A Competitive Layer Model for Cellular Neural Networks [☆]

Wei Zhou[a,*], Jacek M. Zurada[b]

[a] *College of Computer Science and Technology, Southwest University for Nationalities, Chengdu 610041, P. R. China.*
[b] *Computational Intelligence Laboratory, Electrical and Computer Engineering Department, University of Louisville, Louisville, KY, 40292, USA.*

Abstract

This paper discusses a Competitive Layer Model (CLM) for a class of recurrent Cellular Neural Networks (CNN) from continuous-time type to discrete-time type. The objective of the CLM is to partition a set of input features into salient groups. The complete convergence of such networks in continuous-time type has been discussed first. We give a necessary condition, and a necessary and sufficient condition, which allow the CLM performance existence in our networks. We also discuss the properties of such networks of discrete-time type, and propose a novel CLM iteration method. Such method shows similar performance and storage allocation but faster convergence compared with previous CLM iteration method [19]. Especially for a large scale network with many features and layers, it can significantly reduce the computing time. Examples and simulation results are used to illustrate the developed theory, the comparison between two CLM iteration methods, and the application in image segmentation.

Key words: Competitive Layer Model, Cellular Neural Networks, Continuous-time recurrent neural networks, Discrete-time recurrent neural

[☆] The research reported here was supported by National Science Foundation of China under Grant 61105061; the Fundamental Research Funds for the Central Universities, Southwest University for Nationalities (11NZYTD04); National Science Foundation of China under Grant 60973070; and the National Research Foundation for the Doctoral Program of Higher Education of China (20090185120009).

[*] Corresponding author.

Email addresses: wei.zhou.swun@gmail.com (Wei Zhou), jmzura02@louisville.edu (Jacek M. Zurada) networks

---

1. Introduction

For human being's visual perception, perceptual grouping can be defined as the ability to detect structural layout of visual objects. This phenomenon was first studied in the 1920ies by the Gestalt school of psychology and one of their important theories is Gestalt law [11]. By virtue of some Gestalt laws, such as proximity, symmetry, and continuity, human can detect groups in a set of objects. In computer vision, this grouping process can be considered as a procedure for feature binding, which aims at binding some related features into common groups, so as to separate those groups originating from different features [16], [17].

The Competitive Layer Model (CLM) was first proposed to solve spatial feature binding and sensory segmentation problems by Ritter [13], [14]. This model is based on the combination of competitive and cooperative processes in a recurrent neural network (RNN) architecture, which can partition a set of input features into salient groups. Due to competitive interactions among layers, each feature is unambiguously assigned to one layer and feature binding is achieved by a collection of competitive layers. In [19], Wersing et al. designed a continuous-time CLM RNN with linear threshold (LT) neurons for feature binding and sensory segmentation. S. Weng et al. also proposed a hybrid learning method based on CLM [18]. In [23], Zhang Yi proposed to use continuous-time Lotka-Volterra recurrent neural networks to implement the CLM, and proved that the set of stable attractors of such networks equals the set of minimum points of the CLM energy function in the nonnegative orthant. However, most of them focus on studying the CLM properties of different networks, or exploring possible application field, finding a CLM algorithm more efficiently is still a challenge.

The CLM networks can be considered as a kind of multistable Winner-Take-All (WTA) networks. A multi-stable network can have multiple stable equilibriums, while a mono-stable one has always only one stable equilibrium. Those traditional WTA neural networks are almost mono-stable, in which only one neuron among all neurons can be the final 'winner'. The multistablilty property can provide an interesting way to mediate WTA competition between groups of neurons, so the final winner will be a group of neurons. More discussion about multistablility can been found in [5, 6, 7, 20, 22, 25, 26, 28, 29].

In previous CLM work on LT RNN [19], an asynchronous CLM iteration method was proposed based on the convergence proof in [4]. Because the whole network updates only one neuron status each time, this assumption makes iterations time consuming, especially for a large scale network. Therefore, a synchronous CLM iteration method would be helpful to solve this problem. In [30], we proposed a novel synchronous CLM iteration method, which has similar performance and storage allocation but faster convergence compared with the previous asynchronous CLM iteration method. In this paper, we extend CLM to Cellular Neural Networks (CNN) and significantly improve its efficiency.

The CNN model was first proposed by Chua and Yang in 1988 [2]. Since then, CNN have been widely studied both in theory and applications [8], [15]. Some stability analysis about the standard CNNs can been found in [2, 1, 21, 24]. Because popular nonlinear qualitative analysis methods always require the activation function to be differentiable, and the CNN neuron activation function is continuous but non-differentiable, the qualitative analysis of CNN turns out to be difficult.

In this paper, we first discuss a class of continuous-time recurrent CNN based on CLM (CLM-CNN). We prove that such networks can be complete stable. According to the qualitative analysis of our model, we first prove that there is no stable equilibrium in the interior of the network outputs' range. Through discussing the equilibrium existence condition on the ± 1 boundaries of the CNN neuron activation function, we present a necessary condition for producing feature binding phenomena. Furthermore, by using the sub-space method noted in [19], we give a sufficient and necessary condition.

We also discuss the properties of discrete-time type networks and propose a synchronous CLM iteration method based on previous work in [30]. This discrete-time CLM-CNN can have the same solution space for steady states as the continuous-time one under some conditions. Compared with another asynchronous CLM iteration method in [19], our method has similar performance and storage allocation but is less time consuming. Especially for a large scale network with many features and layers, it can greatly reduce the computing time.

The rest of this paper is organized as follows: The architecture of the proposed continuous-time CLM-CNN is described in Section 2. Preliminaries are given in Section 3. In Section 4, a theoretical analysis of the network is given, which includes: the complete stability of the proposed network, a necessary condition and a sufficient and necessary condition for feature binding. The discussion about the discrete-time CLM-CNN and iteration method can be found in Section 5. Simulations and illustrative examples are presented in Section 6. Conclusions are given in Section 7. The details of the theoretical analysis and iteration method about discrete-time CLM-CNN can be found in Appendix.

2. CLM for Continuous-time Cellular Neural Networks

The CLM consists of a set of l layers of feature-selective neurons, and each layer contains n neurons (see in Fig.1). There are two kinds of interactions in the model: the vertical WTA interaction and lateral interaction. Noted here, $x_{i\alpha}$ is the activity of a neuron at position i in layer α, and a column i denotes the set of the neuron activities $x_{i\alpha}$, $\alpha = 1, \cdots, l$ and $i = 1, \ldots, n$, that share a common position i in each layer. All neurons in a column i are equally driven by an external input. More details about the CLM architecture can be found in [19].

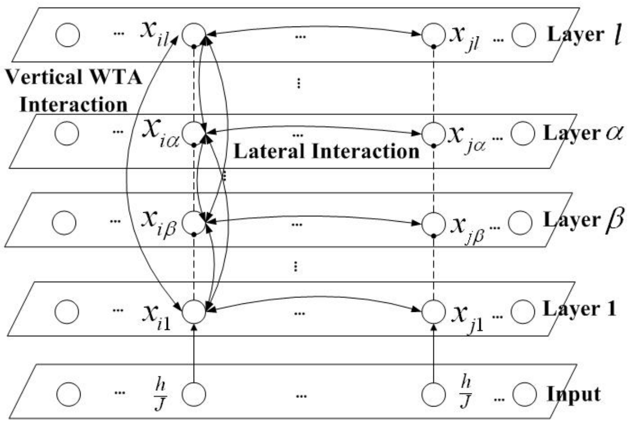

Figure 1: The CLM architecture.

In this paper, we first study a class of continuous-time recurrent Cellular Neural Networks based on CLM, which is described by the following equation:

$$\dot{x}_{i\alpha}(t) = -x_{i\alpha}(t) + \sigma\Big(\frac{h}{J} - \sum_{\beta=1}^{l} x_{i\beta}(t) + \frac{1}{J}\sum_{j=1}^{n} f_{ij}x_{j\alpha}(t) + x_{i\alpha}(t)\Big), \qquad (1)$$

for $t \geq 0$, $i = 1, \ldots, n$, and $\alpha = 1, \ldots, l$. The output function of CNNs is defined as follows: $\sigma(s) = \frac{|s+1|-|s-1|}{2}, s \in R$, which is continuous but non-differentiable. In our CLM, the vertical WTA interaction and lateral interaction are 1 and $\frac{f_{ij}}{J}$, respectively; $f_{ij} = f_{ji}$, for $i, j = 1, \ldots, n$; and the external input is $\frac{h}{J}$, which is used to make the networks stable. Here, the parameters h and J are used to adjust network performance.

For any vector $x \in \Re^n$, we denote

$$\sigma(x(t)) = (\sigma(x_1(t)), \sigma(x_2(t)), \ldots, \sigma(x_n(t)))^T \in \Re^n.$$

Then the equivalent vector form of network (1) can be defined as

$$\dot{x}(t) = -x(t) + \sigma(W_c x(t) + H + x(t)) \tag{2}$$

for $t \geq 0$, and $x(t)$ is a $n \times l$ vector

$$x(t) = [x_{11}(t), \ldots, x_{n1}(t), \ldots, x_{1l}(t), \ldots, x_{nl}(t)]^T,$$

$W_c = (w_{ij})_{nl \times nl}$ is a real symmetric matrix. Each element w_{ij} denotes the synaptic weights and represents the strength of the synaptic connection form neuron i to neuron j. $H \in \Re^{nl}$ denotes external input and $H_{i\alpha} = -\frac{h}{J}$ for all i, α,

$$W_c = \frac{1}{J} f \otimes I_l - \Pi_l \otimes I_n = \frac{1}{J} F - P, \tag{3}$$

where "$\otimes$" is Kronecker product [9], I_m is the $m \times m$ identity matrix (here $m = nl, n$, or l), Π_l is an $l \times l$ matrix of 1's, $f = (f_{ij})_{n \times n}$, $F = f \otimes I_l$, and $P = \Pi_l \otimes I_n$. Clearly, for a symmetric f, W_c is also symmetric. In this paper, $W_c + I_{nl}$ is supposed to be invertible.

Note that the CLM has two types of connections: the vertical interaction is 1, and the lateral interaction within layers is given by $\frac{f}{J}$, which represents the input features for whole system. In real applications, f may be the proximity interaction used for clustering [19], or the continuity interaction used for finding continuous curves [19]. In our examples for image segmentation, we store the pixel-related information in f, which is based on gray and position relationships. The purpose of CLM architecture is to enforce a dynamical assignment of the input features to the layers by using the contextual information stored in the lateral interactions f. This assignment can be considered as a process of feature binding. More detail about the model properties can be found in Section 4. Some examples for image segmentation are discussed in Section 6.

3. Preliminaries

In this section, we provide preliminaries which will be used in the following to establish our theories.

Definition 1. *A vector* $x^* = (x^*_{11}, \ldots, x^*_{nl}) \in R^{nl}$ *is called an equilibrium point (fixed point) of network (1), if it satisfies* $-x^*_{i\alpha} + \sigma(\frac{h}{J} - \sum\limits_{\beta=1}^{l} x^*_{i\beta} + \frac{1}{J}\sum\limits_{j=1}^{n} f_{ij}x^*_{j\alpha} + x^*_{i\alpha}) \equiv 0$ *for all* $i = 1, \ldots, n$*, and* $\alpha = 1, \ldots, l$*.*

Definition 2. *Let* $x(t, x_0)$ *be a trajectory of (1). An equilibrium point is said to be stable (in the sense of Lyapunov) if the following statement is true: if for any* $\varepsilon > 0$*, there exists a* $\delta > 0$ *such that* $\|x_0 - x^*\| \leq \delta$ *implies that* $\|x(t, x_0) - x^*\| \leq \varepsilon$ *for all* $t \geq 0$*.*

Definition 3. *The network (1) is called completely stable if each trajectory of (1) converges to an equilibrium point.*

Definition 4. *A neuron is said to be activated if its output doesn't equals* -1*. A neuron is said to be inactivated if it is not activated. A column of CLM is said to be activated if there exists at least one activated neuron in this column.*

We also denote that f has eigenvalues λ_i, $1 \leq i \leq n$ and

$$\eta = \max_{1\leq i\leq n}\{\sum_{j=1}^{n} |f_{ij}|\}. \tag{4}$$

4. Properties of CLM for CNN

4.1. Theory Proof

In this section, conditions guaranteeing complete stability of the network (1) are presented in Lemma 1. We also present a necessary condition in Theorem 1 to let our networks have CLM phenomena. Furthermore, a sufficient and necessary condition for CLM phenomena is given in Theorem 2.

In order to apply the CLM in the CNN, we first need to force the network to be completely stable, which means that every trajectory of the CNN converges to an equilibrium point.

Lemma 1. *Suppose matrix f is symmetric, then the network (1) is complete stable.*

Proof. By using the similar method in [24], the complete convergence can be proved.

This completes the proof.

After establishing the complete stability, we present below a necessary condition for letting our network have CLM phenomena.

Theorem 1. *Suppose matrix f is symmetric. If the lateral interaction is self-excitatory, $f_{ii} > 0$ for all i, and if there exist constants J and h such that $\begin{cases} J > 0 \\ -Jl - \eta < h \leq -J(l-2) - \eta \end{cases}$, then a stable equilibrium of CLM in network (1) has in each column i at most one activated neuron $x^*_{i\alpha}$ with $\sum\limits_{j=1}^{n} f_{ij}x^*_{j\alpha} \geq \sum\limits_{j=1}^{n} f_{ij}x^*_{j\beta}$.*

Proof. We first prove that the dynamics (1) has an energy function of the form

$$\begin{aligned} E(t) &= -\sum_{i=1}^{n}\sum_{\alpha=1}^{l}\frac{h}{J}x_{i\alpha}(t) + \frac{1}{2}\sum_{i=1}^{n}\sum_{\alpha=1}^{l}\sum_{\beta=1}^{l}x_{i\beta}(t)x_{i\alpha}(t) \\ &\quad -\frac{1}{2J}\sum_{i=1}^{n}\sum_{\alpha=1}^{l}\sum_{j=1}^{l}f_{ij}x_{j\alpha}(t)x_{i\alpha}(t) \qquad (5) \end{aligned}$$

under the constraint $|x_{i\alpha}(t)| \leq 1$ for all i, α.

Clearly, the network (1) is bounded. It follows from that

$$\begin{aligned} &\dot{E}(t) \\ = &\sum_{i=1}^{n}\sum_{\alpha=1}^{l}(-\frac{h}{J} + \sum_{\beta=1}^{l}x_{i\beta}(t) - \frac{1}{J}\sum_{j=1}^{n}f_{ij}x_{j\alpha}(t))\,\dot{x}_{i\alpha}(t) \\ = &-\sum_{i=1}^{n}\sum_{\alpha=1}^{l}D_{i\alpha}(t), \end{aligned}$$

where

$$\begin{cases} E_{i\alpha}(t) = \frac{h}{J} - \sum\limits_{\beta=1}^{l}x_{i\beta}(t) + \frac{1}{J}\sum\limits_{j=1}^{n}f_{ij}x_{j\alpha}(t) \\ D_{i\alpha}(t) = E_{i\alpha}(t)[-x_{i\alpha}(t) + \sigma(E_{i\alpha}\ (t) + x_{i\alpha}(t))] \end{cases}. \qquad (6)$$

Three cases will be considered. For Case 1: $|E_{i\alpha}(t)+x_{i\alpha}(t)| \le 1$, we have $\sigma(E_{i\alpha}(t)+x_{i\alpha}(t)) = E_{i\alpha}(t)+x_{i\alpha}(t)$, then $D_{i\alpha}(t) = E_{i\alpha}(t)[-x_{i\alpha}(t)+E_{i\alpha}(t)+x_{i\alpha}(t)] \ge 0$. For Case 2: $E_{i\alpha}(t)+x_{i\alpha}(t) \le -1$ and Case 3: $E_{i\alpha}(t)+x_{i\alpha}(t) \ge 1$, we can also easily have $D_{i\alpha}(t) \ge 0$. Therefore, we have $D_{i\alpha}(t) \ge 0$, and $\dot{E}(t) = -\sum\limits_{i=1}^{n}\sum\limits_{\alpha=1}^{l} D_{i\alpha}(t) \le 0$ and for all i, α, $t \ge 0$, which means that E(t) is monotonously decreasing. Furthermore, we have $\dot{E}(t) = 0 \Leftrightarrow D_{i\alpha}(t) = 0 \Leftrightarrow \dot{x}_{i\alpha}(t) = 0$.

Now assume the contrary: suppose an equilibrium x^* has at least two neurons which activities are bigger than -1 in a column i at two layer α and β, which means that $x^*_{i\alpha} > -1$, and $x^*_{i\beta} > -1$. Then two cases will be considered.

Case 1: $-1 < x^*_{i\alpha} < 1$ and $-1 < x^*_{i\beta} < 1$. Then at the position x^*, by (5), we have

$$\begin{aligned} E(t)|_{x^*} &= -\sum_{i=1}^{n}\sum_{\varphi=1}^{l}\frac{h}{J}x^*_{i\varphi} + \frac{1}{2}\sum_{i=1}^{n}\sum_{\varphi_1=1}^{l}\sum_{\varphi_2=1}^{l} x^*_{i\varphi_1}x^*_{i\varphi_2} \\ &\quad -\frac{1}{2J}\sum_{i=1}^{n}\sum_{\varphi=1}^{l}\sum_{j=1}^{l} f_{ij}x^*_{i\varphi}x^*_{j\varphi}. \end{aligned} \tag{7}$$

Now consider a small perturbation $x^{'} = x^* + \varepsilon$ near x^*, and

$$\varepsilon = (\varepsilon_{11}, \ldots, \varepsilon_{n1}, \ldots, \varepsilon_{1l} \ldots, \varepsilon_{nl}) \in R^{nl},$$

where $\varepsilon_{i\alpha} = -\varepsilon_{i\beta} \neq 0$ and $\varepsilon_{j\varphi} = 0$ for all other j, φ. By (7), we have

$$\begin{aligned} \Delta E &= E(t)|_{x'} - E(t)|_{x^*} \\ &= -\frac{h}{J}(\varepsilon_{i\alpha}+\varepsilon_{i\beta}) + \sum_{\varphi=1}^{l} x^*_{i\varphi}\varepsilon_{i\alpha} + \sum_{\varphi=1}^{l} x^*_{i\varphi}\varepsilon_{i\beta} \\ &\quad +\frac{1}{2}(\varepsilon_{i\alpha}+\varepsilon_{i\beta})^2 - \frac{1}{J}\sum_{j=1}^{n} f_{ij}x^*_{j\alpha}\varepsilon_{i\alpha} \\ &\quad -\frac{1}{J}\sum_{j=1}^{n} f_{ij}x^*_{j\beta}\varepsilon_{i\beta} - \frac{f_{ii}}{2J}((\varepsilon_{i\alpha})^2 + (\varepsilon_{i\beta})^2) \\ &= -\frac{f_{ii}}{J}(\varepsilon_{i\alpha})^2 < 0. \end{aligned}$$

So x^* can't be a stable equilibrium for the system.

Case 2: There are some neurons' final outputs which equal to 1, and one neuron's final output is no more than 1 and bigger than -1.

For simplicity, we just assume $x^*_{i\alpha} = 1$ and $-1 < x^*_{i\beta} \leq 1$, and other neurons $x^*_{i\varphi} = -1$, for $\varphi \neq \alpha, \beta$. Clearly, at $x^*_{i\alpha} = 1$, we have $\frac{h}{J} + \frac{1}{J}\sum_{j=1}^{n} f_{ij}x^*_{j\alpha} - \sum_{\varphi=1}^{l} x^*_{i\varphi} \geq 0$. Denote $E_{i\alpha} = \frac{h}{J} + \frac{1}{J}\sum_{j=1}^{n} f_{ij}x^*_{j\alpha} - \sum_{\varphi=1}^{l} x^*_{i\varphi}$, and by (4), we have

$$\begin{aligned} E_{i\alpha} &< \frac{h}{J} + \frac{1}{J}\sum_{j=1}^{n} |f_{ij}| \left|x^*_{j\alpha}\right| - [-(l-2) + x^*_{i\alpha} + x^*_{i\beta}] \\ &< \frac{h}{J} + \frac{\eta}{J} + (l-2). \end{aligned}$$

Since $h \leq -J(l-2)-\eta$, then $E_{i\alpha} < 0$. Therefore, x^* can't be an equilibrium.

When more than one neuron's final output equal to 1, under the condition $h \leq -J(l-2)-\eta$, it is also easy to prove that x^* can't be a stable equilibrium for the system.

We also notice that if there exists at most one neuron activity $-1 < x^*_{i\alpha} \leq 1$ and other $x^*_{i\beta} = -1$ for all $\beta \neq \alpha$, at this moment, $E_{i\alpha}$ has

$$E_{i\alpha} < \frac{h}{J} + \frac{\eta}{J} - [-(l-1) + x^*_{i\alpha}] < \frac{h}{J} + \frac{\eta}{J} + l.$$

If $h \leq -Jl-\eta$, there must be no neuron's output greater than -1. Therefore, h must satisfy that $h > -Jl - \eta$.

Furthermore, if there only exist a neuron $-1 < x^*_{i\alpha} \leq 1$ and $x^*_{i\beta} = -1$ for all $\beta \neq \alpha$, we have $\sum_{j=1}^{n} f_{ij}x^*_{j\alpha} \geq \sum_{j=1}^{n} f_{ij}x^*_{j\beta}$. by comparison between $E_{i\alpha}$ and $E_{i\beta}$.

This completes the proof.

From Theorem 1, we can find that the network (1) has two operation modes in a column: the first one with an activated neuron in one column, and the second one with no activated neuron in one column. Since the condition in Theorem 1 is just a necessary condition, the outputs of network (1) may be the second mode for all columns, which is not what we want in real applications. In the next theorem, new conditions for J and h are presented to improve the network performance.

Theorem 2. *Suppose the network (1) satisfies Theorem 1. If there exists constants J and h such that* $\begin{cases} J > \max\{\frac{1}{l}\max\limits_{1\le i\le n}\{\lambda_i\}, \eta/2\} \\ -Jl < h \le -J(l-2)-\eta \\ Jl \gg \lambda_i \end{cases}$, *then a stable equilibrium has at least one activated column.*

Proof. In order to prove that the network (1) has at least one activated column, we just need to prove $x_0 = [-1,-1,\dots,-1]^T \in \Re^{nl}$ can't be a stable equilibrium.

We can consider the CLM dynamical system (1) as

$$\dot{x_{i\alpha}}(t) = \begin{cases} 0 & for\ x_{i\alpha} = 1, E_{i\alpha}(t) \geqslant 0 \\ 0 & for\ x_{i\alpha} = -1, E_{i\alpha}(t) \leqslant 0 \\ E_{i\alpha}(t) & for\ x_{i\alpha} \ne \pm 1, |E_{i\alpha}(t) + x_{i\alpha}| \leqslant 1 \\ 1 - x_{i\alpha} & for\ x_{i\alpha} \ne \pm 1, E_{i\alpha}(t) + x_{i\alpha} > 1 \\ -1 - x_{i\alpha} & for\ x_{i\alpha} \ne \pm 1, E_{i\alpha}(t) + x_{i\alpha} < -1 \end{cases}$$

for all i, α, and $E_{i\alpha}(t)$ is defined in (6).

For the network (1), on the boundary, we have

$$\begin{cases} if\ x_{i\alpha} = -1, and\ E_{i\alpha}(t) > 0 \Rightarrow \dot{x_{i\alpha}}(t) > 0 \\ if\ x_{i\alpha} = -1, and\ E_{i\alpha}(t) < 0 \Rightarrow \dot{x_{i\alpha}}(t) = 0 \\ if\ x_{i\alpha} = \pm 1, and\ E_{i\alpha}(t) = 0 \Rightarrow \dot{x_{i\alpha}}(t) = 0 \\ if\ x_{i\alpha} = 1, and\ E_{i\alpha}(t) < 0 \Rightarrow \dot{x_{i\alpha}}(t) < 0 \\ if\ x_{i\alpha} = 1, and\ E_{i\alpha}(t) > 0 \Rightarrow \dot{x_{i\alpha}}(t) = 0 \end{cases}. \tag{8}$$

And in the interior of $\Phi = \{x_{i\alpha}||x_{i\alpha}| \le 1, i = 1,\dots,n, \alpha = 1,\dots,l\}$, it always holds that

$$\begin{cases} E_{i\alpha}(t) > 0 \Rightarrow \dot{x_{i\alpha}}(t) > 0 \\ E_{i\alpha}(t) = 0 \Rightarrow \dot{x_{i\alpha}}(t) = 0 \\ E_{i\alpha}(t) < 0 \Rightarrow \dot{x_{i\alpha}}(t) < 0 \end{cases}.$$

Thus, in the region Φ, we can discuss the dynamics of network (1) by investigating such linear dynamics

$$\dot{x_{i\alpha}}(t) = E_{i\alpha}(t) = \frac{h}{J} - \sum_{\beta=1}^{l} x_{i\beta}(t) + \frac{1}{J}\sum_{j=1}^{n} f_{ij}x_{j\alpha}(t), \tag{9}$$

except for some boundary conditions which are noted in (8). Furthermore, we can apply the method of eigensubspace analysis in such linear domain, where the constraints are inactive [19].

The equivalent vector form of network (9) is

$$\dot{x}(t) = W_c x(t) + H. \tag{10}$$

From (3), an orthonormal eigenvector basis $\{V^{i\beta}, \Lambda_{i\beta}\}$ for W_c can be obtained from the orthonormal eigenvector bases $\{b_i, \lambda_i\}$ and $\{q^{\beta}, \mu_{\beta}\}$ for f and Π_l respectively [19]:

$$\begin{cases} V^{i1} = \frac{1}{\sqrt{l}}\left(b_i^T, ..., b_i^T\right)^T, \Lambda_{i1} = \frac{1}{J}\left(\lambda_i - Jl\right) \\ V^{i\beta\neq1} = \left(\sqrt{\sum\limits_{\alpha=1}^{l}\left(q_{\alpha}^{\beta\neq1}\right)^2}\right)^{-\frac{1}{2}} \cdot \left(q_1^{\beta\neq1}b_i^T, ..., q_l^{\beta\neq1}b_i^T\right)^T, \Lambda_{i\beta\neq1} = \frac{1}{J}\lambda_i \end{cases}. \tag{11}$$

If $Jl \gg \lambda_i$ and $-Jl < h \leq -J(l-1)$, there is an approximate equilibrium x^F of the linear system (9) in the interior of Φ [19]: $x^F \approx [\frac{h}{Jl}, \frac{h}{Jl}, \ldots, \frac{h}{Jl}]^T \in \Re^{nl}$.

If x_0 is a stable equilibrium of network (1), there exist a $\eta > 0$ and $x^{'} \neq x_0$, which satisfies $\{x^{'} | -1 \leq x_{i\alpha}^{'} < \frac{h}{Jl}\}$ and $\|x^{'} - x_0\| < \eta$, such that

$$\dot{x}_{i\alpha}(t)|_{x'} \leq 0 \Rightarrow E_{i\alpha}(t)|_{x'} \leq 0$$

for all i, α.

Now at the same point $x^{'}$ in the linear system (9), it must hold $\dot{x}_{i\alpha}(t)|_{x'} \leq 0$ for all i, α, which means $\exists \bigtriangleup t > 0$ and $\bigtriangleup t \to 0$, such that

$$\lim_{\bigtriangleup t \to 0} \frac{x_{i\alpha}(t + \bigtriangleup t) - x_{i\alpha}(t)}{\bigtriangleup t}|_{x(t)=x'} \leq 0.$$

Since $x^{'} < x^F$, at $x^{'}$, we have $x(t + \bigtriangleup t) - x^F \leq x(t) - x^F < 0$, which means

$$\begin{aligned} &(x(t + \bigtriangleup t) - x^F)^T (x(t + \bigtriangleup t) - x^F)|_{x(t)=x'} \\ &\quad \geq (x(t) - x^F)^T (x(t) - x^F)|_{x(t)=x'}. \end{aligned} \tag{12}$$

By (10), we have $\dot{x}(t) = W_c(x(t) - x^F)$ and

$$x(t) = \sum_{i=1}^{n}\sum_{\alpha=1}^{l} \xi_{i\alpha}(t) V^{i\alpha}, \tag{13}$$

where $\xi_{i\alpha}(t)$ is the projection of $x(t)$ on $V^{i\alpha}$.

According to the subspace method noted in [19], we can divide the eigenmodes of the linear system into two subspaces: DC-Subspace and AC-Subspace based on V^{i1}, $V^{i\beta\neq 1}$ respectively. The projection operators on both subspaces can be expressed by (11)

$$\left\{\begin{array}{l} P^{DC} = \sum\limits_{i=1}^{n} V_{i1}V_{i1}^{T} = \frac{1}{l}\Pi_l \otimes I_n \\ P^{AC} = \sum\limits_{\beta=2}^{l}\sum\limits_{i=1}^{n} V_{i\beta}V_{i\beta}^{T} = I_{nl} - \frac{1}{l}\Pi_l \otimes I_n \end{array}\right. .$$

By simple calculation, we have

$$\left\{\begin{array}{l} P^{DC}(x^{'} - x^{F}) = x^{'} - x^{F} \\ P^{AC}(x^{'} - x^{F}) = 0 \end{array}\right. . \tag{14}$$

Therefore, the vector $x^{'} - x^{F}$ is in the DC-Subspace and has no components in AC-Subspace, which means that $\xi_{i\beta}(t)|_{x(t)=x'} = 0$, for $i = 1, \cdots, n$, and $\beta = 2, \cdots, n$. Then by (13), we have

$$\begin{aligned} (x(t) - x^{F})|_{x(t)=x'} &= \sum_{i=1}^{n}\sum_{\alpha=1}^{l} \xi_{i\alpha}(t)V^{i\alpha} \\ &= \sum_{i=1}^{n} \xi_{i1}(t)V^{i1}, \end{aligned} \tag{15}$$

$$\begin{aligned} (x(t+\triangle t) - x^{F})|_{x(t)=x'} &= \sum_{i=1}^{n}\sum_{\alpha=1}^{l} e^{\Lambda_{i\alpha}\triangle t}\xi_{i\alpha}(t)V^{i\alpha} \\ &= \sum_{i=1}^{n} e^{\Lambda_{i1}\triangle t}\xi_{i1}(t)V^{i1}. \end{aligned} \tag{16}$$

Since in DC-Subspace

$$J > \max\{\frac{1}{l}\max_{1\leq i\leq n}\{\lambda_i\}, \eta/2\} \Rightarrow \Lambda_{i1} = \frac{1}{J}(\lambda_i - Jl) < 0, \tag{17}$$

by (15), (16) and (17), we have

$$\begin{aligned} &(x(t+\triangle t) - x^{F})^{T}(x(t+\triangle t) - x^{F})|_{x(t)=x'} \\ &\qquad < (x(t) - x^{F})^{T}(x(t) - x^{F})|_{x(t)=x'}. \end{aligned} \tag{18}$$

From (12), (18), we find the contradiction that x_0 is a stable equilibrium of network (1).

This completes the proof.

The condition $Jl \gg \lambda_i$ is important but seems too hard to evaluate. If J is not large enough, (18) can not be satisfied, then x^F will not locate in $(-1, 1)$ and no activated column can be found. Here, we approximately solve such problem by increasing J step by step to make sure that x^F is located in $(-1, 1)$, which can guarantee that at least one activated column exists in the network. This technical approach is used in our proposed synchronous CLM iteration method in Section 5.

5. CLM for Discrete-time Cellular Neural Networks

5.1. CLM for Discrete-time Cellular Neural Networks

There are two main approaches to study the relationship between the discrete-time system and the continuous-time one. The first one is to transform the system into a corresponding deterministic continuous-time system, which is based on a fundamental theorem of stochastic approximation theory [12]. To use this fundamental theorem of stochastic approximation, some crucial conditions must be satisfied, just like the roundoff limitation and tracking requirements. Additional important condition is that the learning rate of algorithms must approach zero. All of these restrictive conditions may not be satisfied in many theoretical models and practical applications. The second one is to transform the system into a deterministic discrete-time system [27]. It focuses on the stability analysis of the system and the qualitative analysis on equilibrium, and it does not require the learning rate to approach zero.

In this article, we choose the last approach to build our discrete-time model described by the following equations:

$$x_{i\alpha}(k+1) = \sigma(x_{i\alpha}(k) + \frac{h}{JC} - \frac{1}{C}\sum_{\beta=1}^{l} x_{i\beta}(k) + \frac{1}{JC}\sum_{j=1}^{n} f_{ij}x_{j\alpha}(k)) \tag{19}$$

for $k \geq 0$, $i = 1, \ldots, n$, and $a = 1, \ldots, l$, which dynamic properties are testified in Appendix 8.1.

Comparing network (1) with network (19), we added a parameter C to make the discrete-time system completely stable. Furthermore, if both networks satisfy Theorem 2 and Theorem 5, their equilibriums are in the same solution space by inspecting positive or negative signs of $\frac{h}{J} - \sum_{\beta=1}^{l} x_{i\beta}^{*} + \frac{1}{J}\sum_{j=1}^{n} f_{ij}x_{j\alpha}^{*}$. Therefore, we can use the discrete-time system instead of the continuous-time one in real applications.

5.2. Comparison between CNN-CLM-Dynamics and CLM-LT-Dynamics [19] implementation methods

In [19], an asynchronous CLM-LT-Dynamics implementation method was proposed based on the convergence theory in [4] (CLM-LT method). The brief introduction about this method can be found in Appendix 8.3.

The CLM-CNN-Dynamics implementation in discrete-time type can be found in Appendix 8.2 (CLM-CNN method), which is done in parallel. Since the CLM-CNN method adopts the similar idea of implementing CLM in synchronous update compared with our previous synchronous LT method in [30], it is not difficult to testify that both methods have similar performance and efficiency. Therefore, we only compare the CLM-CNN method with the CLM-LT method.

Although these methods are used in two different neural networks, CLM-CNN Dynamics is similar to CLM-LT Dynamics by virtue of linear system analysis [19]. Here, we compare two methods in three aspects: memory requirement, performance and iteration speed.

First, we compare the memory requirement. Because the weight W_d always needs an enormous size of memory, it is impossible to directly use network (21) for a large-scale network on a PC. For example, for 10000 features and 10 layers, and assuming 2 bytes to store one data point, then the storage for W_d is $(10000 \times 10 \times 2)^2/(1024)^3 \simeq 37.25GB$. Our CLM-CNN method can greatly decrease the storage requirements. For previous example, we don't need to store W_d but f and Π_l (Here $l = 10$), which respectively occupy $(10000 * 2)^2/(1024)^2 \simeq 381.47M$ and $(10 * 2)^2/1024 \simeq 0.39KB$, and related memory occupation is only about 1% of previous storage request(Here $1GB = 1024MB = 1024^2KB$). Both the CLM-LT and the CLM-CNN methods have similar memory requirements.

In order to estimate the performance of two methods, we use a lateral contribution energy function to provide an indirect measure, which is described as

$$E_{CLM} = -\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\alpha=1}^{l} f_{ij} z_{i\alpha}^{*} z_{j\alpha}^{*}.$$

For CLM-CNN methods, $z^*_{i\alpha}$ is defined by $z^*_{i\alpha} = \begin{cases} 1, & if\ x^*_{i\alpha} \neq -1 \\ 0, & if\ x^*_{i\alpha} = -1 \end{cases}$, where $x^*_{i\alpha}$ is the stable fixed point of network (19). For CLM-LT methods, $z^*_{i\alpha}$ is defined by $z^*_{i\alpha} = \begin{cases} 1, & if\ x^*_{i\alpha} \neq 0 \\ 0, & if\ x^*_{i\alpha} = 0 \end{cases}$, where $x^*_{i\alpha}$ is the final LT network output. In brief, for fixed n and l, the lower energy value, the better the performance.

Here we want to point out that J is an importance parameter for both CLM methods. Through simulations, we find that if J is too small, the iterations will be very fast but the performance will be very low; while for large J, the iteration speed will decrease but the performance will increase. Such relationship can also be observed from their iteration expressions and be obtained from the dynamic analysis of CLM [19]. However, the lateral connections provide the contextual information. If J is too big, the lateral connections may be negligible. Therefore, J should be chosen carefully.

In CLM-LT method, J is fixed as a constant. In CLM-CNN method, we adopt a different strategy to balance the iteration speed and the CLM performance. Here, we first set J to be a small value, which guarantees fast iteration speed. When the iteration variety between $z_{i\alpha}(k+1)$ and $z_{i\alpha}(k)$ decreases to some desirable value, we increase J step by step to improve performance. When J seems to be too big, we stop the whole iterations.

With above parallel execution and variable adjustment for parameter J, our CLM-CNN method greatly improves the CLM efficiency. From the simulation results, we find that CLM-CNN method can run faster than CLM-LT one and have similar memory requirement as CLM-LT one. On the other hand, the performance differences between both methods are not distinct.

Through simulations, we also find that if we fix J as the same value for both methods, the CLM-LT method always performs better than the CLM-CNN one. It seems that the CLM-LT method is easier than the CLM-CNN one to escape from some local minima. But even with the same J, the CLM-CNN method is faster than the CLM-LT method.

An self-inhibitory annealing schedule was used to increase the system performance for CLM-LT method. Through simulation, we find that this annealing schedule is also applicable for our CLM-CNN method. The related simulation results are given in Section 6.

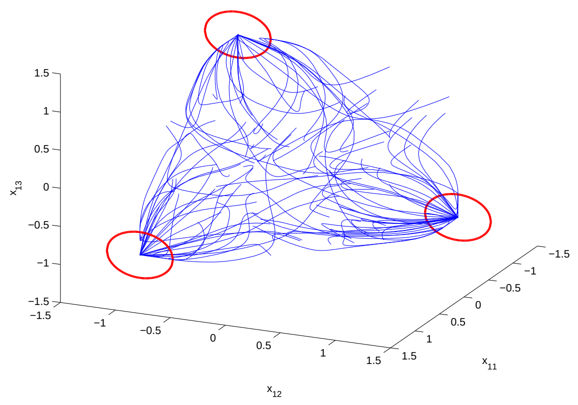

Figure 2: Complete stability of three components of network (20) in one column: x_{11}, x_{12}, x_{13}. Three equilibriums marked with three circles, respectively.

6. Simulation

In this section, we provide examples of simulation results to illustrate and verify the theory developed. Almost all programs, which were coded in MATLAB 2008a, were run in a PC with 1 Intel i7 920@2.67GHz CPU, 6 GB RAM, and Windows Vista Ultimate Service Pack 1 64-bit operation system. We also provide the original code of Example 5 [3].

Example 1. *Consider a CLM neural network with 3 layers with 4 neurons in each layer*

$$\dot{x}(t) = -x(t) + \sigma(W_c x(t) + H + x(t)), \tag{20}$$

where $W_c = \frac{1}{J}[f \otimes I_3 - \Pi_3 \otimes I_4] + I_{12}$*,* $H = \frac{J}{C}[1, 1, \ldots, 1]^T$*,* $J = 1.9667$*,* $h = -1.9667$*, and*

$$f = \begin{bmatrix} 0.5493 & 0.1435 & 0.6019 & -0.5720 \\ 0.1435 & 0.0804 & -0.1325 & 0.6247 \\ 0.6019 & -0.1325 & 0.3964 & 0.0652 \\ -0.5720 & 0.6247 & 0.0652 & 1.4676 \end{bmatrix}.$$

Fig.2 shows the complete convergence of the four-column network (20) for 100 trajectories originating from randomly selected initial points in $[-1, 1]$*.*

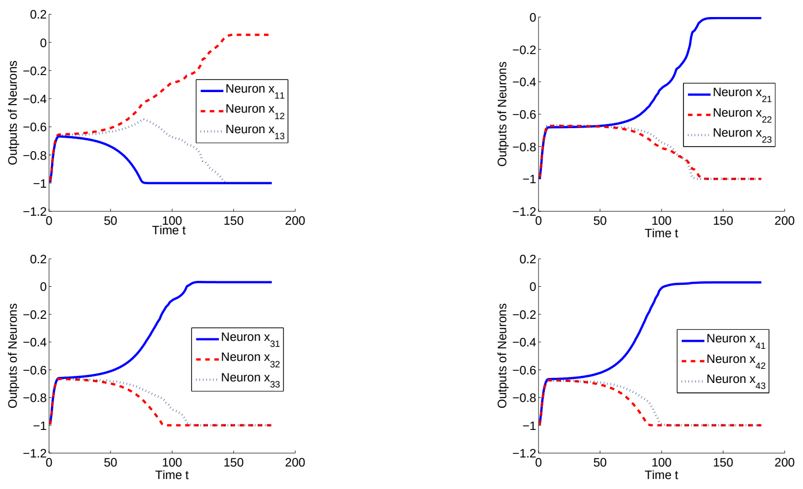

Figure 3: Trajectory of network (20) in four columns and a column WTA behavior can be observed in four graphs.

Example 2. *Consider the network (20) in Example 1, and just set* $J = 0.2$, $h = -2.0667$. *Clearly, network (20) satisfies Theorem 1. And there is only one stable equilibrium:* $x^* = [-1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ -1]^T$, *which means that there is no activated column in this network.*

Example 3. *Consider the network (20) in Example 1, and just set* $J = 19.6670$, $h = -39.3340$. *Clearly, network (20) satisfies Theorem 2. Fig.3 shows the group WTA performance when the initial value* $-1 < x(0) \ll \frac{h}{JL}$, *only one neuron in a column is activated in the stable state:* $x^* = [-1\ -0.0066\ 0.0314\ 0.0297\ 0.0539\ -1\ -1\ -1\ -1\ -1\ -1\ -1]^T$. *Note that there are two layers with* 2 *group features and three group winners can be found: (1)* x_{21}, x_{31}, x_{41} *in the layer 1; (2)* x_{12} *in the layer 2.*

Example 4. *Here, we compare our CLM-CNN iteration method with another CLM-LT iteration method noted in [19].*

We select f *randomly, where* $f_{ij} \in [-1000, 1000]$ *and* $f_{ii} > 0$ *for* $1 \le i, j \le n$. *The feature number* $n = \{50\ 100\ 300\ 600\ 1000\}$, *the layer number* $l = \{5\ 10\ 15\}$. *For each selected* f, *the test number is* 10 *for each layer number. Therefore, these are totally* $5 \times 3 \times 10 \times 4 = 600$ *tests. For both methods with annealing,* T *is set to the biggest eigenvalue of* f, η_T *is* 0.99.

Table 1 *and Table* 2 *are results for* $n = 50$, $n = 1000$, *respectively. The best results, which include running time and performance* (E_{CLM}), *are shown* *in boldface. From the tables, it is hard to tell which method performs better. This conclusion can also be inferred from their similar dynamics. On the other hand, the performance always depends on the initial value, f and some parameters, thus the choice between two methods in real applications may be different. Fig.4 shows the computing time for two methods. With more features and layers, the CLM-LT method needs longer computing time. With the same feature number and layer number, our method runs faster than the CLM-LT method. For example, when $n = 1000$ and $l = 15$, our method is about* 379 *times (without annealing) or* 416 *times (with annealing) faster than previous CLM-LT method. Therefore, it is more suitable to use our method when we deal with a large scale network with lots of features and layers.*

Since f is totally out-of-order and has little group information, it offers an objective comparison environment to evaluate two methods. In the following example, the test data is the Lena image, which can be seen as organizational data embodying lots of group information.

Table 1: Comparisons of CNN-CLM Methods and LT-CLM Method (n=50)

CLM	Number	Time (Seconds)			E_{CLM} ($\times 10^5$)		
Method	of Layers	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*
CLM-LT	5	0.8430	2.7930	1.4198	-1.5111	-1.3071	-1.3916
With	10	1.5760	5.6000	3.2730	-1.4308	-1.2573	-1.3402
Annealing	15	3.2760	6.4900	4.2886	-1.4527	-1.2494	-1.3736
CLM-LT	5	0.8580	2.8540	1.4818	-1.4454	-1.2592	-1.3761
Without	10	1.6070	4.3830	3.0014	-1.5312	-1.2435	-1.3858
Annealing	15	2.0440	8.2520	4.3694	-1.4659	-1.1758	-1.3781
CLM-CNN	5	**0.0310**	0.1870	0.0686	-1.5663	**-1.4029**	-1.4893
With	10	**0.0780**	0.1400	0.1075	**-1.5686**	**-1.4036**	**-1.4932**
Annealing	15	0.1250	0.3280	0.1825	-1.5377	**-1.4639**	**-1.5019**
CLM-CNN	5	**0.0310**	**0.1090**	**0.0530**	**-1.5678**	-1.3987	**-1.4967**
Without	10	**0.0780**	**0.1250**	**0.0905**	-1.5377	-1.3630	-1.4637
Annealing	15	**0.1240**	**0.2340**	**0.1701**	**-1.5546**	-1.3906	-1.5072

Example 5. *Image segmentation is a important task in image processing. It aims to partition the image into meaningful sub-regions or grouping objects with the same attributes. We compare two methods for a* 64×64 *Lena*

Table 2: Comparisons of CNN-CLM Methods and LT-CLM Method (n=1000)

CLM Method	Number of Layers	Time (Seconds)			E_{CLM} ($\times 10^7$)		
		Min	*Max*	*Avg*	*Min*	*Max*	*Avg*
CLM-LT	5	1.0005×10^3	4.5643×10^3	2.4231×10^3	-1.1373	-1.0885	-1.1087
With	10	5.0236×10^3	1.5988×10^4	7.9164×10^3	-1.0966	-1.0184	-1.0616
Annealing	15	7.4818×10^3	2.2760×10^4	1.5436×10^4	-1.1210	**-1.0223**	**-1.0701**
CLM-LT	5	1.5355×10^3	3.3628×10^3	2.4224×10^3	-1.1485	-1.0850	-1.1117
Without	10	4.9369×10^3	1.6160×10^4	8.8480×10^3	-1.1211	-1.0041	-1.0644
Annealing	15	7.5082×10^3	2.2895×10^4	1.4063×10^4	-1.1063	-1.0220	-1.0652
CLM-CNN	5	25.3500	60.1070	39.4181	**-1.2072**	**-1.1574**	-1.1714
With	10	**28.5170**	**44.8040**	**35.6650**	-1.1337	**-1.0552**	-1.0971
Annealing	15	**29.2020**	48.7040	**37.0221**	**-1.1328**	-1.0018	-1.0624
CLM-CNN	5	**25.0380**	**47.7990**	**36.1390**	-1.1967	-1.1517	**-1.1717**
Without	10	28.7820	47.8140	37.6070	**-1.1489**	-1.0524	**-1.1012**
Annealing	15	31.6530	**42.4940**	37.0469	-1.0821	-1.0136	-1.0593

*image(*4096 *features) in* 5 *layers. How to construct the image segmentation feature can be found in Appendix 9.4. Fig.5 shows test results.*

Compared with the previous asynchronous CLM iteration method, our synchronous CLM-CNN iteration method can easily be implemented in GPU (graphics processing unit) parallel computing. The additional simulations were coded in MATLAB 2010a, and were run in another PC with 1 Intel i7 9600k CPU, 8 GB RAM, 1 NVIDIA® GeForce® GTX 560 Ti with 1 GB GDDR DRAM, and Windows 7 Ultimate Service Pack 1 64-bit operation system. The simulation results can be found in Table 3. Under the same condition, the average time cost is 215.2966(no more than 4 minutes).By adjusting some parameters, we can decrease time cost to about 2 minutes while keeping enough accuracy further more. More discussion about GPU computing can be found in [10].

Example 6. *Here, the CLM-CNN method is applied to a* 128×128 *Lena picture(*$128^2 = 16384$ *features), the layer number* $l = \{3\ 6\}$*(see in Fig.6). The time and the energy for 3 layers are* 8.7820×10^3 *seconds,* 3.6972×10^6*, respectively. For 6 layers, the time and the energy are* 2.1317×10^3 *seconds,*

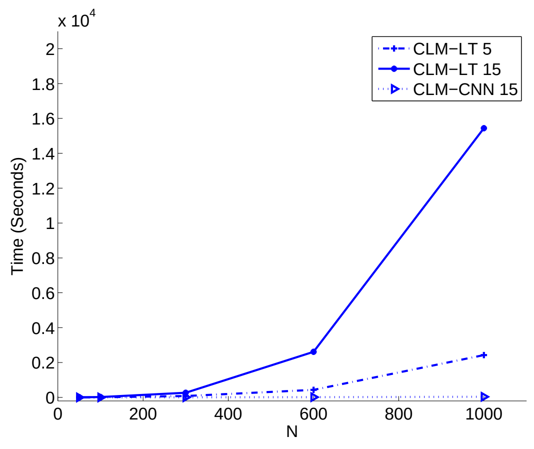

Figure 4: Computing time for the proposed CLM-CNN method with 15 layers, the CLM-LT method with 5, 15 layers.

Table 3: Comparisons of CNN-CLM Method with different parameters by virtue of GPU Computing (n=64 $\times$ 64, layer=5,test time= 50), here W=With Annealing, Wo=Without Annealing.

CNN-CLM method	Time (Seconds)			E_{CLM} $(\times 10^5)$		
	Min	Max	Avg	Min	Max	Avg
default parameters(W)	64.5060	947.877	215.2966	-4.5360	-4.4963	**-4.5159**
$\tau_{max} = 1.04$(Wo)	56.1920	347.1630	129.2018	-4.5377	-4.1006	-4.5077
$J = 1.04, \tau_{max} = 1.04$, $x^1_{i\alpha}(k+1) > 0.0001$(Wo)	**50.2170**	**295.2150**	**124.6740**	**-4.5387**	**-4.0951**	-4.5002

-7.0325×10^7, *respectively. Clearly, the image segmentation quality can be improved with more layers, which decreases the lateral contribution energy as well.*

CLM has its own advantage on feature grouping, which were shown in feature binding and sensory segmentation successfully [19]. Although other methods exist for image segmentation, the simulations presented demonstrate interesting feature grouping ability of CLM. Compared with previous application in sensory segmentation (about $2000 \sim 3500$ *features)[19], our simulations on image segmentation have more features (about* $4000 \sim 16000$ *features) and are more complicated.*

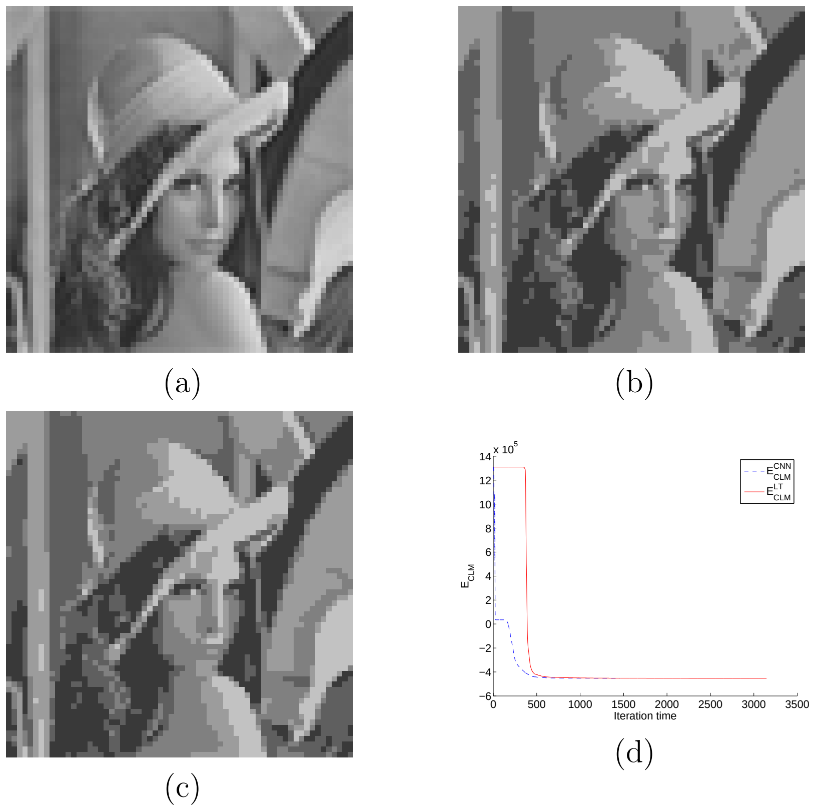

Figure 5: Image segmentation for 64 $\times$ 64 Lena image. (a) Original image; (b) Image segmented with 5 layers using CLM-LT method with annealing, uses 3.5976×10^4 seconds (about 10 hours), and E_{CLM}^{LT} is -4.5324×10^5; (c) Image segmented with 5 layers using CLM-CNN method with annealing, uses 988.129 seconds (about 16 minutes), and E_{CLM}^{CNN} is -4.5298×10^5; (d) Lateral contribution energies of two methods. Here, we calculate E_{CLM}^{CNN} once per l iterative times, and calculate E_{CLM}^{LT} once per $n \times l$ iterative times.

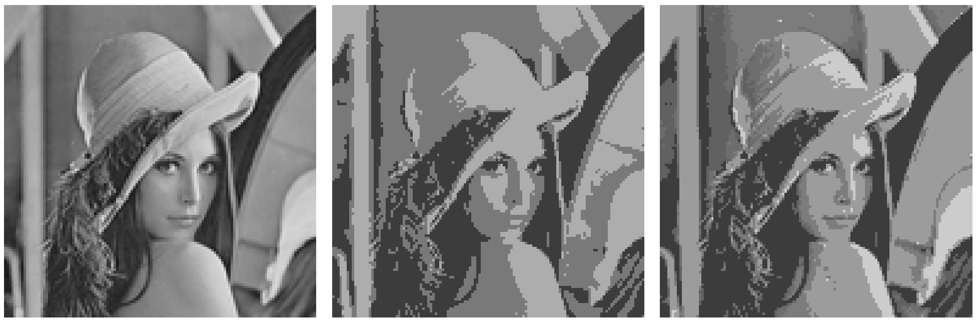
(a) (b) (c)

Figure 6: Image segmentation worked with 128×128 Lena image using CLM-CNN method. (a) Original image; (b) Image segmented with 3 layers ; (c) Image segmented with 6 layers.

7. Conclusions

In this paper, we investigate the Competitive Layer Model for a class of cellular recurrent neural networks from continuous-time type to discrete-time type. We establish complete stability of both networks. In order to define the feature binding phenomena for the discussed class of networks, we present necessary conditions and sufficient and necessary conditions. In addition, we outline some dynamic properties analysis of our model. We also give a novel synchronous CLM-CNN iteration method, which has similar performance and storage allocation but is less time consuming and is more suitable for a large scale network compared with the previous asynchronous CLM iteration method. Simulations are carried out to validate the performance of our theoretical findings and provide the comparison.

It should be noted that this study has examined only some approximate conditions, because few nonlinear qualitative analysis methods can deal with the dynamics with non-differentiable condition well. Notwithstanding the model's limitation, this study does suggest a method which could be further developed to solve some related problems of other class of recurrent neural networks.

Considering current technical trends, the multi-core processor technology significantly increases parallel processing capability as compared with single-core processing. Therefore, with the development of software and hardware of parallel computation, the efficiency and speed of our CLM-CNN method can still be further improved in future. Eventually, dealing with a large scale network with millions of features may be simulated perfectly in finite time, thus makes our approach of high practical value.

Since the essence of CLM can be looked at as an optimization method to search for a better feature-grouping solution among those possible solutions, the method described in this paper may well be extended to other applications dealing with complex optimization problems.

Acknowledgment The authors wish to thank Yonglin Zeng, Dr. Ping Li, Artur Abdullin, Dr. Dongqin Chen, Dr. Lijun Zhang, and Minqing Zhang for their useful discussions and comments. We also want to thank those anonymous reviewers for their suggestions.

8. Appendix

8.1. Proof of Theorems for Network (19)

The equivalent vector form of (19) can be defined as

$$x(k+1) = \sigma(W_d x(k) + H) \tag{21}$$

for $k \geq 0$, and $x(k) = [x_{11}(k), \ldots, x_{nl}(k)]^T$.

Furthermore, we define

$$W_d = G + I_{nl}, \tag{22}$$

$$G = \frac{1}{JC} f \otimes I_l - \frac{1}{C}\Pi_l \otimes I_n. \tag{23}$$

Note that for W_d being invertible, the network (19) is dynamically equivalent to

$$\begin{aligned} y_{i\alpha}(k+1) \quad &= \quad \sigma(y_{i\alpha}(k)) + \frac{h}{JC} - \frac{1}{C}\sum_{\beta=1}^{l}\sigma(y_{i\beta}(k)) \\ &\quad + \frac{1}{JC}\sum_{j=1}^{n} f_{ij}\sigma(y_{j\alpha}(k)). \end{aligned} \tag{24}$$

Clearly, we have

$$x_{i\alpha}(k+1) = \sigma(y_{i\alpha}(k+1)). \tag{25}$$

The equivalent vector form of (24) can be defined as

$$y(k+1) = W_d \sigma(y(k)) + H \tag{26}$$

for $k \geq 0$, and $y(k) = [y_{11}(k), \ldots, y_{nl}(k)]^T$.

8.1.1. Theory Proof

Lemma 2. *If there exists a diagonal positive definite matrix D such that $D(I_{nl} + W_d)$ is a symmetric positive definite matrix, then the network (19) is completely convergent.*

Proof: Define a set

$$S = \left\{ \begin{bmatrix} e_1 & 0 & \cdots & 0 \\ 0 & e_2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & e_{nl} \end{bmatrix} e_i = 0 \text{ } or \text{ } 1, \ 1 \leq i \leq nl \right\}.$$

Clearly, each elements of S is a $nl \times nl$ matrix and the set S has total 2^{nl} elements, Using this notation, a representation of each trajectory of the network (26) can be given.

At each step k, there exists matrices $E(i), E^1(i), E^2(i) \in S$ and $I = [1, 1, \cdots, 1]^T \in R^{nl\times 1}$ such that

$$\sigma(y(k)) = E(k)y(k) + E^1(k)I - E^2(k)I,$$

for all $k \geq 0$.

Thus, (26) can be rewritten as

$$y(k+1) = W_d \left[E(k)y(k) + E^1(k)I - E^2(k)I\right] + H.$$

By iterating, the trajectory of (26) starting from $y(0)$ can be represented as

$$\begin{aligned} y(k+1) \;\; = \;\; & \left[\prod_{j=0}^{k-1} E(k-j)\right] y(0) + W_d \left[E^1(k)I - E^2(k)I\right] \\ & + \sum_{i=0}^{k-1} \left[\prod_{j=0}^{k-1-i} W_d E(k-j)\right] W_d \left[E^1(i)I - E^2(i)I\right] \\ & + \sum_{i=0}^{k-1} \left[\prod_{j=0}^{i} W_d E(k-j)\right] H + H \end{aligned}$$

for all $k \geq 0$.

Since the set S has 2^{nl} elements, from the trajectory representation, each component of $y(k+1)$ must be a polynomial with order at most k.

By using the similar method in [25], [26], the complete convergence can be proved.

The proof is completed.

Theorem 3. *Suppose matrix f is symmetric. If the lateral interaction is self-excitatory, $f_{ii} > 0$ for all i, and if there exist constants J and C such that* $\left\{\begin{array}{l} J > \frac{1}{l}\max\limits_{1\le i\le n}\{|\lambda_i|\} \\ C > l \end{array}\right.$ *, then the network (19) is completely convergent.*

Proof. In order to prove the network is completely convergent, we need to prove the matrix $I_{nl} + W_d$ is a positive definite matrix. From (22),(23), an orthonormal eigenvector basis $\{V_{i\beta}, \Lambda^d_{i\beta}\}$ for W_d can be obtained from the orthonormal eigenvector bases $\{b_i, \lambda_i\}$ and $\{q^\beta, \mu_\beta\}$ for f and Π_l respectively [19]:

$$\left\{\begin{array}{l} V_d^{i1} = \frac{1}{\sqrt{l}}\left(b_i^T, ..., b_i^T\right)^T, \Lambda^d_{i1} = \frac{1}{JC}(\lambda_i - Jl) + 1 \\ V_d^{i\beta\neq 1} = \left(\sqrt{\sum\limits_{\alpha=1}^{l}\left(q_\alpha^{\beta\neq 1}\right)^2}\right)^{-\frac{1}{2}} \cdot \left(q_1^{\beta\neq 1} b_i^T, ..., q_l^{\beta\neq 1} b_i^T\right)^T, \Lambda^d_{i\beta\neq 1} = \frac{1}{JC}\lambda_i + 1 \end{array}\right. .(27)$$

Since $J > \frac{1}{l}\max\limits_{1\le i\le n}\{|\lambda_i|\}$ and $C > l$, by (27), we have

$$\left\{\begin{array}{l} -1 < \Lambda^d_{i1} < 1 \\ \Lambda^d_{i\beta\neq 1} > 0 \end{array}\right. . \qquad (28)$$

Clearly, it holds that $\Lambda_{i\beta} + 1 > 0$ for all i, β. By Lemma 2, the network is convergent.

The proof is completed.

Theorem 4. *Suppose the network (19) satisfies Theorem 3. If there exists constant h such that $-Jl - \eta < h \le -J(l-2) - \eta$, then a stable equilibrium of CLM in network (19) has in each column i at most one activated neuron $x^*_{i\alpha}$ with $\sum\limits_{j=1}^{n} f_{ij}x^*_{j\alpha} \ge \sum\limits_{j=1}^{n} f_{ij}x^*_{j\beta}$.*

Proof. Now assume the contrary: suppose an equilibrium x^* has at least two neurons which activities are bigger than -1 in a column i at two layer α and β, which means that $x^*_{i\alpha} > -1$, and $x^*_{i\beta} > -1$. For the corresponding equilibrium y^* of network (24), we also have $y^*_{i\alpha} > -1$, and $y^*_{i\beta} > -1$. Then two cases will be considered.

Case I : $-1 < x^*_{i\alpha} < 1$ and $-1 < x^*_{i\beta} < 1$. Clearly, by (25), we have $-1 < y^*_{i\alpha} < 1$ and $-1 < y^*_{i\beta} < 1$. By using the similar method in [25], it can prove that the network (24) has an energy function of the form

$$\begin{aligned} E(y(k)) &= -\frac{1}{2}\sigma^T(y(k))W_d\sigma(y(k)) - \sigma^T(y(k))H \\ &\quad +\sigma^T(y(k))y(k) \\ &\quad -\sum_{j=1}^{n}\sum_{\varphi=1}^{l}\int_0^{y_{j\varphi}(k)}\sigma(s)ds. \end{aligned} \tag{29}$$

Then at the position y^*, by (29), we have

$$\begin{aligned} & E^* = E(y(k))|_{y(k)=y^*} \\ = \ & \frac{1}{2}\sigma^T(y^*)W_d\sigma(y^*) - \sum_{j=1}^{n}\sum_{\varphi=1}^{l}\int_0^{y^*_{j\varphi}(k)}\sigma(s)ds. \end{aligned} \tag{30}$$

Now consider a small perturbation $y^{'} = y^*+\varepsilon$ near y^*, and $\varepsilon = (\varepsilon_{11}, \ldots, \varepsilon_{nl}) \in R^{nl}$, where $\varepsilon_{i\alpha} \neq 0$, $\varepsilon_{i\beta} = -\varepsilon_{i\alpha}$ and $\varepsilon_{j\varphi} = 0$ for all other j, φ. Then, by (29),(30), we have

$$\begin{aligned} & \Delta E = E(y(k))|_{y(k)=y'} - E(y(k))|_{y(k)=y^*} \\ = \ & -\frac{f_{ii}}{C}\varepsilon^2_{i\alpha} < 0. \end{aligned}$$

So, x^* can't be a stable equilibrium for the system.

Case II: There are some neurons' final outputs which equal to 1, and one neuron's final output is no more than 1 and bigger than -1. Clearly, this part of the proof is similar to that in Theorem 1 and is skipped here.

This completes the proof.

Theorem 5. *Suppose the network (19) satisfies Theorem 4. If there exist constants J, C and h such that*

$$\begin{cases} J > \max\{\frac{1}{l}\max\limits_{1\leq i\leq n}\{|\lambda_i|\}, \frac{\eta}{2}\} \\ -Jl < h \leq -J(l-2) - \eta \\ Jl \gg \lambda_i \end{cases},$$

then a stable equilibrium of CLM in network (19) has at least one activated column.

Proof. In order to prove that networks (19) has at least one activated column, we just need to prove $x_0 = [-1, -1, \dots, -1]^T \in \Re^{nl}$ can't be a stable equilibrium.

Denote $E_{i\alpha}(k) = x_{i\alpha}(k) + \frac{h}{JC} + \frac{1}{JC}\sum_{j=1}^{n} f_{ij}x_{j\alpha}(k) - \frac{1}{C}\sum_{\varphi=1}^{l} x_{i\varphi}(k)$. We can consider the CLM dynamical system (1) as

$$x_{i\alpha}(k+1) = \begin{cases} 1 & for\ E_{i\alpha}(k) \geqslant 1 \\ -1 & for\ E_{i\alpha}(k) \leqslant -1 \\ E_{i\alpha}(k) & for\ |E_{i\alpha}(k)| \leqslant 1 \end{cases}$$

for $i = 1, \dots, n$, $\alpha = 1, \dots, l$.

Therefore, in the region $\Phi = \{x_{i\alpha} | |x_{i\alpha}| \leq 1, i = 1, \dots, n, \alpha = 1, \dots, l\}$, we can discuss the dynamics of network (19) by investigating such linear dynamics

$$x_{i\alpha}(k+1) = E_{i\alpha}(k) \tag{31}$$

except for some boundary conditions related to $E_{i\alpha}(k)$. The equivalent vector form of network (31) is

$$x(k+1) = W_d x(k) + H. \tag{32}$$

If $Jl \gg \lambda_i$ and $-Jl < h \leq -J(l-2) - \eta$, there is an approximate equilibrium x^F of the linear system (31) in Φ [19]: $x^F \approx [\frac{h}{Jl}, \frac{h}{Jl}, \dots, \frac{h}{Jl}]^T \in \Re^{nl}$.

For the linear system (31), if x_0 is a stable equilibrium of network (19), it must satisfy $x_{i\alpha}(k+1)|_{x_{i\alpha}(k)=x_0} \leq -1$ for all i, α. Since $x_0 < x^F$, at x_0, we have $x(k+1)|_{x(k)=x_0} - x^F \leq x_0 - x^F < 0$, which means

$$\begin{aligned} &(x(k+1) - x^F)^T (x(k+1) - x^F)|_{x(k)=x_0} \\ &\quad \geq (x(k) - x^F)^T (x(k) - x^F)|_{x(k)=x_0}. \end{aligned} \tag{33}$$

By (32), we have $(x(k+1) - x^F) = W_d(x(k) - x^F)$ and

$$x(k) = \sum_{i=1}^{n}\sum_{\alpha=1}^{l} \xi_{i\alpha}(k) V_d^{i\alpha}, \tag{34}$$

where $\xi_{i\alpha}(k)$ is the projection of $x(k)$ on $V_d^{i\alpha}$.

The projections on both AC and DC subspaces can be expressed by (27)

$$\begin{cases} P_d^{DC} = \sum\limits_{i=1}^{n} V_d^{i1}(V_d^{i1})^T = \dfrac{1}{l}\Pi_l \otimes I_n \\ P_d^{AC} = \sum\limits_{\beta=2}^{l}\sum\limits_{i=1}^{n} V_d^{i\beta}(V_d^{i\beta})^T = I_{nl} - \dfrac{1}{l}\Pi_l \otimes I_n \end{cases}.$$

At the point $x(k) = x_0$, by (34), we have

$$\begin{aligned} & (x(k+1) - x^F)^T(x(k+1) - x^F) \\ = \ & (W_d(x(k) - x^F))^T(W_d(x(k) - x^F)) \\ = \ & (x(k) - x^F)^T V\Psi V^T(x(k) - x^F), \end{aligned} \qquad (35)$$

where $V = [V_d^{11}, \cdots, V_d^{n1}, V_d^{12}, \cdots, V_d^{nl}] \in R^{nl\times nl}$ and $\Psi =$ is a $nl\times nl$ diagonal matrix $\Psi = diag((\Lambda_{11}^d)^2, (\Lambda_{21}^d)^2, ..., (\Lambda_{n1}^d)^2, 0, ..., 0)$.

From (28), (35), we have

$$\begin{aligned} & (x(k+1) - x^F)^T(x(k+1) - x^F) \\ < \ & (x(k) - x^F)^T VV^T(x(k) - x^F) \\ = \ & (x(k) - x^F)^T(x(k) - x^F). \end{aligned} \qquad (36)$$

Therefore, by (33),(36), we can find the contradiction that x_0 is a stable equilibrium.

This completes the proof.

8.2. Implementation of CLM-CNN-Dynamics

1. Initialize all $x_{i\alpha}(0)$ with small random values around $x_{i\alpha}(0) \in (-1, -1+\epsilon]$. Calculate the largest eigenvalue λ_{max}, the smallest eigenvalue λ_{min} of the matrix f. Set $C_0 = l$, $J_0 = \max\{\frac{l}{2}\max\{|\lambda_{max}|, |\lambda_{min}|\}, \frac{\eta}{2}\}$, $J = 1.01 * J_0$, $C = 1.01 * C_0$, $\alpha_f = \frac{1}{C*J}$, $\tau = 1.01$, $\tau_{max} = 1.35$, $\rho = 0.001$, $\omega = 1$, $\zeta = 1$, $N_{queue} = 0$, $N^1_{X^1(K+1)} = 0$, $N^1_{queue} = 100$, $X_0 = [-1, \cdots, -1]_{n\times 1}$, $T = \lambda_{max}$, $\eta_T = 0.99$, $k = 0$, $h = -J(l-1)$, $E_C = -\frac{1}{C}\Pi_l$, $X(0) = \begin{bmatrix} x_{11}(0) & \cdots & x_{1l}(0) \\ \vdots & \ddots & \vdots \\ x_{n1}(0) & \cdots & x_{nl}(0) \end{bmatrix}_{n\times l}$, $H = -\frac{1}{C*J}\begin{bmatrix} h & \cdots & h \\ \vdots & \ddots & \vdots \\ h & \cdots & h \end{bmatrix}_{n\times l}$;
2. Calculate $\xi = H + \alpha_f * f * X(k) - \alpha_f * T * X(k) - X(k) * E_C + X(k)$. if $mod(k+1, l) == 0$, then $T = T * \eta_T$;

3. If $\zeta == 1$, then $\gamma = |\xi * (-X_0)^T/l - X_0| - |X(0) * (-X_0)^T/l - X_0|$; Otherwise go to Step 4. If $\gamma <= 0$, then $J = J * \tau$, $\alpha_f = \frac{1}{C*J}$, go to Step 2; Otherwise set $\zeta = 0$, go to Step 4;
4. Set $X(k+1) = \max(\xi, 0)$;
5. If $\omega == 1$, then find $N_{X(K+1)}$, which is the number of $x_{i\alpha}(k+1) == -1$ in $X(k+1)$. If $N_{X(K+1)} < n$, then $X(k) = X(k+1)$, $k = k+1$, go to Step 2; Otherwise set $\omega = 0$, go to Step 6;
6. Calculate the error between $X(k)$ and $X(k+1)$. If $|x_{i\alpha}(k+1) - x_{i\alpha}(k)| < \rho$ for all i, α, then go to Step 7; Otherwise $X(k) = X(k+1)$, $k = k+1$, go to Step 2;
7. Find $X_0(K+1)$, here $X_0(K+1) = \{x_{i\alpha}(k+1) | x_{i\alpha}(k+1) > -1 + \zeta\}$ for all i, α. Set $X^1(K+1) = X_0(K+1) - Mean(X_0(K+1))$, where $Mean(X_0(K+1))$ is the mean number of $X_0(K+1)$, then find $N_{X^1(K+1)}$, the number of $x^1_{i\alpha}(k+1) > 0.1/n$ in $X^1(k+1)$. If $N_{X^1(K+1)} == 0$, then go to the end; Otherwise go to Step 8;
8. If $N_{queue} == 0$, then $N_{queue} = N_{queue} + 1$, $N^1_{X^1(K+1)} = N_{X^1(K+1)}$, go to Step 11; Otherwise go to Step 9;
9. if $N^1_{X^1(K+1)} \neq N_{X^1(K+1)}$, then $N_{queue} = 0$, $N^1_{X^1(K+1)} = N_{X^1(K+1)}$, go to Step 11; Otherwise $N_{queue} = N_{queue} + 1$, Go to Step 10;
10. If $N_{queue} == N^1_{queue}$, then go to the end; Otherwise go to Step 11;
11. Set $J = J * \tau$, $\tau = \tau + 0.01$, $\alpha_f = \frac{1}{C*J}$, $X(k) = X(k+1)$, $k = k+1$. If $\tau < \tau_{max}$, then go to Step 2; Otherwise go to the end.

For simulation without self-inhibitory annealing simply set $T = 0$. Although the condition $Jl \gg \lambda_i$ is hard to evaluate, we can solve this problem approximately by increasing J step by step to make sure x^F located in $(-1, 1)$, which can be found in Step 3. In Step 11, the reason of increasing J is to pursue better performance. From Step 7 to Step 11, we use N_{queue} to track the change of $X(k)$ and stop the iteration when $N_{queue} == N^1_{queue}$, which means that the range and variety of $X(k)$ is very small at that time.

8.3. Implementation of CLM-LT-Dynamics

The CLM-LT-dynamics can be implemented as:

1. Initialize all $x_{i\alpha}(0)$ with small random values around $x_{i\alpha}(t=0) \in (h_i/l - \epsilon, h_i/l + \epsilon]$. In our simulations, h_i is set to 1. Calculate the largest eigenvalue λ_{max}, $T = \lambda_{max}$, $J_w = 1.1 * \max\limits_{1 \le j \le n} \{\sum\limits_{i=1}^{n} \max\{f_{ij}, 0\}\}$;

2. Do $n \times l$ times: Choose (i,α) randomly and update $x_{i,\alpha} = \{\max 0, \xi\}$, where
$$\xi = \frac{J_w h_i - J_w \sum_{\beta \neq \alpha} x_{i\beta} + \sum_{j \neq i} f_{ij} x_{j\alpha}}{J_w - f_{ii} + T}$$
3. Decrease T by $T = \eta_T * T$, here we set $\eta_T = 0.99$. Go to Step 2 until convergence.

For simulation without self-inhibitory annealing simply set $T = 0$. More details can be found in [19].

8.4. Lateral Interaction for Image Segmentation

In this article, the lateral interaction f is identical in all l layers and is symmetric.

$$f_{ij} = \Gamma(\phi_{i,j}) = \begin{cases} 0.5, & i == j \\ \frac{\phi_{i,j}}{\max\limits_{1 \leq i',j' \leq n} \{\phi_{i',j'}\}}, & i \neq j, \phi_{i,j} \geq 0 \\ \frac{\phi_{i,j}}{\max\limits_{1 \leq i',j' \leq n} \{-\phi_{i',j'}\}}, & i \neq j, \phi_{i,j} < 0 \end{cases}.$$

The compatibility $\phi_{i,j}$ between pixel $P(i_x, i_y)$ and $P(j_x, j_y)$ with gray-level $g(i_x, i_y)$ and $g(j_x, j_y)$ is given by $\phi_{i,j} = m_1 e^{-\nu/k_1}(m_2 e^{-d/k_2} + 1) - \theta$, where $v = |g(i_x, i_y) - g(j_x, j_y)|$ is the difference between the two gray-levels, $d = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2}$ is the Euclidean distance of the two pixels, $m_1(> 0)$ and $m_2(> 0)$ balance between the influence of v and d to ϕ, $k_1(> 0)$ controls the sharpness of v, $k_2(> 0)$ controls the spatial range of d, and θ is the threshold. The parameters used in this paper are: $m_1 = 2$, $m_2 = 3$, $k_1 = 100$, $k_2 = 0.5$, $\theta = 1.7$.

References

[1] Chua, L. O. and L. Yang (1988*a*). Cellular neural networks: Application. *IEEE Trans. Circuits Syst.* **35**, 1273–1290.

[2] Chua, L. O. and L. Yang (1988*b*). Cellular neural networks: Theory. *IEEE Trans. Circuits Syst.* **35**, 1257–1272.

[3] Computational Intelligence Laboratory, Electrical and University of Louisville Computer Engineering. http://ci.louisville.edu/zurada/publications/2010/Ex5ForCNNCLM.rar.

[4] Feng, Jianfeng (1997). Lyapunov functions for neural nets with nondifferentiable input-output characteristics. *Neural Computation* **9(1)**, 43–49.

[5] Hahnloser, Richard H. R. (1998). On the piecewise analysis of networks of linear threshold neurons. *Neural Networks* **11**, 691–697.

[6] Hahnloser, Richard H. R., Rahul Sarpeshkar, Misha A. Mahowald, Rodney J. Douglas and H. Sebastian Seung (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature* **405**, 947–951.

[7] Hahnloser, Richard H.R., H. Sebastian and Jean-Jacques Slotine (2003). Permitted and forbidden sets in symmetric threshold-linear networks. *Neural Computation* **15(3)**, 621–638.

[8] Hänggi, Martin and George S. Moschytz (2000). *Cellular Neural Networks: Analysis, Design and Optimization.* Springer.

[9] Horn, Roger A. and Charles R. Johnson (1985). *Matrix Analysis.* Cambridge University Press.

[10] Kirk, David B. and Wen mei W.Hwu (2010). *Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series).* Morgan Kaufmann.

[11] Koffka, Kurt (1962). *Principles of Gestalt psychology.* Routledge & Paul.

[12] Ljung, Lennart (1977). Analysis of recursive stochastic algorithms. *IEEE Trans. Autom. Control* **22**, 551–575.

[13] Ontrup, Jörg and Helge Ritter (1998). Perceptual grouping in a neural model: Reproducing human texture perception. technical report SFB360-TR-98/6. Technical report. University of Bielefeld.

[14] Ritter, Helge (1990). A spatial approach to feature linking. *Proceedings of the International Neural Network Conference Paris. Kluwer Publishers* **2**, 898–901.

[15] Slavova, Angela (2003). *Cellular neural networks : dynamics and modelling.* Kluwer Academic Publishers.

[16] von der Malsburg, Christoph (1981). The correlation theory of brain function. technical report 81-2. Technical report. MPI Göttingen.

[17] von der Malsburg, Christoph (1995). Binding in models of perception and brain function. *Current Opinion in Neurobiology* **5**, 520–526.

[18] Weng, Sebastian, Heiko Wersing, Jochen J. Steil and Helge Ritter (2006). Learning lateral interactions for feature binding and sensory segmentation from prototypic basis interactions. *IEEE Transactions on Neural Networks* **17(2)**, 843–862.

[19] Wersing, Heiko, Jochen J. Steil and Helge Ritter (2001*a*). A Competitive Layer Model for feature binding and sensory segmentation. *Neural Computation* **13**, 357–387.

[20] Wersing, Heiko, Wolf-Jürgen Beyn and Helge Ritter (2001*b*). Dynamical stability conditions for recurrent neural networks with unsaturating piecewise linear transfer functions. *Neural Computation* **13**, 1811–1825.

[21] Wu, Chai Wah and Leon O. Chua (1997). A more rigorous proof of complete stability of cellular neural networks. *IEEE Transactions on Circuits and SystemsI: Fundamental Theory and Applications* **44(4)**, 370–371.

[22] Xie, Xiaohui, Richard H. R. Hahnloser and H. Sebastian Seung (2002). Selectively grouping neurons in recurrent networks of lateral inhibition. *Neural Computation* **14**, 2627–2646.

[23] Yi, Zhang (2010). Foundations of implementing the Competitive Layer Model by Lotka-Volterra recurrent neural networks. *IEEE Transactions on Neural Networks* **21(3)**, 494–507.

[24] Yi, Zhang and Kok Kiong Tan (2004*a*). *Convergence Analysis of Recurrent Neural Networks (Network Theory and Applications, V.13)*. Kluwer Academic Publishers.

[25] Yi, Zhang and Kok Kiong Tan (2004*b*). Multistability of discrete-time recurrent neural networks with unsaturating piecewise linear activation functions. *IEEE Transactions on Neural Networks* **15(2)**, 329–336.

[26] Yi, Zhang, Kok Kiong Tan and T.H. Lee (2003). Multistability analysis for recurrent neural networks with unsaturating piecewise linear transfer functions. *Neural Computation* **15**, 639–662.

[27] Yi, Zhang, Mao Ye, Jian Cheng Lv and Kok Kiong Tan (2005). Convergence analysis of a deterministic discrete time system of Oja's PCA learning algorithm. *IEEE Transactions on Neural Networks* **16(6)**, 1318–1328.

[28] Zhang, Lei, Zhang Yi and Jiali Yu (2008). Multiperiodicity and attractivity of delayed recurrent neural networks with unsaturating piecewise linear transfer functions. *IEEE Transactions on Neual Networks* **19(1)**, 158–167.

[29] Zhang, Lei, Zhang Yi, Stones Lei Zhang and Pheng Ann Heng (2009). Activity invariant sets and exponentially stable attractors of linear threshold discrete-time recurrent neural networks. *IEEE Transactions on Automatic Control* **54(6)**, 1341–1347.

[30] Zhou, Wei and Jacek M. Zurada (2010). Competitive layer model of discrete-time recurrent neural networks with lt neurons. *Neural Computation* **22**, 2137–2160.