CHAPTER 2

REVIEW OF CELLULAR NONLINEAR NETWORKS AND THEIR APPLICATIONS

2.1 FUNDAMENTALS OF CELLULAR NONLINEAR NETWORKS

Due to the advantageous feature of local connectivity, the cellular nonlinear (neural) network (CNN) introduced by Chua and Yang [55] is very suitable for VLSI implementation and used in many applications [56]-[58], [64], [89]-[90]. One of CNN application is as neural associative memories for pattern learning, recognition, and association [61], [87]-[103], [107]-[109], [112]-[117]. Among them, many innovative algorithms and software simulations of CNN associated memories were reported [87]-[88], [107]-[109]. As to the hardware implementation, special learning algorithm, and digital hardware implementation for CNN were proposed in [110]-[112] to solve the sensitivity problems are caused by the limited precision of analog weights. Moreover, CMOS chip implementation of CNN associative memory was also reported in [110]. The stability analysis for CNN is discussed in [148]-[152].

2.1.1 Basic Definition and Mathematical Foundation of CNN [55]-[58], [64-[65], [79], [139]-[140]

(1) Basic Definition and Mathematical Foundation

The fundamental concepts and architectures used in CNN are reviewed. For image processing purposes, usually a 2D CNN structure as shown in Fig. 2.1 is used, where each box denotes a cell. The basic structure of CNN is similar to a typical biological cellular cell is shown in Fig 2.2. The CNN array shown in Fig. 2.1 is called an MxN CNN which consists of M rows and N columns of cells. A cell is a basic CNN unit. A cell has direct coupling only to its near neighbors. This local connectedness is formally defined by the following definitions [55]-[58].

Definition 1: Standard CNN Architecture

A standard CNN architecture is shown in Fig. 2.1, and it consists of an MxN rectangular array of cells $C(i,j)$ with Cartesian coordinate (i,j), $i = 1,...,M$, $j = 1,...,N$.

Definition 2: Sphere of Influence of Cell $C(i,j)$

The sphere of influence, $N_r(i,j)$, of radius r of cell $C(i,j)$ is defined to be the set of all neighboring cells satisfying the following property

$$N_r(i,j) = \{C(k,l) \mid \max_{1 \le k \le M, 1 \le l \le N} \{|k-i|, |l-j|\} \le r\} \quad (2.1)$$

where r is a positive integer.

$N_r(i,j)$ in (2.1) is usually referred the as $(2r+1)$x$(2r+1)$ neighborhood. Fig. 2.3(a) shows the $r = 1$ (3x3) neighborhood, Fig. 2.3(b) shows the $r = 2$ (5x5) neighborhood, and Fig. 2.3(c) shows the $r = 3$ (7x7) neighborhood. In the CNN, every cell is connected to all its neighboring cells in $N_r(i,j)$ via "synaptic" or "weight" circuits. If r = N-1 and M = N, a fully connected CNN is corresponds to the classic Hopfield network. $N_r^0(i,j)$ is the set of r-neighboring cells $N_r(i,j)$ without the center cell $C(i,j)$.

Definition 3: Regular and Boundary Cells.

A cell C(i,j) is called a regular cell with respect to $N_r(i,j)$ if and only if all neighboring cells C(k,l) $\in N_r(i,j)$ exist. Otherwise, C(i,j) is called a boundary cell. The outermost boundary cells are called the edge cells. Not all boundary cells is edge cell if $r > 1$. The illustration of each cell is shown in Fig. 2.4.

Definition 4: A class 1 standard CNN

A class 1 MxN standard CNN is defined by an MxN rectangular array of cells C(i,j) located at site (i,j), $i = 1,...,M$, $j = 1,...,N$. Each cell C(i,j) is defined by the following equations:

1. State Equation of cell C(i,j)

$$C_{ij}\dot{x}_{ij}(t) = -\frac{1}{R_{ij}}x_{ij} + \sum_{C(k,l)\,N_r(i,j)} a_{ijkl}\,y_{kl} + \sum_{C(k,l)\,N_r(i,j)} b_{ijkl}\,u_{kl} + z_{ij} \tag{2.2}$$

where $x_{ij} \in R$, $y_{kl} \in R$, $u_{kl} \in R$, and $z_{ij} \in R$ denote the state variable, output, input, and threshold of the cell C(i,j), respectively, and R is the set of real number. a_{ijkl} and b_{ijkl} denote the feedback and input synaptic operators between cells C(i,j) and C(k,l), respectively. The bias is z_{ij} (also called threshold) of the cell C(i,j), which may be static, temporal-variant, spatial-invariant, or spatial-variant. $C_{ij} > 0$ and $R_{ij} > 0$ are the values of the capacitor and resistor, respectively.

2. Output Equation of cell C(i,j)

$$y_{ij} = f(x_{ij}) = \frac{1}{2}|x_{ij}+1| + \frac{1}{2}|x_{ij}-1| \tag{2.3}$$

This function $f(x_{ij})$ is also called as the activation function. There are several activation function are applied to the neural networks as shown in section 2.2. The standard piecewise linear output function $y_{ij} = f(x_{ij})$ is most used by standard CNN.

3. Input equation of cell C(i,j)

$$u_{ij} = f_u(E_{ij}) \tag{2.4}$$

where E_{ij} is the input signal, for example, the intensity of light detected by an embedded optical sensor in the cell C(i,j). In the elementary CNN, $f_u(.)$ is used to normalize the detected signal to a proper range.

4. Boundary Conditions

Boundary conditions are part of CNN design for different tasks. Some CNNs may fail to work correctly without specified boundary conditions while others are independent of boundary conditions. There are three kinds of boundary conditions as Fixed (Dirichlet) boundary conditions (Fig. 2.5)

Each boundary cell has fixed input and fixed output. The fixed input and output are usually spatial-invariant.

Zero-flux (Neumann) boundary conditions (Fig. 2.6)

The input and output of a boundary cell are the same as that of the regular cells at the symmetry position with respect to the boundary cells.

Periodic (Toroidal) boundary conditions (Fig. 2.7)

The neighboring cells of a boundary cell are those symmetrical with respect to the center of CNN. For an MxN regular cells array CNN, the outputs and inputs of boundary cells are listed as follow:

Left boundary cells: $y_{i0}=y_{iN}$, $u_{i0}=u_{iN}$, $i = 1, 2, \ldots, M$

Right boundary cells: $y_{i(N+1)}=y_{i1}$, $u_{i(N+1)}=u_{i1}$, $i = 1, 2, \ldots, M$

Top boundary cells: $y_{0j}=y_{Mj}$, $u_{0j}=u_{Mj}$, $j = 1, 2, \ldots, N$

Bottom boundary cells: $y_{(M+1)j}=y_{1j}$, $u_{(M+1)j}=u_{1j}$, $j = 1, 2, \ldots, N$

5. Initial State

$$x_{ij}(t=0) \qquad i=1,...,M; \quad j=1,...,N \tag{2.5}$$

$x_{ij}(t=0)$ is the cell state before the CNN starts the image processing. Sometimes, the processing image is stored in the cell state **X** instead the constant inputs **U**. The input u_{kl} is usually the pixel intensity of an MxN gray-scale image or picture,

normalized without loss of generality to have the range $-1 \le u_{kl} \le +1$ where white is coded by -1 and black is coded by +1. In the most general case, both a_{ijkl} and b_{ijkl} are nonlinear operators which operate on $x_{kl}(t)$, $y_{kl}(t)$, $u_{kl}(t)$, $x_{ij}(t)$, $y_{ij}(t)$, and $u_{ij}(t)$, $0 \le t \le t_0$, to produce a scalar $(a_{ijkl} y_{kl})(t_0)$ and $(b_{ijkl} u_{kl})(t_0)$, $0 \le t \le t_0$.

In VLSI implementation of CNNs, u_{ij}, x_{ij}, and y_{ij} are three voltages and z_{ij} is a bias current. The block diagram of a cell realized by electronic circuits is shown in Fig. 2.8, where all diamond-shape symbols denote a voltage-controlled current source that injects a current proportional to the indicated controlling voltage u_{kl} or y_{kl} and weighted by b_{ijkl} or a_{ijkl}. In the internal core which is a nonlinear voltage-controlled current source, the state equation (2.2) is realized with the output voltage $y_{ij} = f(x_{ij})$.

Definition 5: Spatial-invariant or isotropic CNN

A CNN is spatial-invariant or isotropic if and only if both the synaptic operators a_{ijkl} and b_{ijkl} and the threshold z_{ij} do not vary with space. In this case, we have

$$\sum_{C(k,l)\ Nr(i,j)} a_{ijkl} y_{kl} = \sum_{u=-r}^{r} \sum_{v=-r}^{r} a_{uv} y_{kl} \qquad u=k-i, v=l-j$$

$$\sum_{C(k,l)\ Nr(i,j)} b_{ijkl} y_{kl} = \sum_{u=-r}^{r} \sum_{v=-r}^{r} b_{uv} y_{kl} \qquad u=k-i, v=l-j$$

$$z_{ij} = z \tag{2.6}$$

Definition 6: A class 2 standard CNN

A class 2 *M*x*N* standard CNN with linear synaptic operators is defined by the following state equation (using the same notation as in (2.2)):

$$C_{ij}\,\dot{x}_{ij} = -\frac{1}{R_{ij}}x_{ij} + \sum_{C(k,l)\ N_r(i,j)} a_{ijkl}\,y_{kl} + \sum_{C(k,l)\ N_r(i,j)} b_{ijkl}\,u_{kl} + z_{ij} + \sum_{C(k,l)\ N_r(i,j)} c_{ijkl}\,x_{kl} + \sum_{C(k,l)\ N_r(i,j)} d_{ijkl}\,(u_{kl}, x_{kl}, y_{kl}) \tag{2.7}$$

where the notations a_{ijkl} and b_{ijkl} are the same as those in (2.2), the synaptic weights c_{ijkl} (template **C**) and d_{ijkl} (template **D**) depending on the states and mixed variables (inputs, states, and outputs), respectively.

(2) Three CNN classes

Each CNN is uniquely defined by 3 cloning templates {**A**, **B**, **Z**}, which consists of 19 real numbers for a 3x3 neighborhood (r = 1).

Definition 7: Excitatory and inhibitory synaptic weights

A feedback synaptic weight a_{ijkl} is said to be excitatory (inhibitory) if and only if it is positive (negative).

A synaptic weight is excitatory (inhibitory) because it increases (decreases) the derivation of state $\dot{x}$, namely the rate of growth of $x_{ij}(t)$. The signal flow structure of a standard CNN {**A**, **B**, **Z**} with a single neighborhood $N_1(i,j)$ is shown in Fig. 2.9(a). The 2 shaded cones symbolize the weighted contributions of input and output voltages of the cell C(k,l) $\in N_1(i,j)$ to the state voltage. The system structure of a cell C(i,j) is shown in Fig. 2.9 (b) where arrows printed in bold mark parallel data paths from the input u_{kl} and the output y_{kl} of the neighboring cells C(k,l), respectively. Arrows on thinner lines denotes the threshold z_{ij}, input u_{ij}, state x_{ij}, and output y_{ij}, respectively.

Definition 8: Zero-feedback (feed-forward) class {**0**, **B**, **Z**}

A CNN belongs to the zero-feedback class {**0**, **B**, **Z**} if and only if all feedback

template weights are zero, i.e., $\mathbf{A} \equiv 0$.

Each cell of a zero-feedback CNN is described by

$$C_{ij}\,\dot{x}_{ij} = -\frac{1}{R_{ij}}x_{ij} + \sum_{C(k,l)\in Nr(i,j)} b_{ijkl}\,u_{ij} + z_{ij} \tag{2.8}$$

Fig. 2.10(a) shows the signal flow structure of a zero-feedback (feed-forward) CNN with a 3x3 neighborhood. The cone symbolizes the weighted contributions of input voltages of the cells $C(k,l)\in N_1(i,j)$ to the center cell $C(i,j)$. Fig. 2.10(b) shows the system structure of a cell $C(i,j)$.

Definition 9: Zero-input (Autonomous) class {**A**, **0**, **Z**}

A CNN belongs to the zero-input class {**A**, **0**, **Z**} if and only if all feed-forward template weights are zero, i.e., $\mathbf{B} \equiv 0$.

Each cell of a zero-input CNN is described by

$$C_{ij}\,\dot{x}_{ij} = -\frac{1}{R_{ij}}x_{ij} + \sum_{C(k,l)\in Nr(i,j)} a_{ijkl}\,y_{ij} + z_{ij} \tag{2.9}$$

The signal flow structure of a zero-input (Autonomous) CNN with a single neighborhood is shown in Fig. 2.11(a). The shaded cone symbolizes the weighted contributions of the output voltage of the cells $C(k,l)\in N_1(i,j)$ to the center cell $C(i,j)$. In the Fig. 2.11(b), the system structure of a center cell $C(i,j)$ is presented. In this case, there are no input signals.

Definition 10: uncoupled (scalar) class {$\mathbf{A}^0$, **B**, **Z**}

A CNN belongs to the uncoupled class {$\mathbf{A}^0$, **B**, **Z**} if and only if $a_{ijkl} = 0$ except $ij = kl$, i.e., $\overline{\mathbf{A}} \equiv 0$

Each cell of an uncoupled CNN is described by

$$C_{ij}\,\dot{x}_{ij} = -\frac{1}{R_{ij}}x_{ij} + a_{00}\,f(x_{ij}) + \sum_{C(k,l)\in Nr(i,j)} b_{ijkl}\,u_{kl} + z_{ij} \tag{2.10}$$

Fig. 2.12(a) shows the signal flow structure of an uncoupled CNN with a single

neighborhood. The system structure of a center cell C(i,j) is shown in Fig. 2.12(b).

2.1.2 Activation Functions

A rule, which gives the effect of all inputs on the activation of the cell, is needed. Often, the rule that is also called as the activation function $f(x)$ is a non-linear, mono-increasing function of the weighted sum of its inputs. Generally, some sort of threshold function is applied: a hard limit threshold function (step function or sign function), or a semi-linear or piecewise linear function (ramp function), or a smoothly limiting threshold function (sigmoid function) [138]. These activation functions are formulated as follows:

1. Step function

$$y = f(x) = \mathrm{sgn}(x) = \begin{cases} 1 & x \equiv 0 \\ -1 & x < 0 \end{cases} \tag{2.11}$$

2. Ramp function

$$y = f(x) = \begin{cases} -1 & x \leq -1 \\ x & -1 < x < 1 \\ 1 & x \equiv 1 \end{cases} \tag{2.12}$$

3. Sigmoid function

$$y = f(x) = \frac{2}{1 + e^{-nx}} \quad 1 \qquad n = 3 \tag{2.13}$$

4. Gaussian function

$$y = f(x) = e^{-n x^2} \quad n = 1 \tag{2.14}$$

where x is the cell state, y is the cell output of the activation function, and n is the order that represents the grade of transferring curves.

The transferring characteristic curves are shown in Fig. 2.13. Which activation function is used is decided by the choice of the neural network. Generally, the step function and ramp function is usable to the software simulation for the simple

calculation, but they are hard to be implemented in the VLSI technology because of the high gain or piecewise linearity. The sigmoid function and Gaussian function are much similar to the practical transfer function in the VLSI technology, but it needs complicated calculation in the software simulation. Thus, all the four activation functions can be implemented in the software simulation, but only the sigmoid function and Gaussian function can be implemented by the VLSI technology.

2.2 APPLICATIONS OF CELLULAR NONLINEAR NETWORKS

A compilation of the selected templates for cellular nonlinear (neural) networks is presented [55]-[58]. Many items or structures are used in the CNN for variety applications. Some representative applications are shown below. The templates applied in the CNN for simulating Muller-Lyer optical illusion [57], for halftoning [57]-[58], for discrete-time CNN [139], [142], for gray-scale contour detection [95], [113], [179] for gray-diagonal detection [58], for Herring-grid illusion [58], for information coding/decoding [132], for associative memory [61], [94] for character recognition [116], for artificial network [161] applications. Besides the application on associative memory, the image processing is another important application for the CNN. A multistage CNN called CNN universal machine (CNNUM) is developed for handling kinds of image processing.

2.2.1 Cellular Nonlinear Network Applications [55]-[58], [139]-[142]

Since the invention of CNN (L. O. Chua and L. Yang, 1988) [55]-[56], different CNN structures have been proposed for different applications and from different biological models. After the original CNN is invented, another main CNN branch,

discrete-time CNN (DTCNN), was proposed by Harrer in 1994 [142]. The DTCNN is specified by a set of difference equations that represent the discrete dynamics of cells and the nonlinearity of the output activation function. The standard DTCNN presented by Harrer and Nossek in 1992 is given by

State equation of cell C(i,j)

$$x_{ij}[n+1] = \sum_{C(k,l)\in Nr(i,j)} a_{ijkl} y_{kl}[n] + \sum_{C(k,l)\in Nr(i,j)} b_{ijkl} u_{kl}[n] + z_{ij}[n] \tag{2.15}$$

Output equation of cell C(i,j)

$$y_{ij}[n] = f(x_{ij}[n]) = \mathrm{sgn}(x_{ij}[n]) \tag{2.16}$$

where $x_{ij}[n]$ and $z_{ij}[n]$ are the state and threshold of the cell C(i,j), respectively. The output activation function is not limited to only the hard-limit activation function; it can be any kind of function depending on applications and implementing platforms. The original CNN with continue-time dynamics is called as CTCNN to emphasize the difference between the CTCNN and DTCNN. However, the CNN community trends to used CNN and CTCNN interchangeably. In this thesis, the CNN is used loosely as CTCNN for simplicity. Except the DTCNN, there are several CTCNNs being developed. They will be described in the follow section.

First one is the multiplayer CNN (MCNN). A MCNN uses more than one layer of CNN to perform a single task [55], [143]-[144]. Different MCNN prototypes were presented by Majorana and Chua [130]. One widely used form of nth-order CNN is a multilayer CNN that was presented by Chua and Yang [55]. The state equation of a K-layer CNN cell C(i,j) is given by

$$\dot{x}_{ij}^{[p]} = -x_{ij}^{[p]} + \sum_{q=1}^{K} \sum_{C(k,l)} a_{ijkl}^{[pq]} y_{kl}^{[q]} + \sum_{q=1}^{K} \sum_{C(k,l)} b_{ijkl}^{[pq]} u_{kl}^{[q]} + z_{ij}^{[p]} \tag{2.17}$$

$$1 \le p \le K; \quad 1 \le i \le M; \quad 1 \le j \le N$$

where $x_{ij}^{[p]}$ ($z_{ij}^{[p]}$) are the state (threshold) of pth-layer cell C(i,j), and $y_{kl}^{[p]}$ ($u_{kl}^{[p]}$) are

the outputs (inputs) of *p*th-layer neighboring cells C(*k*,*l*) of the central cell C(*i*,*j*) whereas the $a_{ijkl}^{[pq]}$ ($b_{ijkl}^{[pq]}$) are the feedback (feed-forward) weights between the *p*th and *q*th layer. So far, the MCNN has applied to process the binary image processing engine, the linear filters, and adaptive edge detection [143]-[145].

In motion-related applications (T. Roska and L. O. Chua, 1993) [67], time delays are introduced into CNN structures and result in the kinds of delay-type CNNs (DCNNs) that are defined by T. Roska and L. O. Chua [67]-[68] as

$$C_{ij}\dot{x}_{ij} = -\frac{1}{R_{ij}}x_{ij} + \sum_{C(k,l)\in Nr(i,j)} a_{ijkl}y_{kl} + \sum_{C(k,l)\in Nr(i,j)} a_{ijkl}^{\tau}y_{kl}(t-\tau) + \sum_{C(k,l)\in Nr(i,j)} b_{ijkl}u_{kl} + \sum_{C(k,l)\in Nr(i,j)} b_{ijkl}^{\tau}u_{kl}(t-\tau) + z_{ij} \tag{2.18}$$

where τ is called the time delay. The delay-type templates provide us with even more flexibility and new applications, including the detection of some motion-related [67] CNN applications. There are some theoretical results on the stability of DCNN in works by Finnocchiaro and Perfetti [148], Gilli [147], T. Yang [139]. As DCNN is governed by a set of functional partial differential equations (PDE), some complex phenomena, for example, chaos, were observed by Civalleri and Gilli [146] even when only a small number of cells were used. Some results of predicting the chaotic sequence generated by chaotic DCNN are presented by Gilli [147].

Since only linear synaptic weights are not enough to deal with some image processing tasks where nonlinear properties are embedded, the CNN with nonlinear synaptic laws were introduced and called the nonlinear CNN (NCNN) [68]. The state equation of cell C(*i*,*j*) is defined by

$$C_{ij}\dot{x}_{ij} = -\frac{1}{R_{ij}}x_{ij} + A(y_{kl}(t),t) + B(u_{kl}(t),t) + Z(t) \tag{2.19}$$

where $\mathbf{A}(y_{kl}(t),t)$ ($\mathbf{B}(u_{kl}(t),t)$, and $\mathbf{Z}(t)$) denotes nonlinear template **A** (**B**, and **Z**) for the outputs (inputs and threshold) variables within $N_r(i,j)$. In this case, synaptic laws are

functions of time, outputs, inputs, and threshold within the neighborhood system. The NCNN provides a rigorous theoretical framework to solve many complex image-processing problems, for example, noise removal and feature extraction of gray-scale picture [68]. In addition to many new applications, the non-linear cloning templates allow us to model some biological properties of the retina, Moreover; it can also be used for modeling motion dynamics.

There also exist some other kinds of CNN structures such as chaotic CNN (CCNN), where every cell is a chaotic dynamic system [88], [93], [95], [154]-[155] that can be used to model some kinds of emergent behaviors and simulate some wave and pattern formation phenomena in an active medium [153]. In CCNN arrays, some nonlinear dynamic behaviors such as synchronization [154], cluttering, and cooperative phenomena were also found [155]. The existing results of CCNN consist of two main branches. One branch studies how to use the elementary CNN to generate chaotic signals and relevant applications. The other branch studies how to use chaotic elements as elementary cells to model spatial-temporal chaotic processes [156]. For the more complicated image processing, the multistage CNN is developed. A multistage CNN, called CNN universal machine (CNNUM), will be introduced in the next section.

2.2.2 Cellular Nonlinear Network Universal Machine

The CNN universal machine [157] is a programmable CNN. It can perform several complicated functions that cannot be simultaneously realized by the original CNN. Much research effort on the CNNUM has been undertaken and its implementation has been successfully demonstrated. Current CNNUM are based on the single-neighborhood (SN-CNN) structures [157]-[159].

The global architecture of the CNN universal machine [157] (CNNUM) is shown

in Fig. 2.14, where the analogical CNN universal cells are arranged on a regular grid. Fig. 2.15 shows the analog part of the analogical CNN universal cell. The CNN-UM consists of two main parts: (1) the array of analogical CNN universal cells and (2) the Global Ana logic Programming Unit (GAPU). As shown in Fig. 2.14, an analogical CNN universal cell has the following main additions to a CNN nucleus (core cell):

- Local analog memory (LAM): A few continuous (analog) values are stored in the LAM, cell by cell.
- Local logic memory (LLM): A few binary (logic) values are stored in the LLM, cell by cell.
- Local analog output unit (LAOU): A simple programmable multi-input-single output analog operation is executed and the input(s) and output are stored in LAM.
- Local communication and control unit (LCCU): It receives the messages from the central (global) commander, the GAPU, and programs the analogical CNN universal cells accordingly. The messages in the return directions are also possible.

The global analogical programming unit (GAPU) consists of four main parts:

- The analog program register (APR) that stores the templates (or their codes) used in the program.
- The logic program register (LPR) that stores the local logic operators (or their codes) used in the program.
- The switch configuration register (SCR) that stores the switch states (or their codes) and governing the cell configurations used in the program.
- The global analogical control unit (GACU) that is stored the physical machine code of the program.

By using these units, the CNNUM can be programmed to implement analogical CNN algorithms.

The CNN universal machine (CNNUM) [157] is not only an elementary CNN structure, but also a platform for integrating the flow of CNN operations. Moreover, the CNNUM is an important tool for organizing different kinds of CNN structures to perform complicated tasks that a single CNN cannot finish. The CNNUM can also be used to solve some global problems that are difficult to decompose in the CNN structure [158]-[160]. In fact, the CNNUM has been proved to be as universal as a Turing machine [162]. As it is only a platform for CNN operations, any kind of CNN should be included in the core of this platform, including DTCNN [142], and FCNN [163]-[164]. However, the current CNNUM has only an elementary CNN core. Thus this platform needs further improvement.

Local connection is the most significant characteristic of SN-CNN. Thus it can be easily implemented in VLSI technology. Furthermore, the SN-CNN can perform many useful functions in image signal processing.

However, the locally connected SN-CNN restricts their ability to solve complex problems that require large-neighborhood templates. Conceptually, each CNN cell can be connected to more than one layer of neighboring cells. Such a CNN is called the Large-Neighborhood Cellular Nonlinear (Neural) Network (LN-CNN). Recently, the initial design of symmetric LN-CNN has been proposed and implemented by a new device called the neuron BJT (νBJT) [31]-[32], [178].

The CNN universal machine [157] is a programmable CNN. It can perform several complicated functions that cannot be simultaneously realized by the original CNN. Much research effort on the CNNUM has been undertaken and its implementation has been successfully demonstrated. Current CNNUM are based on the SN-CNN structures [158]-[160].

In this thesis, a compact LN-CNN cell with synaptic structure to realize large cell arrays and special architecture to realize the large-neighborhood template is proposed. The LN-CNN cell is used to form the kernel unit of the original CNNUM so that the LN-CNNUM can be designed. The conceptual design, architecture, and realization of the LN-CNNUM are described. Software simulations are performed to verify the function of the LN-CNNUM with corresponding templates.

2.3 LEARNING RULES

The pattern recognition systems with ANN can be categorized into two distinct sorts on learning methodologies, one is supervised pattern recognition system, and the other is unsupervised pattern recognition system. The block diagrams of the supervised and unsupervised pattern recognition system are shown in Fig. 2.16. The learning method of supervised pattern recognition system is supervised learning, which is also called as learning with a teacher or learning with the desired output whereas the learning method of unsupervised pattern recognition system is unsupervised learning, and it is also called as learning without a teacher. Supervised learning in which an external teacher trains the ANN until the real output matches the desired output. Unsupervised learning in which external teacher's guidance is absent. Under this circumstance, the network adaptive weights base exclusively on the experiences with all the input patterns.

2.3.1 Supervised Learning

Well-known examples of supervised neural network are Perceptron [3], [138], ADALINE [119], Multi-layer ADALINE [120], MADLINE [121], and various

multi-layer feed-forward networks, etc. Due to the nature of supervised learning, the training patterns must be provided in terms of input/output patterns pairs, and the trainee will be told by the teacher about how to make proper adjustment to improve the performance. Some basic supervised learning rule is presented as follow:

Widrow-Hoff Learning [119]-[121]

The Widrow-Hoff learning is presented by Widrow and Hoff in 1960 and 1962 [119]. In 1960, the Adaptive Linear Combiner (ALC) is presented. The learning system of ALC is shown in Fig. 2.17(a). The learning rule of ALC is a Least Mean Square Error (LMSE) learning rule. The derivation weight Δw_{ij} is shown as

$$\Delta w_{ij} = c\,(d_i - y_i)\,u_j \tag{2.20}$$

where c is a constant when d_i is the desired output, and y_i is the real output of cell C(i) whereas u_j is the input of cell C(j). In 1962, the binary output is added to the ALC, and the new structure is called as ADAptive LINear Element (ADALINE) and its block diagram is shown in Fig. 2.17(b).

Delta Learning [103]

The Delta learning rule is proposed by Rumelhart, Hinton, and Williams in 1986 [122], and is evolved from the Widrow-Hoff learning. The learning system of Delta learning is shown in Fig. 2.18. The derivation of weight Δw_{ij} is proportional to the multiplication of the derivation of activation function and the difference of the desired and real outputs, and is shown as

$$\Delta w_{ij} = c\,(d_i - y_i)\,f'(x_i)\,u_j \tag{2.21}$$

where c, d_i, y_i, and u_j are the same as the above description whereas $f(x_i)$ is the activation function, and $f'(x_i)$ is its derivation. The limit of the Delta learning is that the activation function must be a differentiable function.

2.3.2 Unsupervised Learning

Unsupervised learning may be essential to classification in the absence of teacher's guidance. Since there will be no teacher, unsupervised learning exclusively on the characteristic inherently associated with the learning patterns. Various kinds of competitive learning network have been developed by Kohonen [51]-[52], [123], von der Malsburg [124]-[125], Fukushima [127], and Grossberg [4], [128]-[135], etc. The unsupervised learning systems have the following common features: 1) the training is based on competition. 2) Well-known examples of unsupervised neural network are Self-organization feature map [21], [51], [124], ART I-II [12]-[13].

Hebbian learning [2], [136]-[138]

The Hebbian learning rule is proposed by D. O. Hebb in 1949 [2]. The derivation of weight Δw_{ij} is proportional to the cross product of input and output, and can be written as

$$\Delta w_{ij} = c\, x_i\, u_j \tag{2.22}$$

where c is a constant, u_j is the input of cell C(j) and x_i is the state of cell C(i). The modified Hebbian learning is proposed by Zurada in 1992 [138]. The derivation of weight Δw_{ij} is proportional to the cross product of cell input and the other cell input. So the derivation Δw_{ij} can be written as

$$\Delta w_{ij} = c\, u_i\, u_j \tag{2.23}$$

and the learning system is shown in Fig. 2.19.

Winner-take-all learning

Only the weighting vector connected to the output node that has maximum response output will be adjusted. The learning system is shown in Fig. 2.20 [64], and the derivation of weight Δw_{ij} can be written as

$$\Delta w_{ij} = c\,(\,u_j - w_{ij}\,) \tag{2.24}$$

where w_{ij} is the weight of cell C(j) to the cell C(i), and the output y_i of the cell C(i) has the maximum output response.

The above four learning methods are the fundamental learning methods applied in neural network. Many learning methods are evolved from the above four learning methods. Sometimes, the neural network has more than one layer cells, so more than one learning methods are combined to be used in the multi-layer neural networks.

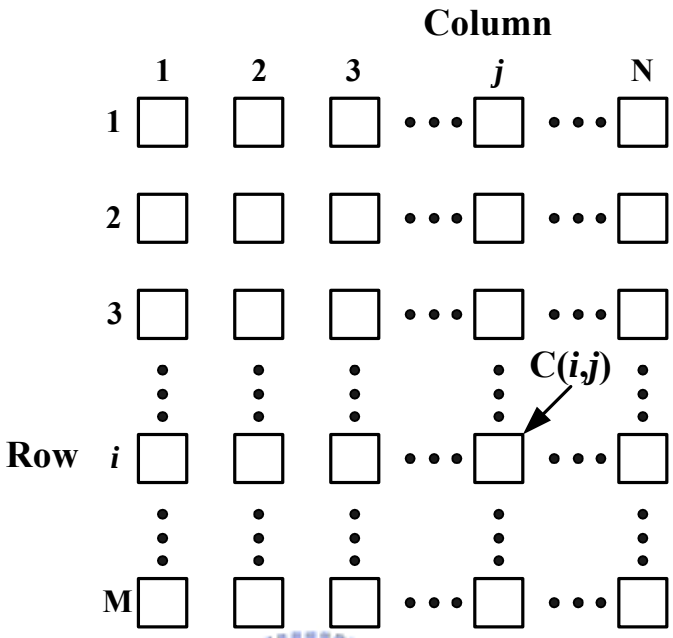


Fig. 2.1 The basic configuration of an MxN CNN array.

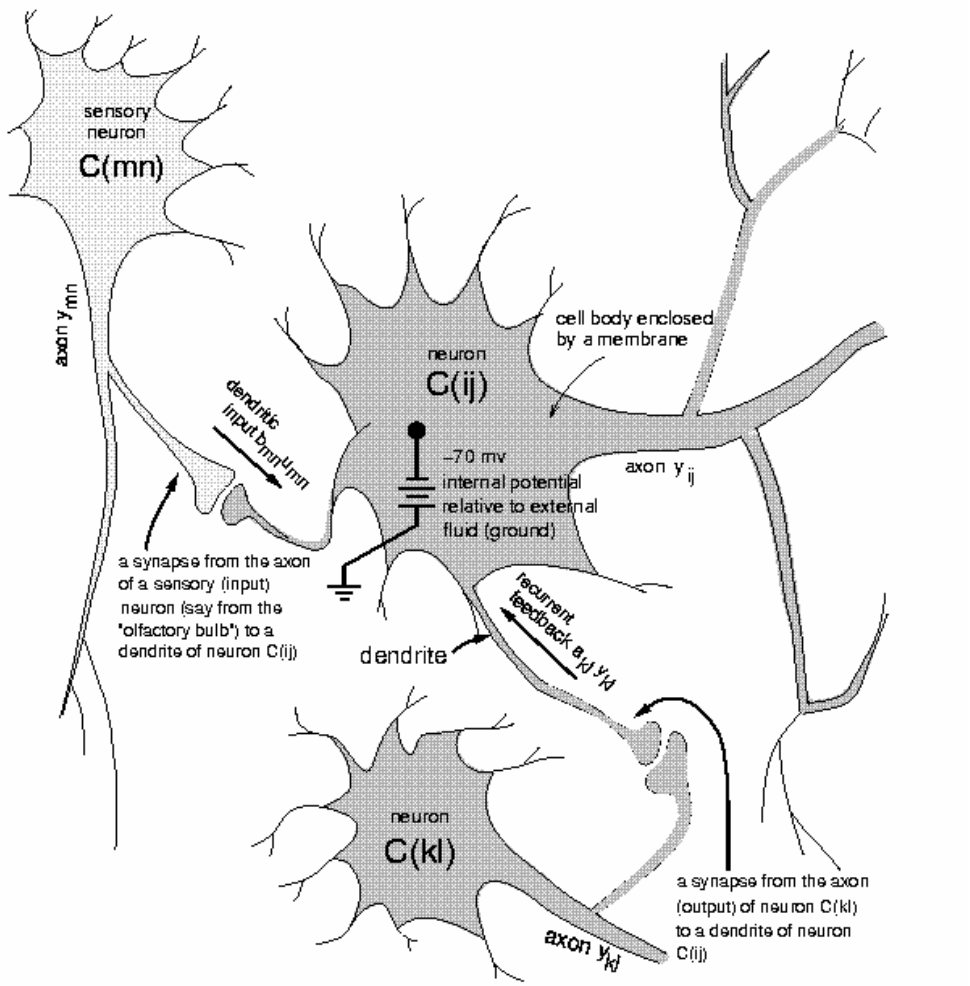


Fig. 2.2 A caricature of a typical cell C(i,j) receiving an input from a sensory cell on the left and a neighbor cell below through respective synapse.

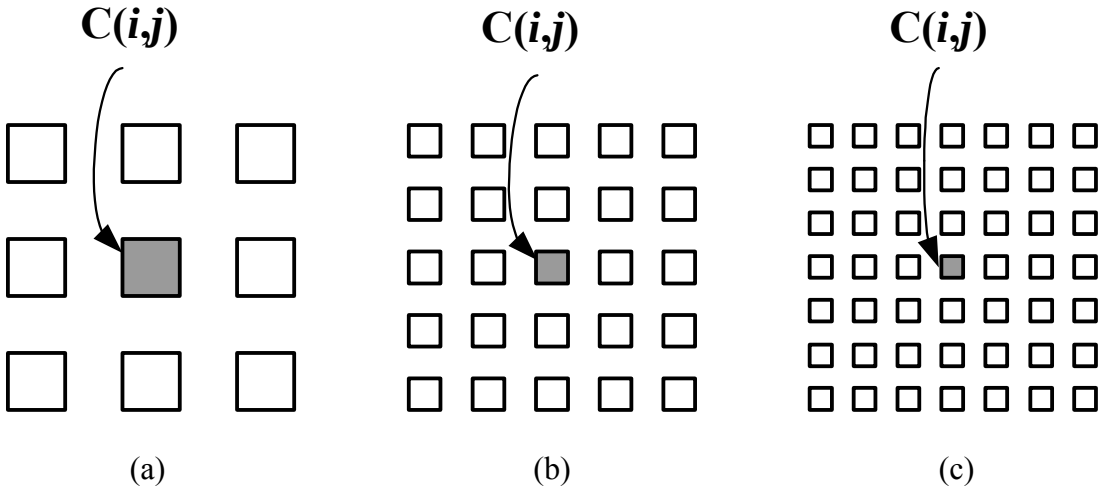


Fig. 2.3 Three examples of neighborhood systems of central cell C(i,j): (a) r = 1 neighborhood system $N_1(i,j)$; (b) r = 2 neighborhood system $N_2(i,j)$; (c) r = 3 neighborhood system $N_3(i,j)$.

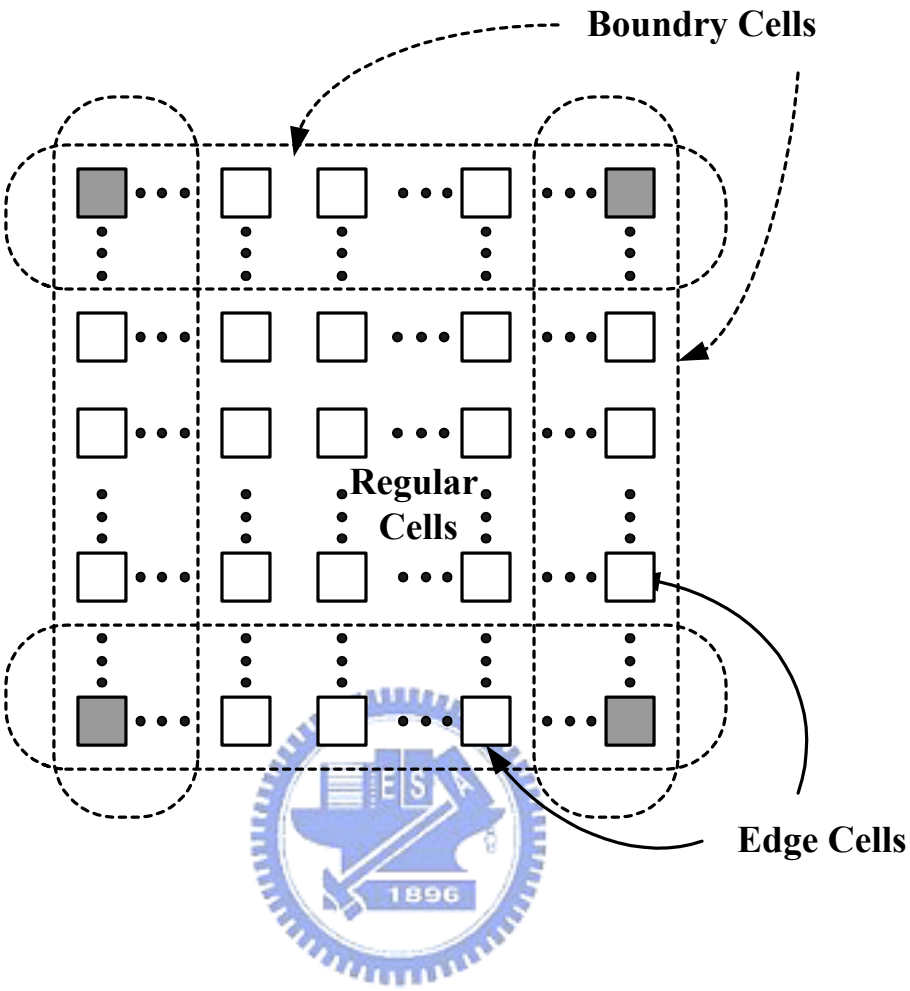


Fig. 2.4 Illustration of boundary cells, edge cells, corner cells, and regular cells in a CNN cell array.

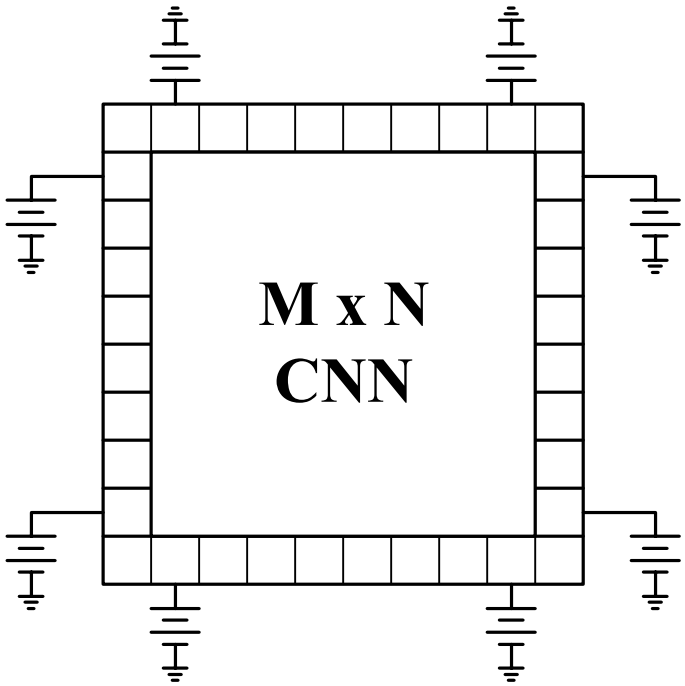


Fig. 2.5 The fixed (Dirichlet) boundary condition.

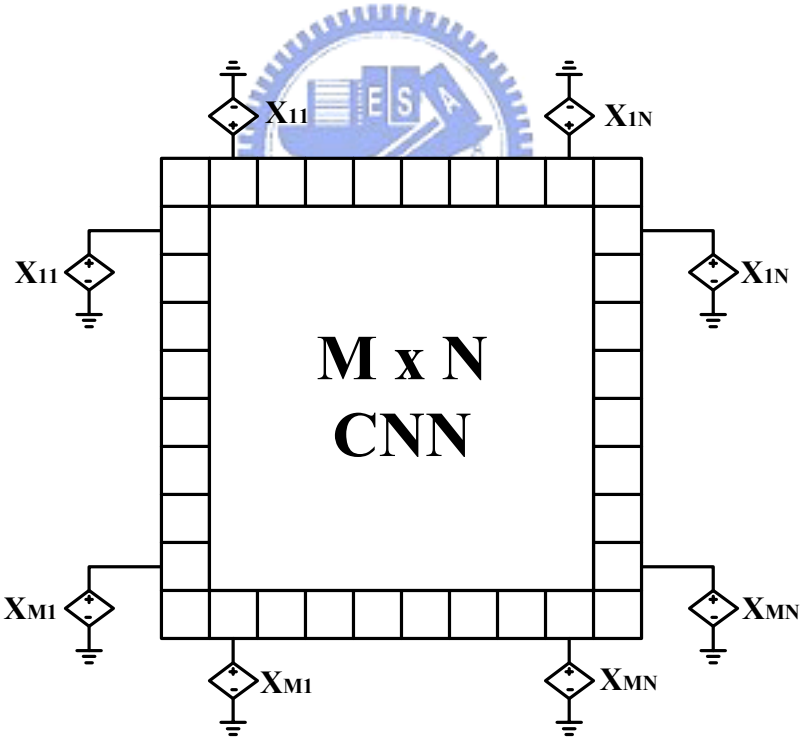


Fig. 2.6 The zero-flux (Neumann) boundary condition.

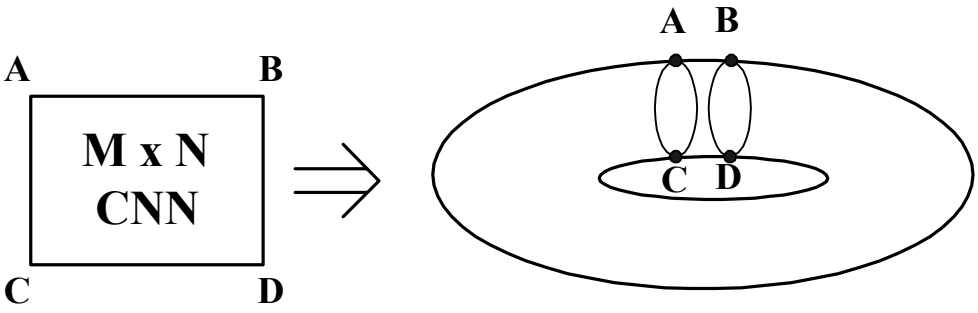


Fig. 2.7 The periodic (Toroidal) boundary condition.

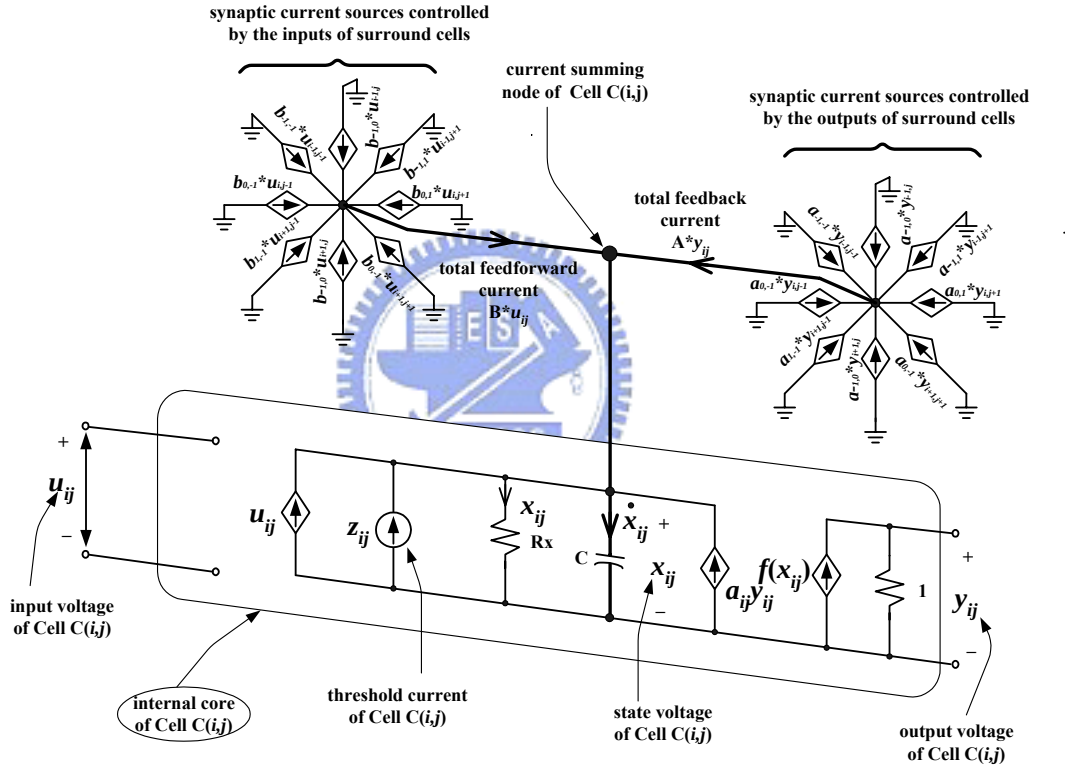
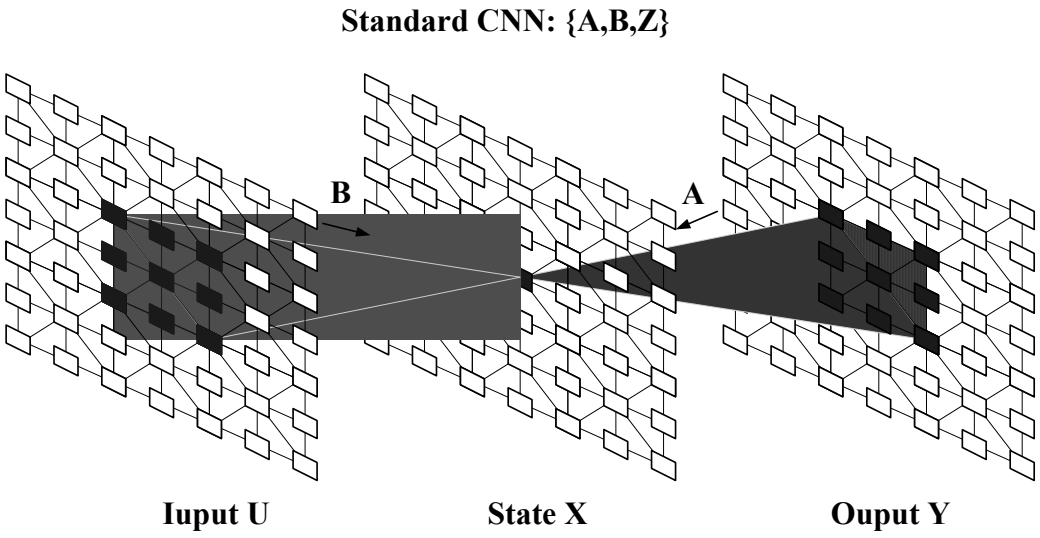
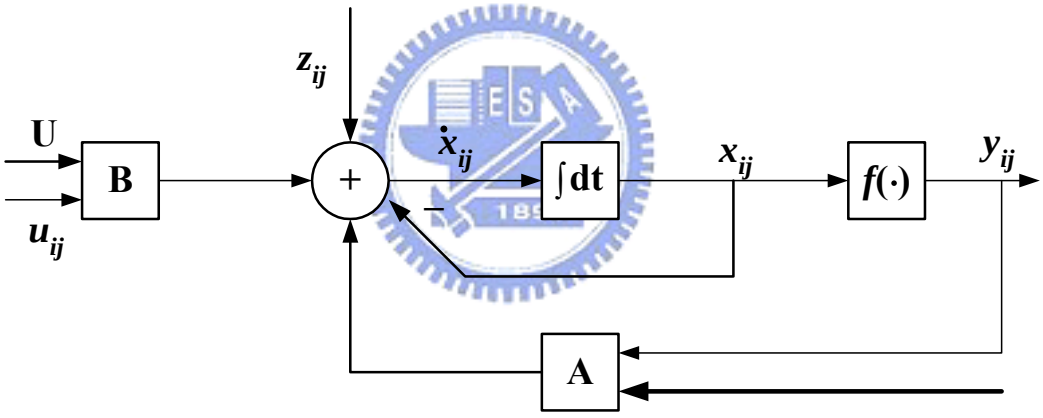


Fig. 2.8 The cell realization of a standard CNN cell C(i,j).

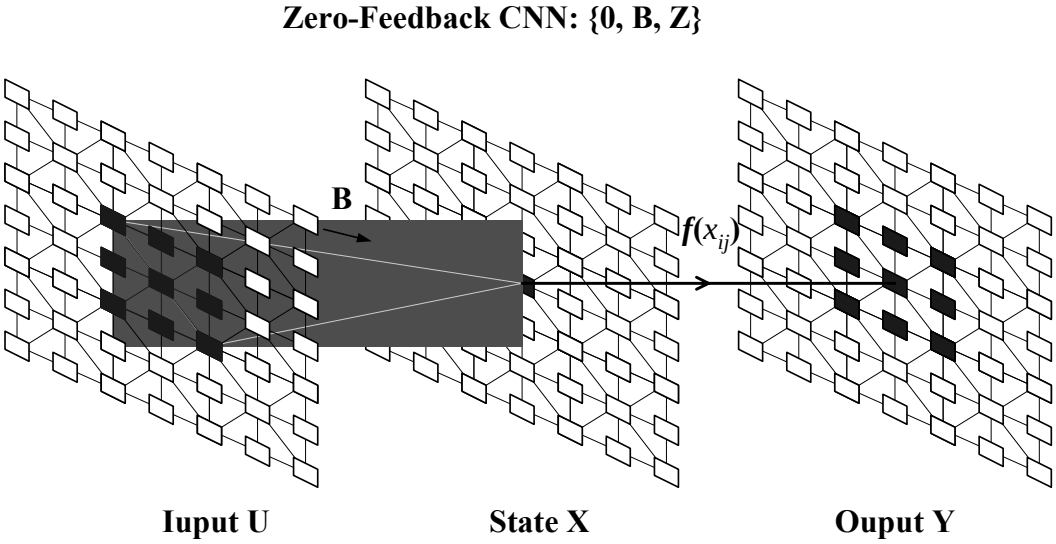


(a)

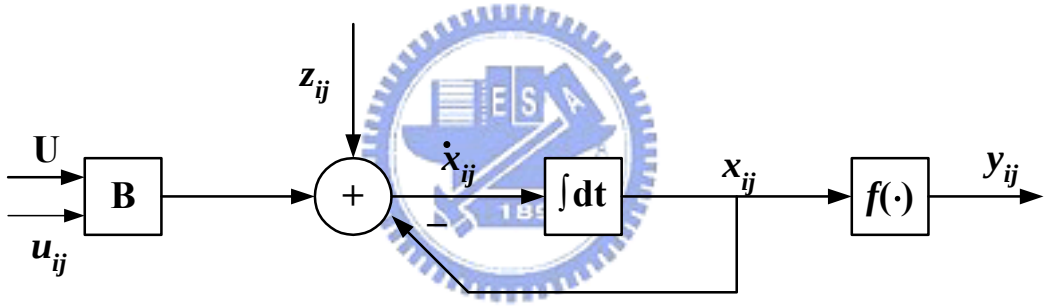


(b)

Fig. 2.9 (a) The signal flow structure of a standard CNN {**A**, **B**, **Z**} with a single-neighborhood $N_1(i,j)$; and (b) the system structure of a cell C(i,j).

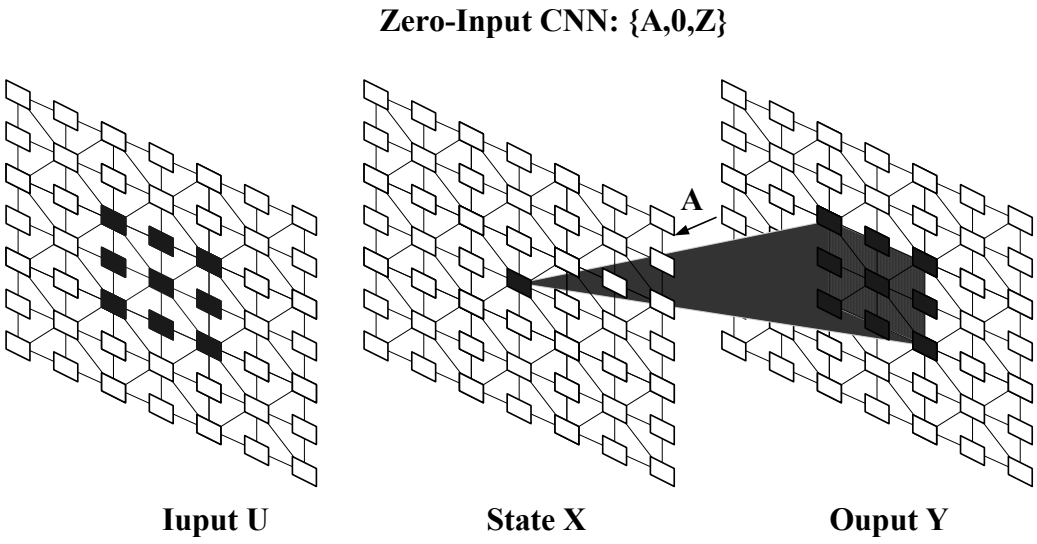


(a)

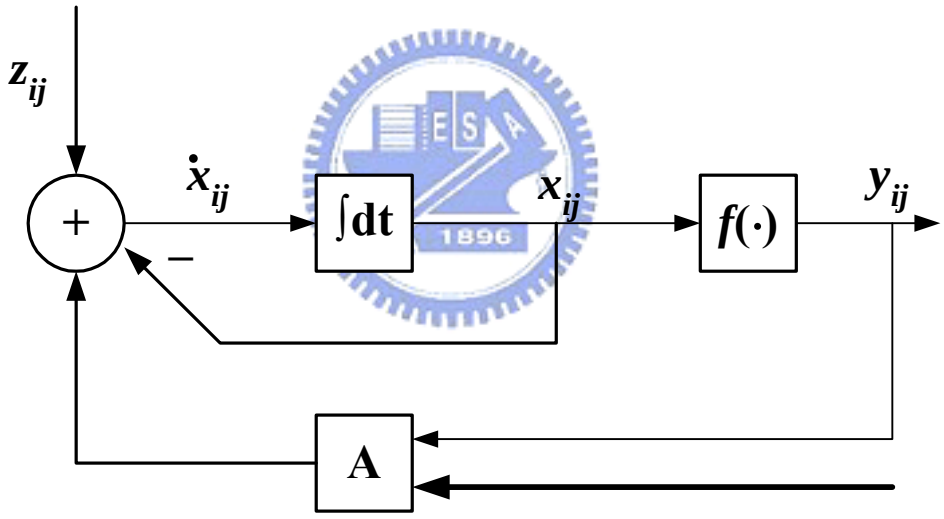


(b)

Fig. 2.10 (a) The signal flow structure of a zero-feedback (feed-forward) CNN with a single-neighborhood; and (b) the system structure of a cell C(i,j).



(a)



(b)

Fig. 2.11 (a) The signal flow structure of a zero-input (Autonomous) CNN with a single neighborhood; and (b) the system structure of a cell C(*i*,*j*).

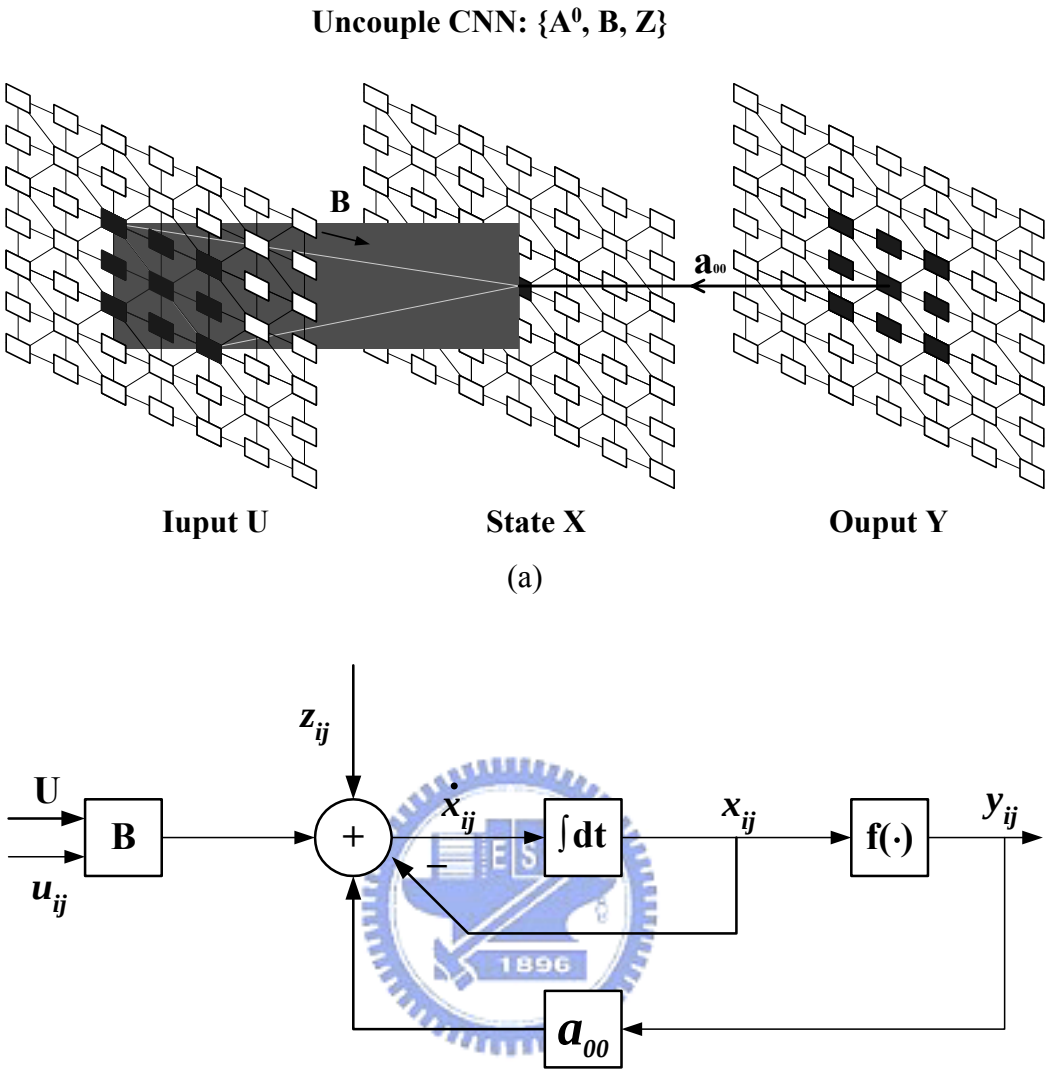


Fig. 2.12 (a) The signal flow structure of a uncouple CNN {$\mathbf{A^0}$, **B**, **Z**} with a single neighborhood; and (b) the system structure of a cell C(i,j).

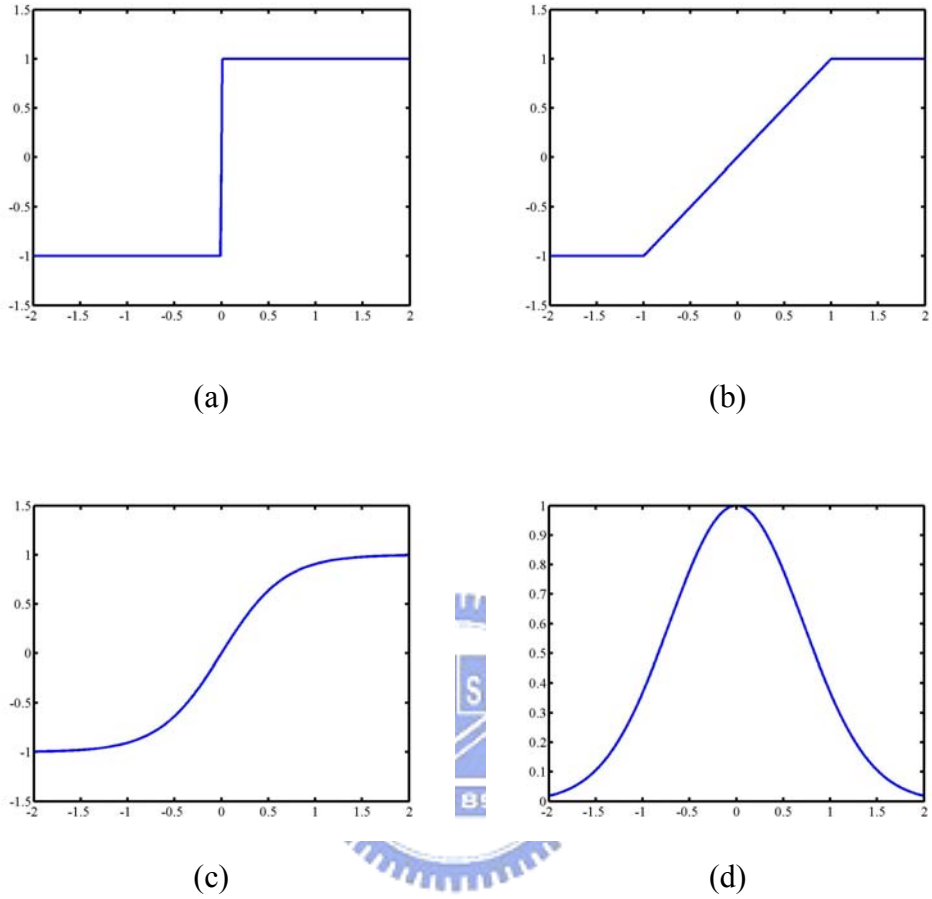


Fig. 2.13 The basic nonlinear activation functions of the cell. (a) Hard limiter (Step) function. (b) Ramp function. (c) Sigmoid function. (d) Gaussian function.

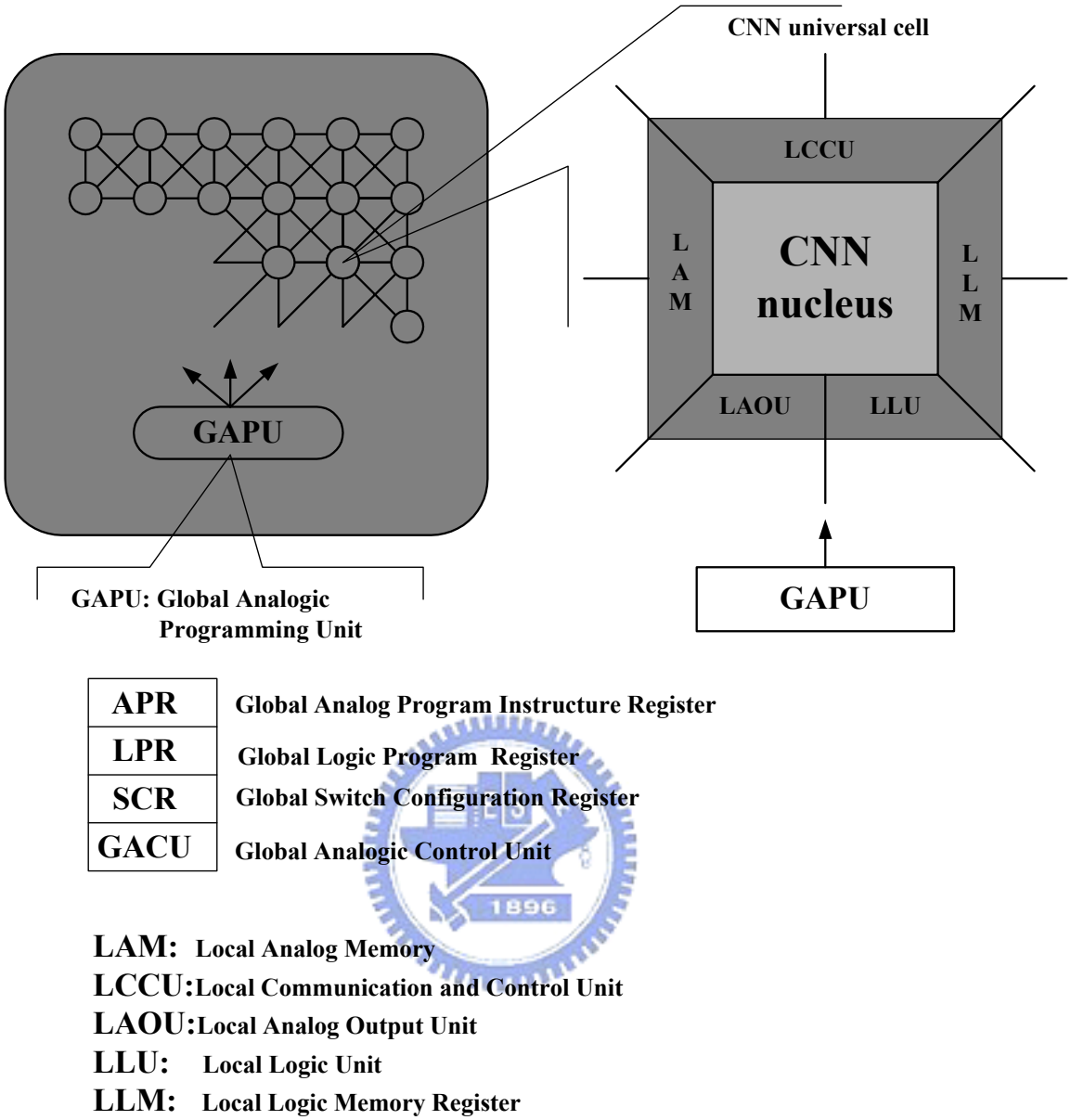


Fig. 2.14 The architecture of the CNN Universal Machine (CNN-UM).

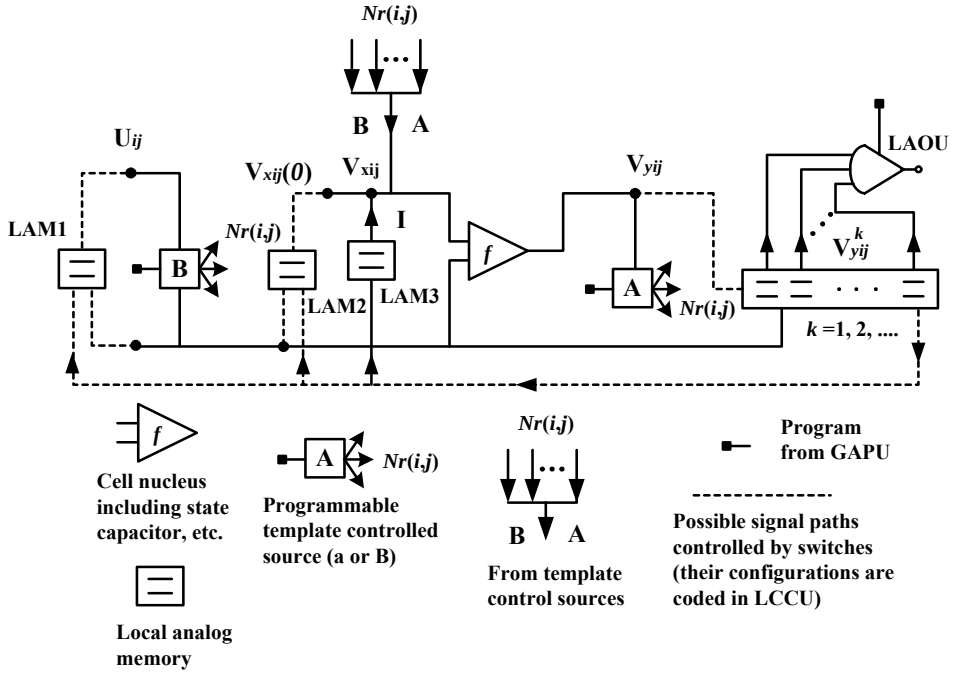
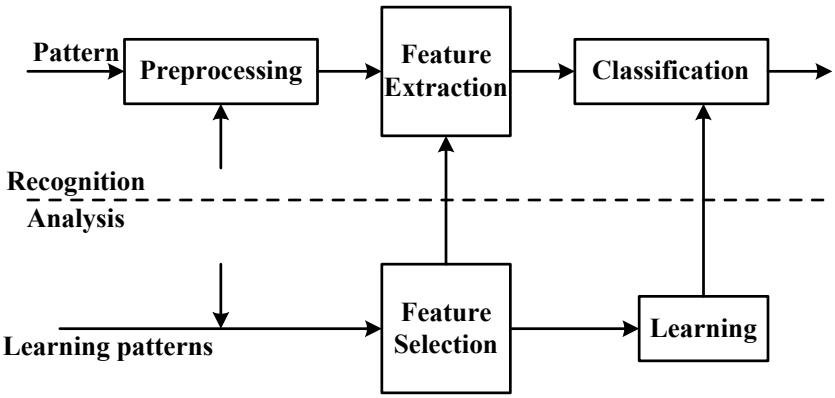
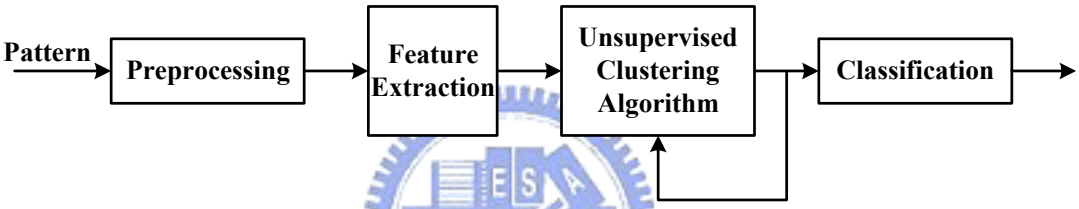


Fig. 2.15 The analog part of the analogic CNN universal cell with a symbolic analog cell unit.

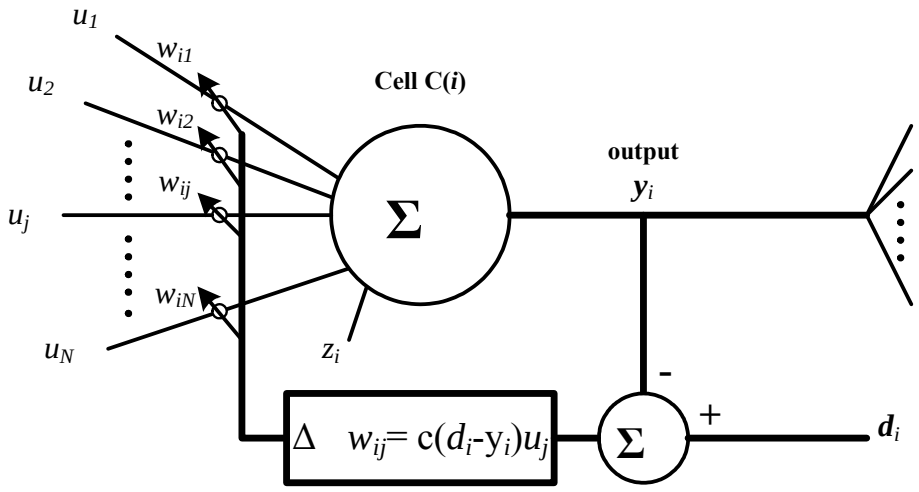


(a)



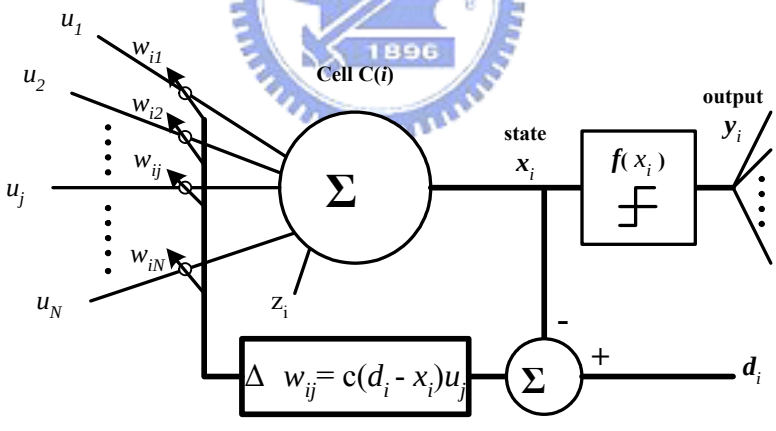
(b)

Fig. 2.16 The block diagrams of (a) the supervised and (b) the unsupervised pattern recognition system



w_{ij} : synapse weight
z_i : neuron threshold

(a)



w_{ij} : synapse weight
z_i : cell threshold
$f(x_i)$: activation function

(b)

Fig. 2.17 The learning systems of Widrow-Hoff learning in (a) Adaptive Linear Combiner (ALC) structure; (b) ADAptive LINear Element (ADALINE) structure.

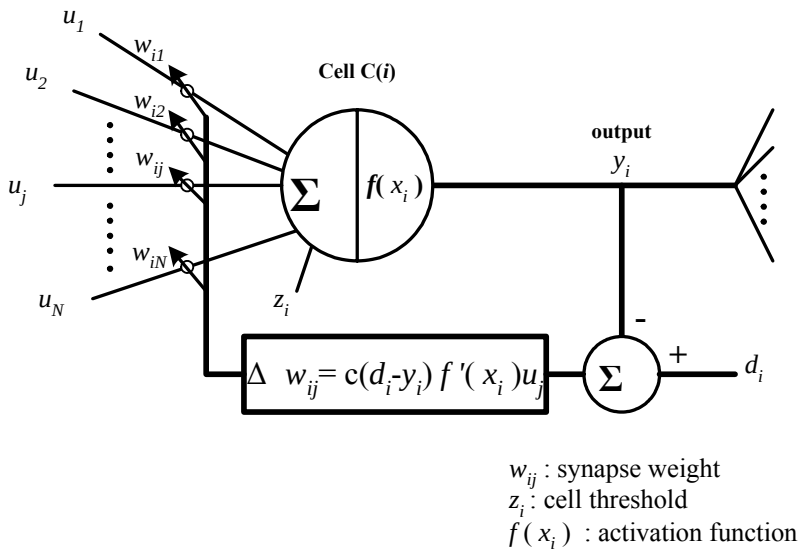
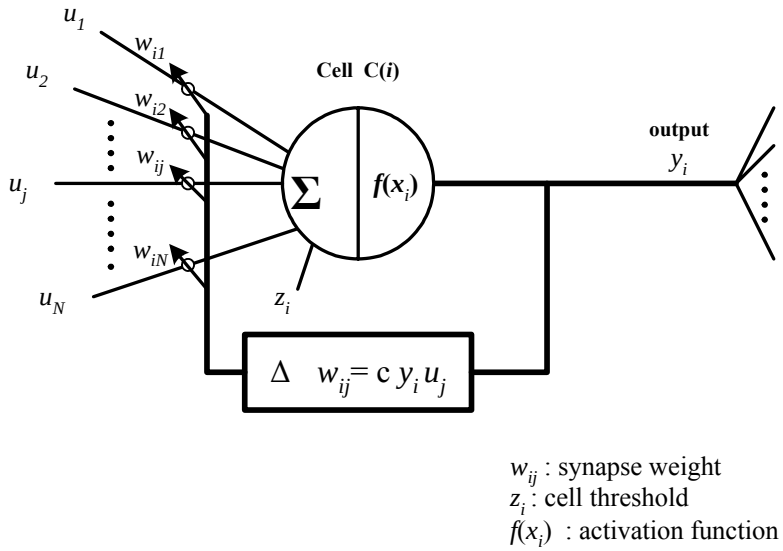
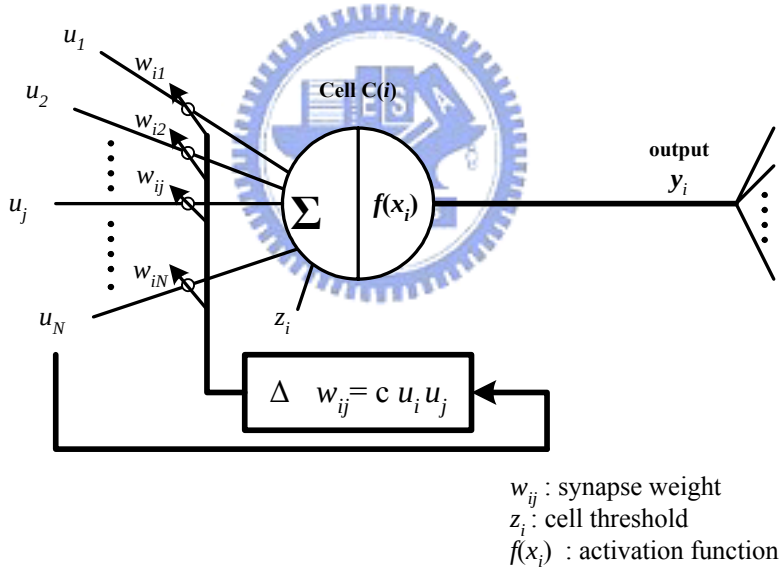


Fig. 2.18 The learning system of the Delta learning.



(a)



(b)

Fig. 2.19 The learning systems of (a) the original Hebbian learning, and (b) the modified Hebbian learning.

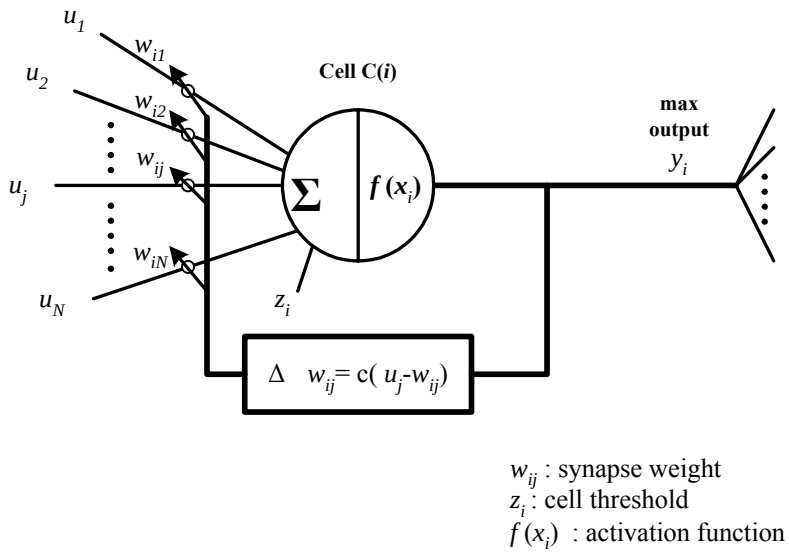


Fig. 2.20 The learning system of Winner-Take-All learning.