is $I_{ij}=-\left(1.0-g_{ij}\right)$. A not gate mc' is needed for $I_{ij}=-\left(1.0-g_{ij}\right)$.

$$P=m_{11}\wedge m_{12}\wedge m_{13}\wedge m_{21}\wedge m_{23}\wedge m_{31}\wedge m_{33}\wedge\bar{m}_{22}\wedge\bar{m}_{32} \tag{1.21}$$

In order to recognize a binary pattern, an "and" neuron with index i is needed (see **Figure 5**) for every 2nd-layer minicolumn, and the weights of this "and" neuron to the 1st-layer minicolumns are set as Equation (1.22), the corresponding threshold $T_i=5.1$, the parameter $\lambda=0.9$ and the coefficient of a in Equation (1.17) is set to 1.

$$w_{ij}=\begin{cases}1, & \text{if the } j\text{th bit of a binary pattern}=1\\ -1, & \text{if the } j\text{th bit of a binary pattern}=0\end{cases} \tag{1.22}$$

where for LIPW and LBIPW, $j=1,2,3,\cdots,9$; for LBPW,the center 1st-layer minicolumn is useless, so $j=1,2,3,\cdots,8$.

5.2.3. The 3rd Layer

The output of a hypercolumn in the 2nd layer, which has 3×256 or 3×512 dimensions, is transformed to the 3rd-layer minicolumns to compute the α_i in Equation (1.19) by psvm, the target is provided by so called dark channel prior which is computed by the approach mentioned in [8]. As the small windows focused by hypercolumns in the 2nd-layer are overlapped, the focuses of 3rd–layer's minicolumns are also overlapped.

5.2.4. The 4th Layer

There are two kinds minicolumns in this layer. At first, every 1st kind 4th-layer minicolumn, which is just a Hopfield neuron, computes a pixel value of the alpha matte image from the overlapped output of minicolumns in the 3rd layer, then the 2nd kind minicolumn tries to remove the haze from original image. A 2nd kind minicolumn in the 4th layer computes a pixel of a haze free image according to the Equation (1.23)

$$\begin{aligned} I_i(x) &= J_i(x) \oplus_f A_i \\ &= F_{\oplus_f}\left(J_i(x), A_i, \alpha_i(x), (1-\alpha_i(x))\right) \\ &= \min\left(q, \alpha_i(x)\cdot J_i(x) + (1-\alpha_i(x))A_i\right) \end{aligned} \quad (1.23)$$

where q is max gray or RGB value of a pixel, and where J_i is the haze free image, I_i is the original image, A_i is the global atmospheric light which can be estimated from dark channel prior, α_i is the alpha matte generated by 3rd layer. We can use back propagation approach to compute pixels' value $J_i(x)$ given the haze image pixel value $I_i(x)$. For the sake of simplicity, we directly use the Equation (1.24) mentioned by [8] to compute the haze free image.

$$J_i(x) = \frac{I_i(x) - A_i}{\max\left(\alpha_i(x), \alpha_0\right)} + A_i \quad \text{b} \quad (1.24)$$

where J_i is the haze free image, I_i is the original image, A_i is the global atmospheric light which can be estimated from dark channel prior, α_i is the alpha matte generated by 3rd layer, and α_0 is a threshold, a typical value is 1. In order to compute Equation (1.24).

5.3. Experiments Result

1. Experiment about the ability of a 2nd–layer's minicolumn

This experiment is about the ability of a 2nd-layer's minicolumn to recognize a local pattern in the way of LIPW. Here the input color image is transformed to gray image by Equation (1.25).

$$\text{gray} = 1.0 - \left(wr \times r + wg \times g + wb \times b\right)/256 \quad (1.25)$$

The minicolumn is running in the iterative mode.

The effect of threshold T_i in Equation (1.1) is to control the function of a neural cell from logical "or" to "and". L in **Figure 7(a)** is a horizontal line with a one pixel width.

Figures 7(b) and **(c)** show the outputs of 2 minicolumns after repeating 3 steps. The models m_0 and m_1 have 3×3 pixels. The 1st minicolumn $(m0)$ tries to recognize a horizontal bar wm_0 in **Figure 6**, and the 2nd minicolumn $(m1)$ tries to recognize a vertical bar wm_1. These two minicolumns try to recognize L at three different positions $P0$, $P1$ and $P2$. At $P0$, the focus of these 2 minicolumns is just upon the line L; at $P1$, these 2 minicolumns focus the nearby of L; at $P2$, the line L is out of the focus of these 2 minicolumns.

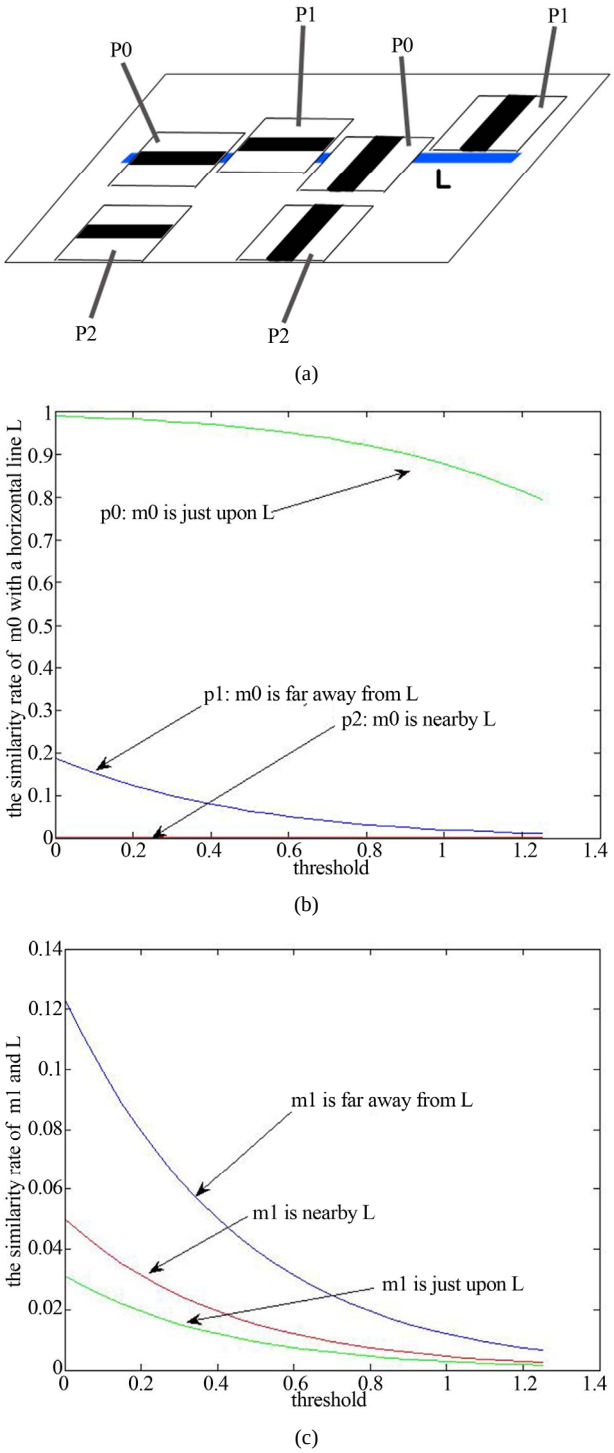

Figure 7. The threshold can control the function of a Hopfield neural cell from logical "or" to "and". b) using a horizontal model m0 to recognize a horizontal line L; c) using a vertical model m1 to recognize a horizontal line L

These two minicolumns are constructed by a fuzzy formula similar to Eqaution (1.21). In the experiment, the threshold T_i changes from 0.0 to 1.26. The threshold can control the function of a neural cell from logical "or" to "and", when T_i is too small, the neuron works in the way of logic "or", so two minicolumns both output high value at $P0$.

Same as Equation (1.21), the minicolumn of $m0$ has a inhibit region when the weights of $m0$ have negative values. At position $P1$, L is dropped into the inhibit region of model $m0$, so $m0$ has a lowest matching rate at p1 where L is nearby than $p0$ and $p2$. As "vertical" has a reverse meaning of "horizontal", the curve of m0 at $p0$ is totally different with the curve of $m1$ at $p0$.

An optimized threshold T_i can be selected. As a horizontal bar and a vertical bar can be viewed as a couple of opposite shapes, the output of the minicolumn $m0$ is opposite to m1's output in the task of recognize a horizontal bar or a vertical bar, so the most suitable threshold for logical operator "and" can be selected by Equation (1.26).

$$T_i = \arg\max_{\theta}\left(fr\left(m_0\right) - fr\left(m_1\right)\right) = 0.6 \qquad (1.26)$$

Here $fr\left(m_i\right)$ is the fitting rate (output) at the place P1 of the minicolumn m_i.

2. The Haze-Free Experiment Result

(a) The Haze-Free and texture information entropy

Texture information can give out a rough measure about the effect of haze-freeing, we use the entropy of the texture histogram to measure the effect of deleting haze from images. The entropy of the histogram is described in Equation (1.27).

Haze makes the texture of an image unclear, so theoretically speaking, haze removing will increase the entropy of the texture histogram.

$$\text{Entropy}: H = -\sum_{i=0}^{G-1} p(i)\log_2\left[p(i)\right] \qquad (1.27)$$

The $p(i)$ denote the rate of each pattern in histogram. In general $p(i)$ define in Eqaution (1.28). Here patterns we use are the LBPs in Equaiton (1.20), where $S\left(i_n - i_C\right) = 1$, if $i_n > i_C + 10$ else $S\left(i_n - i_C\right) = 0$.

$$p(i) = h(i)/NM, i = 0,1,\cdots,G-1 \qquad (1.28)$$

In **Figures 8 (a)-(e)** are the results of LBPW, LBIPW, LIPW, the linear mode by [8], and the original image respectively. From **Figure 8**, we can see that the texture structure in the waist of a mountain becomes vaguer from LBPW, LBIPW, LIPW to LMKH. For the sake of the 2nd kind processing in the 1st-layer's minicolumns pays much more attention to the contrast, LBPW has the highest ability to remove the haze, LBPW and LIPW are complementary approaches, LBIPW, which is the cooperation of them, has a similar ability as the linear approach proposed by [8].

According to the results showed in the **Table 1**, which are about texture information entropy of the image, we can see that the texture information entropy is increased after haze-free processing, so our approaches have higher ability to increase the texture information entropy than the linear approach proposed by [8]. Theoretically speaking, LBPW is a pure texture processing, so LBPW has a highest value, LIPW is much more weaker than LBPW, LBIPW is the hybrid of LBPW and LIPW, so it has a average ability. The texture information entropy of the Area1 correctly reflects this fact. But for the Area2, as it already has a clearest texture structure in the original image, the deleting of haze may cause overdone. The texture information is over emphasized by LBPW in the Aera2, so it has a lowest texture information entropy and almost becomes a dark area. This fact means that overtreatment is more easier to appear in a non linear processing than a linear one in the haze-free task.

(b) The effect about the degree of fuzzyness (**Figure 9**)

Just as the theorem 1 mentioned above, the parameter λ in Eqaution (1.17) can control the fuzzyness of a Hopfield

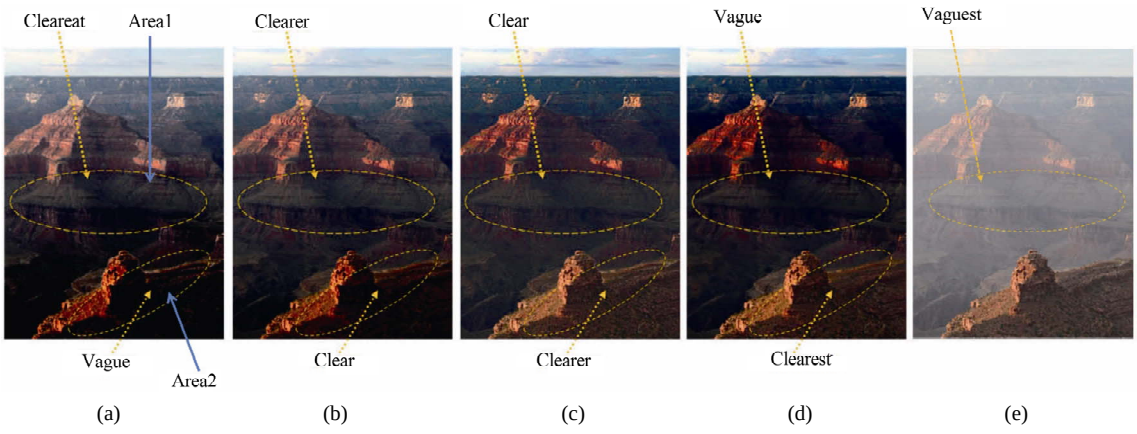

Figure 8. The processing result of neural system for visual haze-free task. a) LBPW; b) LBIPW; c) LIPW; d) LMKH; e) Original.

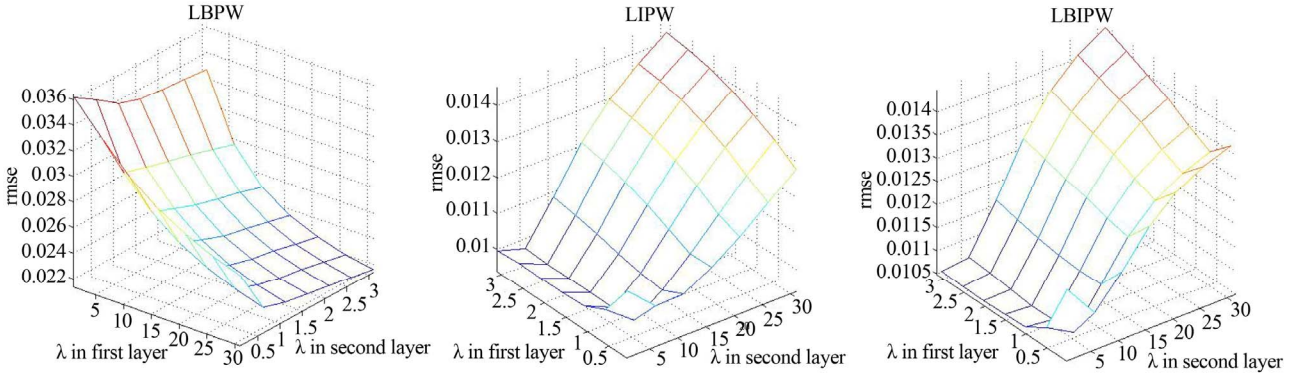

Figure 9. Rmse affected by_in first layer and second layer, we set Threshold in Equation (1.17) = 1.6.

Table 1. The texture information entropy of the image blocks (Area1: the waist of a mountain; Area2: Right bottom corner) in the Figure 8.

Area	LBPW	LBIPW	LIPW	LMKH	Original
Area1	5.4852	5.2906	5.1593	4.8323	1.0893
Area2	6.1091	10.3280	10.2999	9.1759	8.3718

neuron, when the parameter λ in Eqaution (1.17) tends to infinite, a Hopfield neuron behaves from a fuzzy logical formula to a binary logical formula. This experiment is about the relation among the precision (*rmse*) of PSVM learning and λ parameters in the first and second layer. LBPW is a pure texture processing and pays much more attention to the contrast of an image's nearby pixels, a set of large λ is necessary for a low *rmse*, which cor- responds to binary logic; but LBIPW and LIPW appear to prefer fuzzy logic for a set of small λ when *rmse* is small. A possible explanation for this fact is that LBP proposed by T. Ojala *et al*. (1996) is binary, not fuzzy, and has a sound classification ability for image under- standing under binary pattern, but LBIPW and LIPW are not binary, they have fuzzy information at least for the center pixel of a 3×3 small window.

6. Discussion

It is very difficult to design or analyze a large-scale nonlinear neural network. Fortunately, almost all neural models which are described by the first order differential equations can be simulated by Hopfield neural models or fuzzy logical functions with Weighted Bounded operator. We can find fuzzy logical frameworks for almost all neural networks, so it becomes possible to debug thousands of parameters of a huge neural system with the help of fuzzy logic, for more fuzzy logic can help us to find useful feature for visual tasks, e.g. haze-free.

7. Acknowledgements

National Natural Science Foundation of China (No. 61072085, 61035003,61202212, 60933004),

Supported by the National Program on Key Basic Research Project (973 Program) (No. 2013CB329502),

National High-tech R&D Program of China (863 Program) (No.2012AA011003),

National Science and Technology Support Program (2012BA107B02).

REFERENCES

[1] H. De Garis, C. Shuo, B. Goertzel and L. Ruiting, "A World Survey of Artificial Brain Projects, Part 1: Large-Scale Brain Simulations," *Neurocomputing*, Vol. 74, No. 1, 2010, pp. 3-29. http://dx.doi.org/10.1016/j.neucom.2010.08.004

[2] M. Djurfeldt, M. Lundqvist, C. Johansson, M. Rehn, O. Ekeberg and A. Lansner, "Brain-Scale Simulation of the Neocortex on the Ibm Blue Gene/l Supercomputer," *IBM Journal of Research and Development*, Vol. 52, No. 1-2, 2008, pp. 31-41.

[3] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, C. Tang and D. Rasmussen, "A Large-Scale Model of the Functioning Brain," *Science*, Vol. 338, No. 6111, 2012, pp. 1202-1205.

[4] J. O'Kusky and M. Colonnier, "A Laminar Analysis of the Number of Neurons, Glia, and Synapses in the Visual Cortex (Area 17) of Adult Macaque Monkeys," *Journal of Comparative Neurology*, Vol. 210, No. 3, 1982, pp. 278-290. http://dx.doi.org/10.1002/cne.902100307

[5] K. Hirota and W. Pedrycz, "Or/And Neuron in Modeling Fuzzy Set Connectives," *IEEE Transactions on Fuzzy Systems*, Vol. 2, No. 2, 1994, pp. 151-161. http://dx.doi.org/10.1109/91.277963

[6] W. Pedrycz and F. Gomide, "An Introduction to Fuzzy Sets: Analysis and Design," The MIT Press, Massachusetts, 1998.

[7] L. Zhaoping, "Pre-Attentive Segmentation and Correspondence in Stereo," *Philosophical Transactions of the Royal Society of London. Series B*: *Biological Sciences*, Vol. 357, No. 1428, 2002, pp. 1877-1883. http://dx.doi.org/10.1098/rstb.2002.1158

[8] K. He, J. Sun and X. Tang, "Single Image Haze Removal Using Dark Channel Prior," *IEEE Transactions on Pat-*

tern Analysis and Machine Intelligence, Vol. 33, No. 12, 2011, pp. 2341-2353. http://dx.doi.org/10.1109/TPAMI.2010.168

[9] R. FitzHugh, "Impulses and Physiological States in Theoretical Models of Nerve Membrane," *Biophysical Journal*, Vol. 1, No. 6, 1961, pp. 445-466. http://dx.doi.org/10.1016/S0006-3495(61)86902-6

[10] H. R. Wilson and J. D. Cowan, "Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons," *Biophysical Journal*, Vol. 12, No. 1, 1972, pp. 1-24. http://dx.doi.org/10.1016/S0006-3495(72)86068-5

[11] J. Hindmarsh and R. Rose, "A Model of Neuronal Bursting Using Three Coupled First Order Differential Equations," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, Vol. 221, No. 1222, 1984, pp. 87-102. http://dx.doi.org/10.1098/rspb.1984.0024

[12] J. J. Hopfield, D. W. Tank, *et al*., "Computing with Neural Circuits: A Model," *Science*, Vol. 233, No. 4764, 1986, pp. 625-633. http://dx.doi.org/10.1126/science.3755256

[13] H. Hu and Z. Shi, "The Possibility of Using Simple Neuron Models to Design Brain-Like Computers," In: *Advances in Brain Inspired Cognitive Systems*, Springer, Shenyang, 2012, pp. 361-372. http://dx.doi.org/10.1007/978-3-642-31561-9_41

[14] H. Abarbanel, M. I. Rabinovich, A. Selverston, M. Bazhenov, R. Huerta, M. Sushchik, and L. Rubchinskii, "Synchronisation in Neural Networks," *Physics-Uspekhi*, Vol. 39, No. 4, 1996 pp. 337-362. http://dx.doi.org/10.1070/PU1996v039n04ABEH000141

[15] Z. Li, "A Neural Model of Contour Integration in the Primary Visual Cortex," *Neural Computation*, Vol. 10, No. 4, 1998, pp. 903-940. http://dx.doi.org/10.1162/089976698300017557

[16] E. Fransen and A. Lansner, "A Model of Cortical Associative Memory Based on a Horizontal Network of Connected Columns," *Network*: *Computation in Neural Systems*, Vol. 9, No. 2, 1998, pp. 235-264. http://dx.doi.org/10.1088/0954-898X/9/2/006

[17] H. Sun, L. Liu and A. Guo, "A Neurocomputational Model of Figure-Ground Discrimination and Target Tracking," *IEEE Transactions on Neural Networks*, Vol. 10, No. 4, 1999, pp. 860-884. http://dx.doi.org/10.1109/72.774238

[18] H. B. Barlow, C. Blakemore and J. D. Pettigrew, "The Neural Mechanism of Binocular Depth Discrimination," *The Journal of Physiology*, Vol. 193, No. 2, 1967, p. 327.

[19] D. H. Hubel and T. N. Wiesel, "Stereoscopic Vision in Macaque Monkey: Cells Sensitive to Binocular Depth in area 18 of the Macaque Monkey Cortex," *Nature*, Vol. 225, 1970, pp. 41-42. http://dx.doi.org/10.1038/225041a0

[20] K. S. Rockland and J. S. Lund, "Intrinsic Laminar Lattice Connections in Primate Visual Cortex," *Journal of Comparative Neurology*, Vol. 216, No. 3, 1983, pp. 303-318. http://dx.doi.org/10.1002/cne.902160307

[21] C. D. Gilbert and T. N. Wiesel, "Clustered Intrinsic Connections in Cat Visual Cortex," *The Journal of Neuroscience*, Vol. 3, No. 5, 1983, pp. 1116-1133.

[22] R. von der Heydt, H. Zhou, H. S. Friedman, *et al*., "Representation of Stereoscopic Edges in Monkey Visual Cortex," *Vision Research*, Vol. 40, No. 15, 2000, pp. 1955-1967. http://dx.doi.org/10.1016/S0042-6989(00)00044-4

[23] J. S. Bakin, K. Nakayama and C. D. Gilbert, "Visual Responses in Monkey Areas v1 and v2 to Three-Dimensional Surface Configurations," *The Journal of Neuroscience*, Vol. 20, No. 21, 2000, pp. 8188-8198.

[24] L. A. Zadeh, "Information and Control," *Fuzzy Sets*, Vol. 8, No. 3, 1965, pp. 338-353.

[25] S. S. Haykin, "Neural Networks: A Comprehensive Foundation," Prentice Hall Englewood Cliffs, 2007.

[26] J. Nagumo, S. Arimoto and S. Yoshizawa, "An Active Pulse Transmission Line Simulating Nerve Axon," *Proceedings of the IRE*, Vol. 50, No. 10, 1962, pp. 2061-2070. http://dx.doi.org/10.1109/JRPROC.1962.288235

[27] A. L. Hodgkins and A. F. Huxley, "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve," *American Journal of Physiology*, Vol. 117, No. 4, 1952, pp. 500-544.

[28] V. der Pol B, "The Nonlinear Theory of Electrical Oscillations," *Proceedings of the Institute of Radio Engineers*, Vol. 22, No. 9, 1934, pp. 1051-1086.

[29] J. A. Connor, D. Walter and R. McKowN, "Neural Repetitive Firing: Modifications of the Hodgkin-Huxley Axon Suggested by Experimental Results from Crustacean Axons," *Biophysical Journal*, Vol. 18, No. 1, 1977, pp. 81-102. http://dx.doi.org/10.1016/S0006-3495(77)85598-7

[30] C. Morris and H. Lecar, "Voltage Oscillations in the Barnacle Giant Muscle Fiber," *Biophysical Journal*, Vol. 35, No. 1, 1981, pp. 193-213. http://dx.doi.org/10.1016/S0006-3495(81)84782-0

[31] T. R. Chay, "Chaos in a Three-Variable Model of an Excitable Cell," *Physica D*: *Nonlinear Phenomena*, Vol. 16, No. 2, 1985, pp. 233-242. http://dx.doi.org/10.1016/0167-2789(85)90060-0

[32] T. R. Chay, "Electrical Bursting and Intracellular Ca2+ Oscillations in Excitable Cell Models," *Biological Cybernetics*, Vol. 63, No. 1, 1990, pp. 15-23. http://dx.doi.org/10.1007/BF00202449

[33] D. Golomb, J. Guckenheimer and S. Gueron, "Reduction of a Channel-Based Model for a Stomatogastric Ganglion Lpneuron," *Biological Cybernetics*, Vol. 69, No. 2, 1993, pp. 129-137. http://dx.doi.org/10.1007/BF00226196

[34] H. R. Wilson and J. D. Cowan, "A Mathematical Theory of the Functional Dynamics of Cortical and Thalamic Nervous Tissue," *Kybernetik*, Vol. 13, No. 2, 1973, pp. 55-80. http://dx.doi.org/10.1007/BF00288786

[35] F. Buchholtz, J. Golowasch, I. R. Epstein and E. Marder, "Mathematical Model of an Identified Stomatogastric Ganglionneuron," *Journal of Neurophysiology*, Vol. 67, No. 2, 1992, pp. 332-340.

[36] A. Burkitt, "A Review of the Integrate-and-Fire Neuron Model: I. Homogeneous Synaptic Input," *Biological Cybernetics*, Vol. 95, No. 1, 2006, pp. 1-19. http://dx.doi.org/10.1007/s00422-006-0068-6

[37] H.-X. Li and C. P. Chen, "The Equivalence between Fuzzy Logic Systems and Feedforward Neural Networks,"

IEEE Transactions on Neural Networks, Vol. 11, No. 2, 2000, pp. 356-365. http://dx.doi.org/10.1109/72.839006

[38] G. Fung and O. L. Mangasarian, "Proximal Support Vector Machine Classifiers," *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, 2011, pp. 77-86. http://dx.doi.org/10.1145/502512.502527

[39] D. Wielaard, M. Shelley, D. McLaughlin and R. Shapley, "How Simple Cells Are Made in a Nonlinear Network Model of the Visual Cortex," *The Journal of Neuroscience*, Vol. 21, No. 14, 2001, pp. 5203-5211.

[40] J. Wielaard and P. Sajda, "Simulated Optical Imaging of Orientation Preference in a Model of V1," *Proceedings of First International IEEE EMBS Conference on Neural Engineering*, Capri Island, 20-22 March 2003, pp. 499-502.

[41] Kokkinos, R. Deriche, O. Faugeras and P. Maragos, "Computational Analysis and Learning for a Biologically Motivated Model of Boundary Detection," *Neurocomputing*, Vol. 71, No. 10, 2008, pp. 1798-1812. http://dx.doi.org/10.1016/j.neucom.2007.11.031

[42] V. B. Mountcastle, "The Columnar Organization of the Neocortex," *Brain*, Vol. 120, No. 4, 1997, pp. 701-722. http://dx.doi.org/10.1093/brain/120.4.701

[43] T. Ojala, M. Pietikäinen and D. Harwood, "A Comparative Study of Texture Measures with Classification Based on Featured Distributions," *Pattern Recognition*, Vol. 29, No. 1, 1996, pp. 51-59. http://dx.doi.org/10.1016/0031-3203(95)00067-4

Appendix A

A Hopfield neuron can approximately simulate Bounded operator.

Bounded operator $F\left(\oplus_f, \otimes_f\right)$

Bounded product $p \otimes_f q = \max\left(0, p+q-1\right)$,

Bounded sum $p \oplus_f q = \min\left(1, p+q\right)$.

Based on Eqaution (1.1)), the membrane potential's fixed point under input I_k is $\bar{U}_i = \sum_k w_{ik} I_k / a_i$ and the output at the fixed point is $\bar{V}_i = 1/\left(\exp\left(-U_i + T_i\right)+1\right)$.

If there are only two inputs $I_1, I_2 \left(I_1, I_2 \in [0,1]\right)$ and we set $a_i = 1.0$, $w_1 = 1.0$ and $w_2 = 1.0$, then $\bar{U}_i = I_1 + I_2$.

Now we try to prove that the Bounded operator $F\left(\oplus_f, \otimes_f\right)$ is the best fuzzy operator to simulate neural cells described by (3) and the threshold T_i can change the neural cell from the bounded operator $\oplus_f$ to $\otimes_f$ by analyzing the output at the fixed point $\bar{V}_i = 1/\left(\exp\left(-U_i + T_i\right)+1\right)$. If $C > 0$ is a constant and $\bar{U}_i = I_1 + I_2 \geq C$, then $1/\left(\exp\left(-C + T_i\right)+1\right) \leq \bar{V}_i < 1$. When $\bar{U}_i = I_1 + I_2 \to +\infty$ $\bar{V}_i \to 1$, so in this case, if C is large enough, $\bar{V}_i \approx 1$. If $-C \leq \bar{U}_i = I_1 + I_2 \leq C$, then

$$1/\left(\exp\left(C + T_i\right)+1\right) \leq \bar{V}_i \leq 1/\left(\exp\left(-C + T_i\right)+1\right),$$

according to equation (a). We can select a T_i, that makes

$$\left| T_i + \sum_{j=2}^{\infty} \left(-U_i + T_i\right)^j / j! - \sum_{k=2}^{\infty} (-1)^k \exp\left(-k\left(\bar{U}_i \_ T_i\right)\right) \right|$$

small enough, then $\bar{V}_i \approx I_1 + I_2$.

$$\begin{aligned}
\bar{V}_i &= 1/\left(\exp\left(-U_i + T_i\right)+1\right) \\
&= 1 - \exp\left(-\bar{U}_i + T_i\right) + \sum_{k=2}^{\infty} (-1)^k \exp\left(-k\left(\bar{U}_i - T_i\right)\right) \\
&= 1 - 1 + \bar{U}_i - T_i - \sum_{j=2}^{\infty} \left(-U_i + T_i\right)^j / j! \\
&\quad + \sum_{k=2}^{\infty} (-1)^k \exp\left(-k\left(\bar{U}_i _ T_i\right)\right) \\
&= \bar{U}_i - T_i - \sum_{j=2}^{\infty} \left(-U_i + T_i\right)^j / j! \\
&\quad + \sum_{k=2}^{\infty} (-1)^k \exp\left(-k\left(\bar{U}_i _ T_i\right)\right).
\end{aligned}$$

So in this case, $\bar{V}_i \approx I_1 \oplus_f I_2 = \min\left(1, I_1 + I_2\right)$.

Similarly, if $\bar{U}_i = I_1 + I_2 \to -\infty$ $\bar{V}_i \to 0$. So when C is large enough and $\bar{U}_i = I_1 + I_2 \leq -C < 0$, then $\bar{V}_i \approx 0$. When $-C \leq \bar{U}_i = I_1 + I_2 \leq C$, if we select a suitable T_i which makes

$$T_i + \sum_{j=2}^{\infty} \left(-U_i + T_i\right)^j / j! - \sum_{k=2}^{\infty} (-1)^k \exp\left(-k\left(\bar{U}_i - T_i\right)\right) \approx 1,$$

then $\bar{V}_i \approx I_1 \otimes_f I_2 = \max\left(0, I_1 + I_2 - 1\right)$.

Based on above analysis, the Bounded operator fuzzy system is suitable for neural cells described by Equation (1.1) when $a_i = 1.0$, $w_1 = 1.0$ and $w_2 = 1.0$. For arbitrary positive a_i, w_1 and w_2, we can use corresponding q-value weighted universal fuzzy logical function based on Bounded operator to simulate such kind neural cells. If a weight w is negative , a N-norm operator $N(x) = 1 - x$ should be used.

Experiments done by scanning the whole region of $\left(I_1, I_2\right)$ in $[0,1]^2$ to find the suitable coefficients for $\oplus_f$ and $\otimes_f$ show that above analysis is sound. We denote the input in (5b) as $\vec{x}(t) = \left(I_1(t), I_2(t)\right)$. The "error" for $\oplus_f$ and "errAnd" for $\otimes_f$ are shown in Figure A as the solid line and the dotted line respectively. In **Figure 10**, the threshold T_i is scanned from 0 to 4.1 with step size 0.01. The best T_i in Equation (4) for $\otimes_f$ is 2.54 and the best T_i in Equation (4) for $\oplus_f$ is 0, when $a = 1.0$, $w_1 = 1.0$ and $w_2 = 1.0$. In this case the "errOr" and "errAnd" is less than 0.01. Our experiments show that suitable T_i can be found. So in most cases, the bounded operator $F\left(\oplus_f, \otimes_f\right)$ mentioned above is the suitable fuzzy logical framework for the neuron defined by Equation (3). If the weight $0 < w_1$ and $0 < w_2$, we should use a q-value weighted bounded operator $F\left(\oplus_f, \otimes_f\right)$ to represent above neuron.

Appendix B

It is easily to see $\oplus_f$ follows the associative condition and

$$x_1 \oplus_f x_2 \oplus_f x_3 \oplus_f \cdots \oplus_f x_n = \min\left(q, \sum_{1 \leq i \leq n} w_i x_i\right).$$

For $\otimes_f$, we can prove the associative condition is

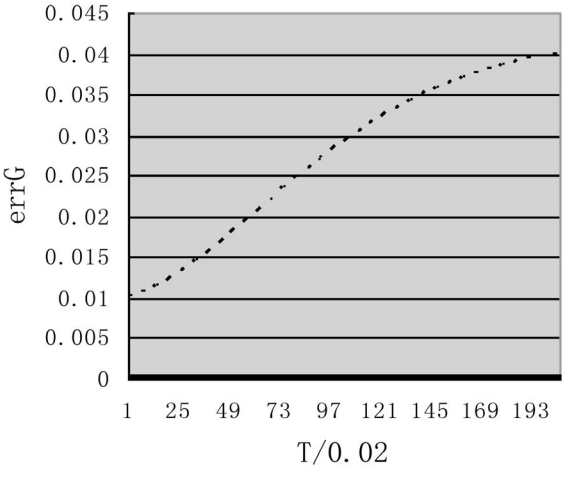

Figure 10. Simulating fuzzy logical and-or by changing thresholds of neural cells. The X-axis is the threshold value divided by 0.02, the Y-axis is errG. The real line is perrAndq between $I_1 \oplus_f I_2$ and $\bar{V}_i$, and the dot line is the perrorq between $I_1 \oplus_f I_2$ and $\bar{V}_i$.

hold also. The proof is listed as below:

If $w_1 p_1 + w_2 p_2 - (w_1 + w_2 - 1)q \geq 0$, we have:

$$\begin{aligned}
&(p_1 \otimes_f p_2) \otimes_f p_3 \\
&= F_{\otimes_f}\left(F_{\otimes_f}(p_1, p_2, w_1, w_2), p_3, 1, w_3\right) \\
&= F_{\otimes_f}\left(w_1 p_1 + w_2 p_2 - (w_1 + w_2 - 1)q, p_3, 1, w_3\right) \\
&= \max\left(0, w_1 p_1 + w_2 p_2 - (w_1 + w_2 - 1)q \right. \\
&\qquad \left. + w_3 p_3 - (1 + w_3 - 1)q\right) \\
&= \max\left(0, w_1 p_1 + w_2 p_2 + w_3 p_3 - (w_1 + w_2 + w_3 - 1)q\right)
\end{aligned} ;$$

if $w_1 p_1 + w_2 p_2 - (w_1 + w_2 - 1)q < 0$, we have

$$\begin{aligned}
&(p_1 \otimes_f p_2) \otimes_f p_3 \\
&= F_{\otimes_f}\left(F_{\otimes_f}(p_1, p_2, w_1, w_2), p_3, 1, w_3\right) \\
&= F_{\otimes_f}(0, p_3, 1, w_3) \\
&= \max\left(0, 0 + w_3 p_3 - (1 + w_3 - 1)q\right) \\
&= \max\left(0, w_3 p_3 - w_3 q\right) \overset{\text{for } 0 \leq p_3 \leq q}{=} 0 \\
&= \max\left(0, w_1 p_1 + w_2 p_2 + w_3 p_3 - (w_1 + w_2 + w_3 - 1)q\right)
\end{aligned} ;$$

So

$$\begin{aligned}
&(p_1 \otimes_f p_2) \otimes_f p_3 = p_1 \otimes_f (p_2 \otimes_f p_3) \\
&= \max\left(0, w_1 p_1 + w_2 p_2 + w_3 p_3 - (w_1 + w_2 + w_3 - 1)q\right)
\end{aligned}.$$

By inductive approach, we can prove that $\otimes_f$ also follows the associative condition and

$$\begin{aligned}
&x_1 \otimes_f x_2 \otimes_f x_3 \otimes_f \cdots \otimes_f x_n \\
&= \max\left(0, \sum_{1 \leq i \leq n} w_i x_i - \left(\sum_{1 \leq i \leq n} w_i - 1\right)q\right).
\end{aligned}$$

For more if we define $N(p) = q - p$ (usually, a negative weight w_i corresponds a N-norm), above weighted bounded operator $F(\oplus_f, \otimes_f)$ follows the Demorgan Law, *i.e.*

$$\begin{aligned}
&N(x_1 \oplus_f x_2 \oplus_f x_3 \oplus_f \cdots \oplus_f x_n) \\
&= q - \min\left(q, \sum_{1 \leq i \leq n} w_i x_i\right) \\
&= \max\left(0, q - \sum_{1 \leq i \leq n} w_i x_i\right) \\
&= \max\left(0, \sum_{1 \leq i \leq n} w_i (q - x_i) - \left(\sum_{1 \leq i \leq n} w_i - 1\right)q\right) \\
&= N(x_1) \otimes_f N(x_2) \otimes_f N(x_3) \otimes_f \cdots \otimes_f N(x_n)
\end{aligned}$$

