

# Chapter 4

## Cellular Neural Networks for Seismic Pattern Recognition

### 4.1 Introduction

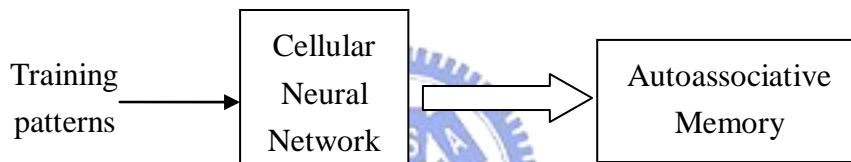
Seismic pattern recognition can help us to analyze seismic data and interpret seismic data. We use associative memories to store seismic patterns and recognize noisy patterns.

The neural network models which are used for associative memories mostly operate by utilizing a large amount of memory and the process of information distribution. Some neural network models can be used for associative memories, such as linear associative memory [1], temporal associative memory [2], correlation matrix memory [3], Hopfield memory [4], etc. Content address memory that Hopfield proposed is most famous among them. This thesis uses cellular neural networks to design associative memories. Many researchers had proposed the methods of using cellular neural networks to design associative memories: Liu and Michel have already proposed a design method for sparsely interconnected neural networks and applied it to cellular neural networks [5]. Their method is a synthesis procedure based on singular value decomposition and it is a modification of the eigenstructure method [6] which is well known as an effective design technique for fully connected neural networks. Seiler, Schuler, and Nossek developed an algorithm to design a space-varying cellular neural network with prescribed stable and unstable output patterns while maximizing its robustness with respect to changes of its parameters [7]. The algorithm consists of formulating and solving a set of linear inequalities using linear programming. The method proposed by Grassi [8]-[9] is different from others because it uses the cellular neural networks that have a unique equilibrium point which is globally asymptotically stable and input patterns are fed into such cellular neural networks as bias vectors. Park, etc. [10] have proposed one kind synthesis procedure to design space-varying cellular neural networks. This procedure is based on generalized eigenvalue minimization (GEVM) [11]-[13]. It produces a cellular neural network associative memory that its connection weighting matrix is symmetric and diagonal elements of the connection weighting matrix are all 1. Liu and Lu showed that the perceptron training algorithm can be applied to design space-varying cellular neural networks as well as fully connected feedback neural networks [14].

Chan and Zak proposed “designer” neural network for the synthesis of associative memories based on a class of discrete cellular neural networks [15]. Perfetti presented a local learning algorithm which can efficiently be implemented on chip by exploiting the parallel analog computation of cellular neural networks [16].

The network model we used for associative memory is discrete time cellular neural network (DT-CNN). We adopt the synthesis procedure that is proposed by Grassi [8]-[9]. This synthesis procedure design DT-CNN to behave as an associative memory. Each memory pattern corresponds to a unique globally asymptotically stable equilibrium point of the network. Experimental results show the satisfying performance of this synthesis procedure. We use this synthesis procedure to design the motion equation of a cellular neural network to behave as an associative memory, and then use the associative memory to recognize patterns. So the whole process of pattern recognition will be divided into two parts and discussed respectively, as Fig. 4.1 shows.

**Training :**



**Recognition :**

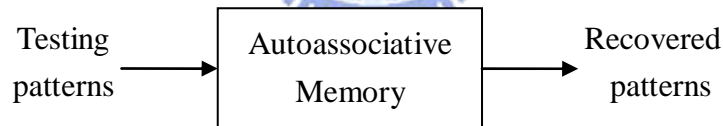


Fig. 4.1 The process of cellular neural networks for seismic pattern recognition.

## 4.2 Associative Memories

There are two kinds of associative memories. They are autoassociative memory and heteroassociative memory. Suppose we have  $M$  pairs of vector  $\{(X^1, Y^1), (X^2, Y^2), \dots, (X^M, Y^M)\}$ , where  $X^i \in \mathfrak{R}^n$  and  $Y^i \in \mathfrak{R}^p$ . In the autoassociative memory, it is assumed that  $X^i = Y^i$  and the network implements a mapping  $\Phi$  of  $\mathfrak{R}^n$  to  $\mathfrak{R}^n$  such that  $\Phi(X^i) = X^i$ . If some arbitrary pattern  $X$  is closer to  $X^i$  than to any other  $X^j, j = 1, 2, \dots, M, j \neq i$ , then  $\Phi(X) = X^i$ ; that is, the network will produce the stored pattern  $X^i$  when the key pattern  $X$  is presented as input. In the

heteroassociative memory, the network implements a mapping  $\Phi$  of  $\mathcal{R}^n$  to  $\mathcal{R}^p$  such that  $\Phi(X^i) = Y^i$ , and if some arbitrary pattern  $X$  is closer to  $X^i$  than to any other  $X^j, j = 1, 2, \dots, M, j \neq i$ , then  $\Phi(X) = Y^i$ . In the above, “closer” means with respect to some proper distance measure, for example, the Hamming distance. Fig. 4.2 shows a general block diagram of an associative memory performing an associative mapping of an input vector  $X$  into an output  $Y$ . The system maps vector  $X$  to vector  $Y$  where the input space is  $\mathcal{R}^n$  and the output space is  $\mathcal{R}^p$ , by performing the transformation

$$Y = M[X]$$

The operator  $M[\cdot]$  denotes a general nonlinear matrix-type operator. For different memory model, the operator has different meanings.

An efficient associative memory can store a large set of memory patterns and recall each pattern when a key containing a portion of the memory’s information is presented as an input. And if sufficient amount of the content of the memory is provided, then an associative memory may be able to correct erroneous data.

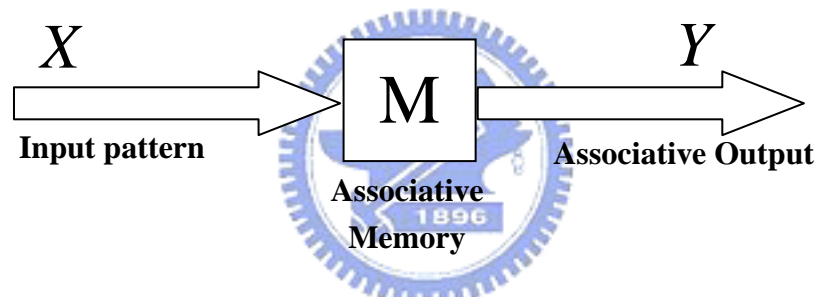


Fig. 4.2 Associative memory.

### 4.3 DT-CNNs for Associative Memories

We use the design method proposed by G. Grassi [9] to design DT-CNNs for associative memories. The detail of the design method and the stability results are summarized in the following.

Consider a DT-CNN with a two-dimensional  $M \times N$  cell array. We can renumber all cells from 1 to  $n, n = M \times N$ , then the following motion equation (4-1) and output equation (4-2) can be rewritten as vector form, as Equation (4-3) and (4-4) show:

$$x_{ij}(h+1) = \sum_{C_{kl} \in N_{ij}} A(i, j; k, l) y_{kl}(h) + \sum_{C_{kl} \in N_{ij}} B(i, j; k, l) u_{kl}(h) + I_{ij} \quad (4-1)$$

$$y_{ij}(h+1) = f(x_{ij}(h+1)) = \begin{cases} +1 & \text{for } x_{ij}(h+1) \geq 0 \\ -1 & \text{for } x_{ij}(h+1) < 0 \end{cases} \quad (4-2)$$

$$1 \leq i \leq M; 1 \leq j \leq N$$

$$\mathbf{x}(h+1) = \mathbf{A} \mathbf{y}(h) + \mathbf{B} \mathbf{u} + \mathbf{e} \quad (4-3)$$

$$\mathbf{y}(h) = \mathbf{f}(\mathbf{x}(h)) \quad (4-4)$$

In Equation (4-3) and (4-4):

$\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]^T \in \mathfrak{R}^{n \times 1}$  is a vector compose of state value of every cell;

$\mathbf{y} = [y_1 \ y_2 \ \dots \ y_n]^T \in \mathfrak{R}^{n \times 1}$  is a vector compose of output of every cell;

$\mathbf{u} = [u_1 \ u_2 \ \dots \ u_n]^T \in \mathfrak{R}^{n \times 1}$  is a vector compose of input of every cell;

$\mathbf{e} = [I_1 \ I_2 \ \dots \ I_n]^T \in \mathfrak{R}^{n \times 1}$  is a vector compose of extra input of every cell;

$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \ddots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \in \mathfrak{R}^{n \times n}$  is a matrix compose of all  $A(i, j; k, l)$ ;

$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \ddots & & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{bmatrix} \in \mathfrak{R}^{n \times n}$  is a matrix compose of all  $B(i, j; k, l)$ .

$\mathbf{f} = [f(x_1) \ f(x_2) \ \dots \ f(x_n)]^T \in \mathfrak{R}^{n \times 1}$  is a vector contains output functions.

### 4.3.1 Design of Feedback Template

#### Linear Neighboring

A one-dimensional space-invariant template is used as the feedback template. The corresponding network structure ( $r=1$  and  $n=16$ ) is shown in Fig. 4.3 [9].

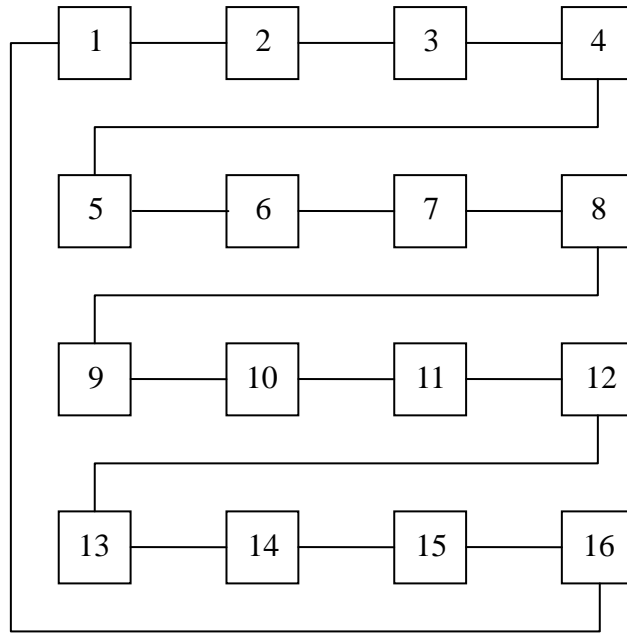
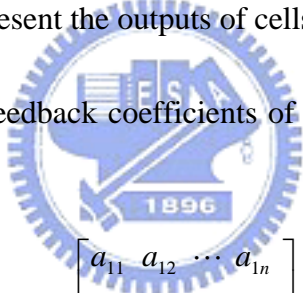


Fig. 4.3 The connection relation of cells for designing associative memories ( $r = 1$  and  $n = 16$ ). The links represent the outputs of cells.

If there are  $n$  cells, then the feedback coefficients of the cells can be expressed with the following matrix  $\mathbf{A}$ .



$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \quad (4-5)$$

$a_{11}$  represents the self-feedback coefficient of 1st cell,  $a_{12}$  represents the feedback coefficient of the next cell (the 2nd cell) of 1st cell in clockwise order,  $a_{21}$  represents the feedback coefficient of the next cell (the 1st cell) of 2nd cell in counterclockwise order. Since the feedback template is a one-dimensional space-invariant template, so  $a_{11} = a_{22} = \cdots = a_{nn} = \alpha_1$  represent the self-feedback coefficient of each cell,  $a_{12} = a_{23} = \cdots = a_{(n-1)n} = a_{n1} = \alpha_2$  represent the feedback coefficient of the next cell in clockwise order,  $a_{1n} = a_{21} = a_{32} = \cdots = a_{n(n-1)} = \alpha_n$  represent the feedback coefficient of the next cell in counterclockwise order,  $a_{13} = a_{24} = \cdots = a_{(n-2)n} = a_{(n-1)1} = a_{n2} = \alpha_3$  represent the feedback coefficient of the next two cell in clockwise order, and so on. So matrix  $\mathbf{A}$  can be expressed as

$$\mathbf{A} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_n & \alpha_1 & \cdots & \alpha_{n-1} \\ \vdots & & \ddots & \vdots \\ \alpha_2 & \alpha_3 & \cdots & \alpha_1 \end{bmatrix}$$

Therefore  $\mathbf{A}$  is a circulant matrix. The following one-dimensional space-invariant template is considered:

$$[ a(-r) \dots a(-1) \quad a(0) \quad a(1) \dots a(r) ]$$

$r$  is neighborhood radius,  $a(0)$  is the self-feedback,  $a(1)$  is the feedback of next cell in the clockwise order,  $a(-1)$  is the feedback of next cell in the counterclockwise order, and so on. So according to Equation (4-5), we rearrange the template elements as the following row vector.

$$[ a(0) \quad a(1) \dots a(r) \quad 0 \dots 0 \quad a(-r) \dots a(-1) ] \quad (4-6)$$

The Equation (4-6) is the first row of matrix  $\mathbf{A}$ . We arrange the last element of the first row of matrix  $\mathbf{A}$  in the first position of the second row of matrix  $\mathbf{A}$ , it is regarded as the first element of the second row of matrix  $\mathbf{A}$ , other elements of the first row of matrix  $\mathbf{A}$  cycle right shift one position, they form the second to the last element of the second row of matrix  $\mathbf{A}$ . The same, we take the previous row to cycle right shift once, the new sequence is the next row of matrix  $\mathbf{A}$ , then we can define matrix  $\mathbf{A}$ .

$$\mathbf{A} = \begin{bmatrix} a(0) & a(1) & \cdots & a(r) & 0 & \cdots & 0 & a(-r) & \cdots & a(-1) \\ a(-1) & a(0) & \cdots & a(r) & 0 & \cdots & 0 & a(-r) & \cdots & a(-2) \\ \vdots & & & & \cdots & & & & & \vdots \\ \vdots & & & & \cdots & & & & & \vdots \\ \vdots & & & & \cdots & & & & & \vdots \\ a(1) & a(2) & \cdots & \cdots & \cdots & \cdots & \cdots & a(-1) & a(0) \end{bmatrix} \quad (4-7)$$

It only needs to design the first row of matrix  $\mathbf{A}$  when we design matrix  $\mathbf{A}$  as a circulant matrix. Each next row is the previous row which is cycle right shifted once. The number of 0s of Equation (4-6) is decided by radius  $r$  and  $n$ .  $\mathbf{A} \in \mathfrak{R}^{n \times n}$ , namely each row of matrix  $\mathbf{A}$  has  $n$  elements, and there are  $n$  rows in  $\mathbf{A}$ . If  $n = 9$ , then there are nine elements in each row. When  $r = 1$ , the one dimensional template is sorted according to the Equation (4-6), as the following Equation (4-8) shows.

$$[ a(0) \quad a(1) \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad a(-1) ] \quad (4-8)$$

In the Equation (4-8), there are six 0s in the middle of the template, add other three elements, there are nine elements in a row.

### 4.3.2 Stability Results

If a dynamic system has a unique equilibrium point which attracts every trajectory in state space, then it is called globally asymptotically stable. A criterion for the global asymptotic stability of the equilibrium point of DT-CNNs with circulant matrices has been introduced [17]. The criterion is summarized in the following. DT-CNNs described by (4-3) and (4-4), with matrix  $\mathbf{A}$  given by (4-7), are globally asymptotically stable, if and only if

$$|S(2\pi q/n)| < 1, q = 0, 1, 2, \dots, n-1 \quad (4-9)$$

where

$$S(2\pi q/n) = \sum_{h=-r}^r a(h)e^{-j2\pi hq/n} \quad (4-10)$$

The stability criterion (4-9) can be easily satisfied by choosing small values for the elements of the one-dimensional space-invariant template. In particular, the larger the network dimension  $n$  is, the smaller the values of the elements will be (see (4-10)). On the other hand, the feedback values cannot be zero, since the stability properties considered herein require that (4-3) be a dynamical system. These guidelines can help the designer in fixing the values of the feedback parameters. Namely, the lower bound is zero, whereas the upper bound is related to the network dimension.

### 4.3.3 Design of DT-CNNs for Associative Memories

The method is designing the motion equation of a cellular neural network to behave as an associative memory. Given  $m$  bipolar (value for +1 or -1) training patterns as input vectors  $\mathbf{u}^i$ ,  $i = 1, 2, \dots, m$ , for each  $\mathbf{u}^i$ , there is only one equilibrium point  $\mathbf{x}^i$  satisfying motion equation:

$$\begin{cases} \mathbf{x}^1 = \mathbf{A}\mathbf{y}^1 + \mathbf{B}\mathbf{u}^1 + \mathbf{e} \\ \mathbf{x}^2 = \mathbf{A}\mathbf{y}^2 + \mathbf{B}\mathbf{u}^2 + \mathbf{e} \\ \vdots \\ \mathbf{x}^m = \mathbf{A}\mathbf{y}^m + \mathbf{B}\mathbf{u}^m + \mathbf{e} \end{cases} \quad (4-11)$$

Design cellular neural networks to behave as associative memories, mainly set up  $\mathbf{A}$  and calculate  $\mathbf{B}$  and  $\mathbf{e}$  according to training patterns.

In order to express Equation (4-11) into a matrix form, we define the following matrices first:

$$\begin{aligned} \mathbf{X} = [\mathbf{x}^1 \ \mathbf{x}^2 \ \cdots \ \mathbf{x}^m] &= \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^m \\ x_2^1 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \ddots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^m \end{bmatrix} \in \mathfrak{R}^{n \times m} \\ \mathbf{Y} = [\mathbf{y}^1 \ \mathbf{y}^2 \ \cdots \ \mathbf{y}^m] &= \begin{bmatrix} y_1^1 & y_1^2 & \cdots & y_1^m \\ y_2^1 & y_2^2 & \cdots & y_2^m \\ \vdots & \vdots & \ddots & \vdots \\ y_n^1 & y_n^2 & \cdots & y_n^m \end{bmatrix} \in \mathfrak{R}^{n \times m} \\ \mathbf{A}_y = \mathbf{A}\mathbf{Y} = [\mathbf{A}\mathbf{y}^1 \ \mathbf{A}\mathbf{y}^2 \ \cdots \ \mathbf{A}\mathbf{y}^m] &= [\mathbf{d}^1 \ \mathbf{d}^2 \ \cdots \ \mathbf{d}^m] \in \mathfrak{R}^{n \times m} \\ \mathbf{d}^i = [d_1^i \ d_2^i \ \cdots \ d_n^i]^T &\in \mathfrak{R}^{n \times 1}, i = 1, \dots, m \\ \mathbf{U} = [\mathbf{u}^1 \ \mathbf{u}^2 \ \cdots \ \mathbf{u}^m] &= \begin{bmatrix} u_1^1 & u_1^2 & \cdots & u_1^m \\ u_2^1 & u_2^2 & \cdots & u_2^m \\ \vdots & \vdots & \ddots & \vdots \\ u_n^1 & u_n^2 & \cdots & u_n^m \end{bmatrix} \in \mathfrak{R}^{n \times m} \\ \mathbf{J} = [\mathbf{e} \ \mathbf{e} \ \cdots \ \mathbf{e}] &= \begin{bmatrix} I_1 & I_1 & \cdots & I_1 \\ I_2 & I_2 & \cdots & I_2 \\ \vdots & \vdots & \ddots & \vdots \\ I_n & I_n & \cdots & I_n \end{bmatrix} \in \mathfrak{R}^{n \times m} \end{aligned}$$

The Equation (4-11) can be expressed in the matrix form:

$$\mathbf{X} = \mathbf{A}\mathbf{Y} + \mathbf{B}\mathbf{U} + \mathbf{J} \quad (4-12)$$

$$\mathbf{B}\mathbf{U} + \mathbf{J} = \mathbf{X} - \mathbf{A}\mathbf{Y}$$

$$\mathbf{B}\mathbf{U} + \mathbf{J} = \mathbf{X} - \mathbf{A}_y \quad (4-13)$$

$\mathbf{U} = [\mathbf{u}^1 \ \mathbf{u}^2 \ \cdots \ \mathbf{u}^m]$ ,  $\mathbf{u}^i$  is a training pattern,  $i = 1, 2, \dots, m$

$\mathbf{U}$  has been already known. Because  $\mathbf{Y}$  is the desired output, so  $\mathbf{Y} = \mathbf{U}$  has been already known. Under global asymptotic stability condition, choose a



sequence  $\{a(-r) \cdots a(-1) a(0) a(1) \cdots a(r)\}$ , which satisfies the criterion (4-9). And design  $\mathbf{A}$  as a circulant matrix, so  $\mathbf{A}$  has been already known too. We can know from output function, if  $y$  is  $+1$ , then  $x > 1$ ; if  $y$  is  $-1$ , then  $x < -1$ .  $\mathbf{U}$  is a bipolar vector, so  $\mathbf{Y}$  is a bipolar vector too, namely all elements in  $\mathbf{Y}$  are  $+1$  or  $-1$ , so the elements of the state vector  $\mathbf{X}$  corresponded to  $\mathbf{Y}$  are all greater than  $+1$  or less than  $-1$ , so we can establish  $\mathbf{X} = \alpha \mathbf{Y} = \alpha \mathbf{U}$ ,  $\alpha > 1$ . So  $\mathbf{U}$ ,  $\mathbf{Y}$ ,  $\mathbf{A}$  and  $\mathbf{X}$  have been already known, we want to calculate  $\mathbf{B}$  and  $\mathbf{J}$ .

Define the following matrices:

$$\mathbf{R} = [\mathbf{U}^T \ \mathbf{h}] \in \mathfrak{R}^{m \times (n+1)}$$

$$= \begin{bmatrix} u_1^1 & u_2^1 & \cdots & u_n^1 & 1 \\ u_1^2 & u_2^2 & \cdots & u_n^2 & 1 \\ \vdots & & \ddots & & \vdots \\ u_1^m & u_2^m & \cdots & u_n^m & 1 \end{bmatrix}$$

$$= \begin{bmatrix} u_1^1 & u_2^1 & \cdots & u_n^1 & 1 \\ u_1^2 & u_2^2 & \cdots & u_n^2 & 1 \\ \vdots & & \ddots & & \vdots \\ u_1^m & u_2^m & \cdots & u_n^m & 1 \end{bmatrix}$$

$$\mathbf{h} = [1 \ 1 \ \cdots \ 1]^T \in \mathfrak{R}^{m \times 1}$$

$$\mathbf{X}_j = [x_j^1 \ x_j^2 \ \cdots \ x_j^m] \in \mathfrak{R}^{1 \times m} \text{ is the } j\text{th row of matrix } \mathbf{X}$$

$$\mathbf{A}_{y,j} = [d_j^1 \ d_j^2 \ \cdots \ d_j^m] \in \mathfrak{R}^{1 \times m}$$

$$[\mathbf{B} | \mathbf{e}] = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} & I_1 \\ b_{21} & b_{22} & \cdots & b_{2n} & I_2 \\ \vdots & & \ddots & & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} & I_n \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_n \end{bmatrix}$$

$$\mathbf{w}_j = [b_{j1} \ b_{j2} \ \cdots \ b_{jn} \ I_j] \in \mathfrak{R}^{1 \times (n+1)}$$

$$j = 1, 2, \dots, n$$

The Equation (4-13) can be rewritten as Equation (4-14):

$$\mathbf{B}\mathbf{U} + \mathbf{J} = \mathbf{X} - \mathbf{A}_y \quad (4-13)$$

$$\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \ddots & & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix} \begin{bmatrix} u_1^1 & u_1^2 & \cdots & u_1^m \\ u_2^1 & u_2^2 & \cdots & u_2^m \\ \vdots & \ddots & & \vdots \\ u_n^1 & u_n^2 & \cdots & u_n^m \end{bmatrix} + \begin{bmatrix} I_1 & I_1 & \cdots & I_1 \\ I_2 & I_2 & \cdots & I_2 \\ \vdots & \ddots & & \vdots \\ I_n & I_n & \cdots & I_n \end{bmatrix} \\ = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^m \\ x_2^1 & x_2^2 & \cdots & x_2^m \\ \vdots & \ddots & & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^m \end{bmatrix} - \begin{bmatrix} d_1^1 & d_1^2 & \cdots & d_1^m \\ d_2^1 & d_2^2 & \cdots & d_2^m \\ \vdots & \ddots & & \vdots \\ d_n^1 & d_n^2 & \cdots & d_n^m \end{bmatrix}$$

In the view of the  $j$ th row

$$\begin{bmatrix} b_{j1} & b_{j2} & \cdots & b_{jn} \end{bmatrix} \begin{bmatrix} u_1^1 & u_1^2 & \cdots & u_1^m \\ u_2^1 & u_2^2 & \cdots & u_2^m \\ \vdots & \ddots & & \vdots \\ u_n^1 & u_n^2 & \cdots & u_n^m \end{bmatrix} + [I_j \ I_j \ \cdots \ I_j] \\ = [x_j^1 \ x_j^2 \ \cdots \ x_j^m] - [d_j^1 \ d_j^2 \ \cdots \ d_j^m]$$

$$\mathbf{R}\mathbf{w}_j^T = \mathbf{X}_j^T - \mathbf{A}_{y,j}^T \quad (4-14)$$

$$\begin{bmatrix} u_1^1 & u_2^1 & \cdots & u_n^1 & 1 \\ u_1^2 & u_2^2 & \cdots & u_n^2 & 1 \\ \vdots & \ddots & & \vdots & \vdots \\ u_1^m & u_2^m & \cdots & u_n^m & 1 \end{bmatrix} \begin{bmatrix} b_{j1} \\ b_{j2} \\ \vdots \\ b_{jn} \\ I_j \end{bmatrix} = \begin{bmatrix} x_j^1 \\ x_j^2 \\ \vdots \\ x_j^m \end{bmatrix} - \begin{bmatrix} d_j^1 \\ d_j^2 \\ \vdots \\ d_j^m \end{bmatrix}, j = 1, 2, \dots, n$$

The Equation (4-14) is the transpose of the  $j$ th row of Equation (4-13), so we can rewrite Equation (4-13) as Equation (4-14).

Because each cell is only influenced by its neighboring cells, so matrix  $\mathbf{B}$  is a sparse matrix, and elements in  $\mathbf{w}_j$  are mostly 0. We remove 0 elements of  $\mathbf{w}_j$ , then we can get  $\tilde{\mathbf{w}}_j$ . And we remove the corresponding columns of  $\mathbf{R}$ , then we can get  $\tilde{\mathbf{R}}_j$ , and  $\tilde{\mathbf{R}}_j \tilde{\mathbf{w}}_j^T = \mathbf{R}\mathbf{w}_j^T$ .

$$\tilde{\mathbf{R}}_j \tilde{\mathbf{w}}_j^T = \mathbf{X}_j^T - \mathbf{A}_{y,j}^T \quad (4-15)$$

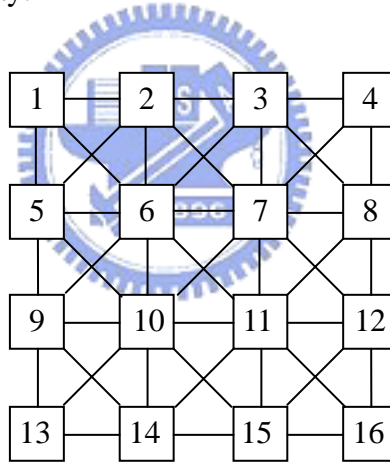
$$\tilde{\mathbf{w}}_j^T = \tilde{\mathbf{R}}_j^+ (\mathbf{X}_j^T - \mathbf{A}_{y,j}^T), j = 1, 2, \dots, n \quad (4-16)$$

$\tilde{\mathbf{R}}_j$  is got from  $\mathbf{R}$  according to the connection relation of the input of the  $j$ th cell and inputs of other cells. We express the connection relation of the inputs of cells by matrix  $\mathbf{S}$ , so  $\tilde{\mathbf{R}}_j$  can be got by taking out partly vectors of  $\mathbf{R}$  according to the  $j$ th row of  $\mathbf{S}$ .  $\tilde{\mathbf{R}}_j^+$  is the pseudoinverse of  $\tilde{\mathbf{R}}_j$ .  $\tilde{\mathbf{R}}_j \in \mathcal{R}^{m \times h_j}$ ,  $\tilde{\mathbf{w}}_j \in \mathcal{R}^{1 \times h_j}$ ,  $h_j = \left( \sum_{i=1}^n S_{ji} \right) + 1$ .

Matrix  $\mathbf{S}$  is the matrix represents the connection relation of cells' inputs.  $\mathbf{S} \in \mathcal{R}^{n \times n}$ , if the  $i$ th cell's input and  $j$ th cell's input have connection relation, then  $\mathbf{S}_{ij} = 1$ , on the other hand, if the  $i$ th cell's input and  $j$ th cell's input have no connection relation,  $\mathbf{S}_{ij} = 0$ .

$$\mathbf{S}_{ij} = \begin{cases} 1, & \text{if the } i\text{th cell's input and } j\text{th cell's input have connection relation} \\ 0, & \text{if the } i\text{th cell's input and } j\text{th cell's input have no connection relation} \end{cases}$$

For example, a  $4 \times 4$  cell array:

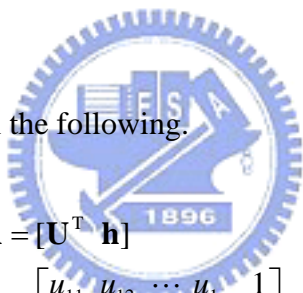


and radius  $r = 1$ , then  $\mathbf{S}$  is written as the following:

$$\mathbf{S} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

We explain  $\tilde{\mathbf{R}}_j \tilde{\mathbf{w}}_j^T = \mathbf{R} \mathbf{w}_j^T$  in the following.

*Proof:*



$$\begin{aligned} \mathbf{R} &= [\mathbf{U}^T \ \mathbf{h}] \\ &= \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} & 1 \\ u_{21} & u_{22} & \cdots & u_{2n} & 1 \\ \vdots & & \ddots & & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mn} & 1 \end{bmatrix} \\ \mathbf{w}_j &= [b_{j1} \ b_{j2} \ \cdots \ b_{jn} \ I_j] \\ \mathbf{R} \mathbf{w}_j^T &= \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} & 1 \\ u_{21} & u_{22} & \cdots & u_{2n} & 1 \\ \vdots & & \ddots & & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mn} & 1 \end{bmatrix} \cdot \begin{bmatrix} b_{j1} \\ b_{j2} \\ \vdots \\ b_{jn} \\ I_j \end{bmatrix} \end{aligned}$$

The  $j$ th row  $[s_{j1} \ s_{j2} \ \cdots \ s_{jn}]$  of matrix  $\mathbf{S}$  represents that whether the  $j$ th cell and other cells (include the  $j$ th cell) are neighbors,  $s_{ji} = 1$  represents that the  $j$ th cell and the  $i$ th cell are neighbors, on the contrary,  $s_{ji} = 0$  represents that the  $j$ th cell and the  $i$ th cell are not neighbors. The  $j$ th cell and the  $i$ th cell are not neighbors, it represents that the  $j$ th cell can not be influenced by the input of the  $i$ th cell, so  $b_{ji} = 0$ , and when

$\mathbf{R}$  multiplies  $\mathbf{w}_j^T$ ,  $u_{1i}$ ,  $u_{2i}$ , ...,  $u_{mi}$  are all multiplied by 0, so we can remove the  $i$ th column  $[u_{1i} \ u_{2i} \ \cdots \ u_{mi}]^T$  of matrix  $\mathbf{R}$ , then it forms  $\tilde{\mathbf{R}}_j$ , and remove  $b_{ji}$  of  $\mathbf{w}_j^T$ , then it forms  $\tilde{\mathbf{w}}_j^T$ , and  $\tilde{\mathbf{R}}_j \tilde{\mathbf{w}}_j^T = \mathbf{R} \mathbf{w}_j^T$ .

In Equation (4-14), we can get  $\mathbf{w}_j^T$  by the following equation.

$$\mathbf{w}_j^T = \mathbf{R}^+ (\mathbf{X}_j^T - \mathbf{A}_{y,j}^T)$$

$\mathbf{R}^+$  may not be unique, so  $\mathbf{w}_j^T$  may not be unique.  $\mathbf{B}$  may not be unique. Then  $\mathbf{B}$  may not accord with the interconnecting structure of network inputs. So we must use a matrix  $\mathbf{S}$  to represent the interconnecting structure of network inputs and use the above skill to calculate matrix  $\mathbf{B}$ . We summarize the algorithm of using cellular neural networks to design associative memories in the following.

**Algorithm 4.1:** Design the DT-CNN to behave as an associative memory

**Input:**  $m$  bipolar patterns  $\mathbf{u}^i$ ,  $i = 1, \dots, m$

**Output:**  $\mathbf{w}_j = [b_{j1} \ b_{j2} \ \cdots \ b_{jn} \ I_j]$ ,  $j = 1, \dots, n$ , namely  $\mathbf{B}$  and  $\mathbf{e}$

Start:

Step 1: Calculate matrix  $\mathbf{U}$  from known  $\mathbf{u}^i$

$$\mathbf{U} = [\mathbf{u}^1 \ \mathbf{u}^2 \ \cdots \ \mathbf{u}^m]$$

Step 2: Establish matrix  $\mathbf{Y} =$  matrix  $\mathbf{U}$ .

$$\mathbf{Y} = \mathbf{U} = [\mathbf{u}^1 \ \mathbf{u}^2 \ \cdots \ \mathbf{u}^m]$$

Step 3: Establish matrix  $\mathbf{S}$ .

$$\mathbf{S}_{ij} = \begin{cases} 1, & \text{if the } i\text{th cell's input and } j\text{th cell's input have connection relation} \\ 0, & \text{if the } i\text{th cell's input and } j\text{th cell's input have no connection relation} \end{cases}$$

Namely  $\mathbf{S} \in \mathcal{R}^{n \times n}$  is a square matrix. If the  $i$ th cell's input and  $j$ th cell's input have connection relation, then  $\mathbf{S}_{ij} = 1$ . On the other hand, if they have no connection relation, then  $\mathbf{S}_{ij} = 0$ .

Step 4: Design matrix  $\mathbf{A}$  as the circulant matrix which satisfies globally asymptotically stable condition.

Step 5: Set the value of  $\alpha$  ( $\alpha > 1$ ), and calculate  $\mathbf{X} = \alpha \mathbf{Y}$ .

Step 6: Calculate  $\mathbf{A}_y = \mathbf{A}\mathbf{Y}$ .

Step 7: for ( $j=1$  to  $n$ ) do:

Calculate  $\mathbf{X}_j$  by  $\mathbf{X}$ .

Calculate  $\mathbf{A}_{y,j}$  by  $\mathbf{A}_y$ .

Calculate  $\mathbf{R}$ ,  $\mathbf{R} = [\mathbf{U}^T \mathbf{h}]$ .

Establish matrix  $\tilde{\mathbf{R}}_j$  from matrix  $\mathbf{S}$  and matrix  $\mathbf{R}$ .

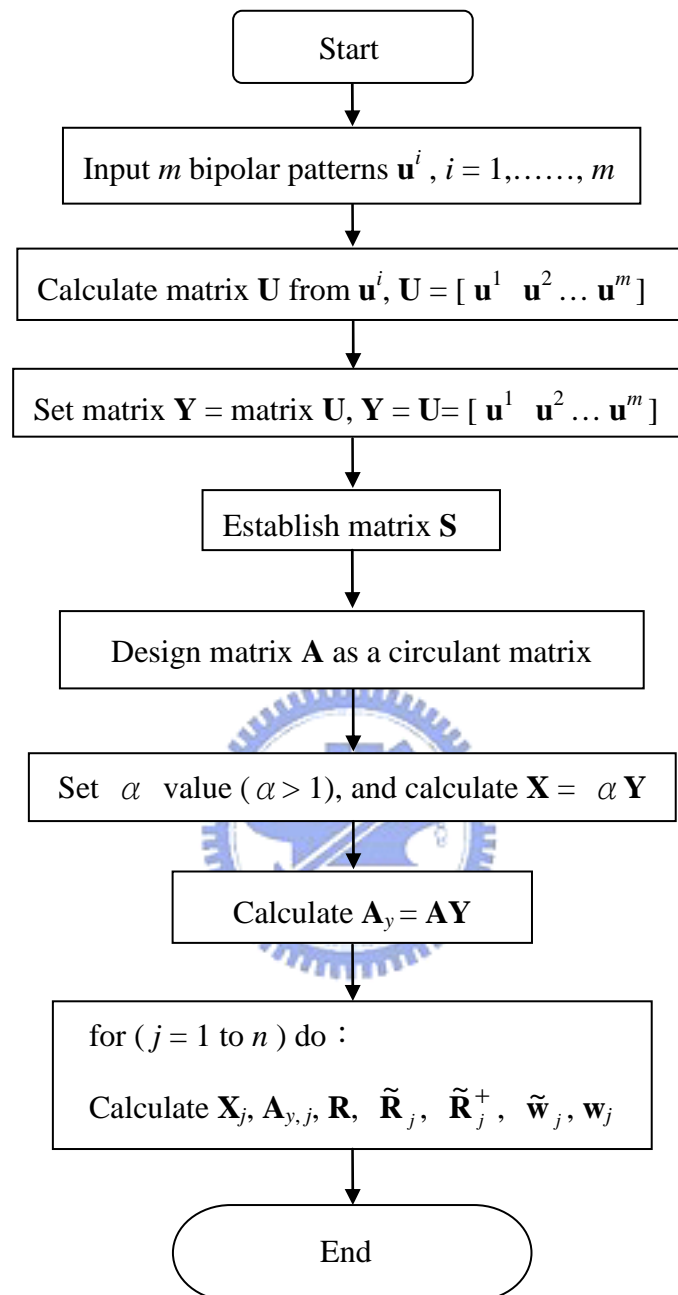
Calculate pseudoinverse matrix  $\tilde{\mathbf{R}}_j^+$  of  $\tilde{\mathbf{R}}_j$ .

Calculate  $\tilde{\mathbf{w}}_j^T$ ,  $\tilde{\mathbf{w}}_j^T = \tilde{\mathbf{R}}_j^+ (\mathbf{X}_j^T - \mathbf{A}_{y,j}^T)$ .

Recover  $\mathbf{w}_j$  from  $\tilde{\mathbf{w}}_j^T$ .

End

**The flowchart of Algorithm 4.1:**



#### 4.3.4 Associative Memories Design Principle

$$\mathbf{X} = [\mathbf{x}^1 \ \mathbf{x}^2 \ \cdots \ \mathbf{x}^m]$$

$$\mathbf{x}^i = [x_1^i \ x_2^i \ \cdots \ x_n^i]^T \in \mathfrak{R}^{n \times 1}, \ i = 1, 2, \dots, m$$

$$\mathbf{Y} = [\mathbf{y}^1 \ \mathbf{y}^2 \ \cdots \ \mathbf{y}^m]$$

$$\mathbf{y}^i = [y_1^i \ y_2^i \ \cdots \ y_n^i]^T \in \mathfrak{R}^{n \times 1}, \ i = 1, 2, \dots, m$$

$$\mathbf{U} = [\mathbf{u}^1 \ \mathbf{u}^2 \ \cdots \ \mathbf{u}^m]$$

$$\mathbf{u}^i = [u_1^i \ u_2^i \ \cdots \ u_n^i]^T \in \mathfrak{R}^{n \times 1}, \ i = 1, 2, \dots, m$$

$$\mathbf{J} = [\mathbf{e} \ \mathbf{e} \ \cdots \ \mathbf{e}]$$

$$\mathbf{e} = [I_1 \ I_2 \ \cdots \ I_n]^T \in \mathfrak{R}^{n \times 1}$$

$$\mathbf{X} = \mathbf{A}\mathbf{Y} + \mathbf{B}\mathbf{U} + \mathbf{J}$$

$$\begin{aligned} [\mathbf{x}^1 \ \mathbf{x}^2 \ \cdots \ \mathbf{x}^m] &= \mathbf{A}[\mathbf{y}^1 \ \mathbf{y}^2 \ \cdots \ \mathbf{y}^m] + \mathbf{B}[\mathbf{u}^1 \ \mathbf{u}^2 \ \cdots \ \mathbf{u}^m] + [\mathbf{e} \ \mathbf{e} \ \cdots \ \mathbf{e}] \\ &= [\mathbf{A}\mathbf{y}^1 \ \mathbf{A}\mathbf{y}^2 \ \cdots \ \mathbf{A}\mathbf{y}^m] + [\mathbf{B}\mathbf{u}^1 \ \mathbf{B}\mathbf{u}^2 \ \cdots \ \mathbf{B}\mathbf{u}^m] + [\mathbf{e} \ \mathbf{e} \ \cdots \ \mathbf{e}] \end{aligned}$$

$$\mathbf{x}^i = \mathbf{A}\mathbf{y}^i + \mathbf{B}\mathbf{u}^i + \mathbf{e}, \ i = 1, 2, \dots, m$$

$\mathbf{u}^i, i = 1, 2, \dots, m$  are training patterns, namely memorized patterns. Because we want to design autoassociative memories, so outputs are memorized patterns. Output  $\mathbf{y}^i =$  input  $\mathbf{u}^i$ , and  $\mathbf{u}^i$  are bipolar patterns, so  $\mathbf{y}^i$  are bipolar patterns too, namely the elements of  $\mathbf{y}^i$  are +1 or -1. From output function we can know that if output is +1, then the state is greater than +1; if output is -1, then the state is less than -1. So we set up  $\mathbf{X} = \alpha \mathbf{Y}$ ,  $\alpha > 1$ . And then we design matrix  $\mathbf{A}$  as a circulant matrix which satisfies the globally asymptotically stable condition, then calculate  $\mathbf{B}$  and  $\mathbf{e}$ .

Then  $\mathbf{G}(\mathbf{x}) = \mathbf{A}\mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u} + \mathbf{e}$  is a contraction and there is a unique solution to the nonlinear equation  $\mathbf{G}(\mathbf{x}) = \mathbf{x}$ , namely there is a unique solution to  $\mathbf{x} = \mathbf{A}\mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u} + \mathbf{e}$ , where  $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_n]^T$ . So if  $\mathbf{u} = \mathbf{u}^i$ , then  $\mathbf{x}^i$  is the unique solution. No matter what initial  $\mathbf{x}$  is,  $\mathbf{x}$  must converge to  $\mathbf{x}^i$  finally. There is a unique solution for each element  $x_i$  of  $\mathbf{x}$  too. We can write the motion equation of the  $i$ th cell as the following equation:

$$\dot{x}_i = \frac{dx_i(t)}{dt} = g(x_i) + w_i = h(x_i; w_i)$$

where

$$g(x_i) = -x_i + a_i f(x_i) = \begin{cases} -x_i + a_i, & x_i \geq 1 \\ (a_i - 1)x_i, & |x_i| < 1 \\ -x_i - a_i, & x_i \leq -1 \end{cases}$$



$$w_i = I + \sum_{C_j \in N_i, j \neq i} a_j f(x_j) + \sum_{C_j \in N_i} b_j u_j$$

The dynamic routes of  $x_i$  are shown in Fig. 4.4.

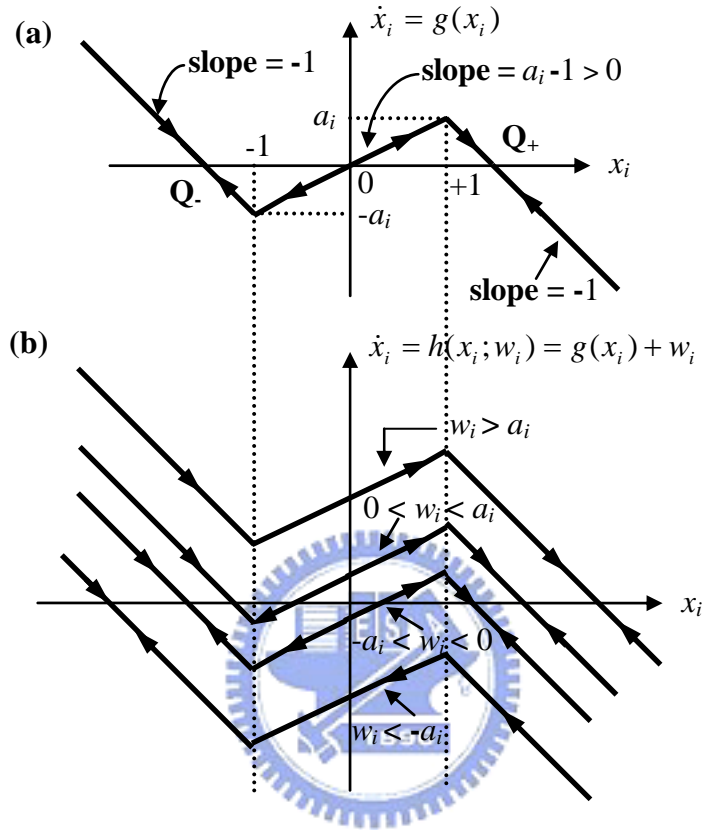


Fig. 4.4 The dynamic routes of  $x_i$ .

Because there is a unique solution for each element  $x_i$  of  $\mathbf{x}$ , so in Fig. 4.4, the second and third dynamic routes of (b) cannot be appeared. Only the first and fourth dynamic routes can be appeared. The first dynamic route is the case of  $w_i > a_i$ ; the fourth dynamic route is the case of  $w_i < -a_i$ . The elements of the one dimensional space-invariant template we designed for matrix  $\mathbf{A}$  are very small, namely  $a_i$  are very small, so  $w_i$  are greater than  $a_i$  or less than  $-a_i$  ( maybe only  $I$  has already been greater than  $a_i$  or less than  $-a_i$  ). So only the first and fourth dynamic routes can be appeared. On the first and fourth dynamic routes, no matter what is the initial value of each  $x_i$ , each  $x_i$  will converge to a unique equilibrium point. So if  $\mathbf{u} = \mathbf{u}^j$ , then each element  $x_i$  of  $\mathbf{x}$  converges to a unique equilibrium point  $x_i^j$ . Namely  $\mathbf{x}$  converges to  $\mathbf{x}^j$ . Therefore designing cellular neural networks to work as associative memories is designing each training pattern to behave as an equilibrium point of the

network.

We consider the situation that input is a noisy pattern  $\mathbf{q}$  after the network is trained,  $\mathbf{q} = [q_1 \ q_2 \ \cdots \ q_n]$ . We suppose that the noisy pattern  $\mathbf{q}$  is closer to training pattern  $\mathbf{u}^k$  than to any other  $\mathbf{u}^j, j = 1, 2, \dots, m, j \neq k$ . We compare the situations that input is  $\mathbf{u}^k$  and input is  $\mathbf{q}$ . The elements of  $\mathbf{q}$  are mostly equal to the corresponding elements of  $\mathbf{u}^k$ , only some elements of  $\mathbf{q}$  are not equal to the corresponding elements of  $\mathbf{u}^k$ . And in the equation of  $w_i$ , only  $u_j$  of input =  $\mathbf{q}$  is possibly different to  $u_j$  of input =  $\mathbf{u}^k$ , other parameters of input =  $\mathbf{q}$  are all equal to the corresponding parameters of input =  $\mathbf{u}^k$ . So  $w_i$  of input =  $\mathbf{q}$  are mostly equal to  $w_i$  of input =  $\mathbf{u}^k$ . Several  $w_i$  of input =  $\mathbf{q}$  are not equal to the corresponding  $w_i$  of input =  $\mathbf{u}^k$  due to some  $u_j$  of the equation of  $w_i$  when input is  $\mathbf{q}$  are not equal to the corresponding  $u_j$  of the equation of  $w_i$  when input is  $\mathbf{u}^k$ . Because the different  $u_j$  are not many, so the sum  $\sum_{C_j \in N_i} b_j u_j$  when input is  $\mathbf{q}$  is similar to the sum  $\sum_{C_j \in N_i} b_j u_j$  when input is  $\mathbf{u}^k$ , namely the difference of the sum  $\sum_{C_j \in N_i} b_j u_j$  when input is  $\mathbf{q}$  and the sum  $\sum_{C_j \in N_i} b_j u_j$  when input is  $\mathbf{u}^k$  is small, therefore the dynamic route of input =  $\mathbf{q}$  is similar to the dynamic route of input =  $\mathbf{u}^k$ . So the equilibrium point of input =  $\mathbf{q}$  is similar to the equilibrium point of input =  $\mathbf{u}^k$ . If the equilibrium point of input =  $\mathbf{u}^k$  is greater than +1, then the equilibrium point of input =  $\mathbf{q}$  is greater than +1 too; if the equilibrium point of input =  $\mathbf{u}^k$  is less than -1, then the equilibrium point of input =  $\mathbf{q}$  is less than -1 too. So the output when input is  $\mathbf{q}$  is equal to the output when input is  $\mathbf{u}^k$ . Therefore we can recover the pattern  $\mathbf{u}^k$  which is noisy pattern  $\mathbf{q}$  most similar to when we input  $\mathbf{q}$  to the network. But if the number of different elements of input  $\mathbf{q}$  correspond to its most similar pattern  $\mathbf{u}^k$  is large, then the difference of the sum  $\sum_{C_j \in N_i} b_j u_j$  when input is  $\mathbf{q}$  and the sum  $\sum_{C_j \in N_i} b_j u_j$  when input is  $\mathbf{u}^k$  is large, so  $w_i$  of input =  $\mathbf{q}$  is not similar to  $w_i$  of input =  $\mathbf{u}^k$ . Therefore the dynamic route of input =  $\mathbf{q}$  is not similar to the dynamic route of input =  $\mathbf{u}^k$ . So the equilibrium point of input =  $\mathbf{q}$  is not similar to the equilibrium point of input =  $\mathbf{u}^k$ . If the equilibrium point of input =  $\mathbf{u}^k$  is greater than +1, then the equilibrium point of input =  $\mathbf{q}$  is possibly less than -1; if the equilibrium point of input =  $\mathbf{u}^k$  is less than -1, then the equilibrium point of input =  $\mathbf{q}$  is possibly greater than +1. So the output when input is  $\mathbf{q}$  is possibly not equal to the output when input is  $\mathbf{u}^k$ . Therefore we cannot recover the pattern  $\mathbf{u}^k$  which is noisy pattern  $\mathbf{q}$  most similar to when we input  $\mathbf{q}$  to the

network. So the number of different elements of input  $\mathbf{q}$  correspond to its most similar pattern  $\mathbf{u}^k$  cannot be too large. Namely the Hamming distance between input  $\mathbf{q}$  and its most similar pattern  $\mathbf{u}^k$  cannot be too large. This states that the error correcting ability of associative memories is limited.

#### 4.3.5 Comparison of DT-CNN Associative Memory and Hopfield Associative Memory

When a Hopfield neural network is used for designing associative memories, the weightings of the Hopfield neural network are calculated by Hebb's rule. In this thesis, we use the synthesis procedure presented by G. Grassi to design DT-CNNs for associative memories. The input data are fed via initial conditions in Hopfield neural network and the outputs reach their steady state value at an equilibrium point which depends on the initial conditions, this approach presents a drawback from the VLSI implementation point of view, that is, initial conditions are required to be set to zero each time the network is run. Obviously, this is an undesirable feature for networks running in real time [19]. But the input data are fed via external inputs in cellular neural networks, each trajectory converges to a unique equilibrium point, which depends only on the input and not on the initial state. This feature makes perceive the possibility of implementing associative memories via cellular neural networks running in real time.

#### 4.4 Pattern Recognition of Using DT-CNN Associative Memories

After calculating  $\mathbf{B}$  and  $\mathbf{J}$ , we finish designing the part of associative memories, then we do the part of recognition. We input the known  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{e}$  and the testing pattern as  $\mathbf{u}$  to the motion equation  $\mathbf{x} = \mathbf{A} \mathbf{y} + \mathbf{B} \mathbf{u} + \mathbf{e}$ , then we can get the state value at the next time, and then we use output function to calculate the output at the next time, we use the output function of the standard cellular neural network here. We calculate state value and output as so until all output values are not changed anymore, then final output vector is the classification of the testing pattern. The algorithm and flowchart are shown in the following:

**Algorithm 4.2:** Use DT-CNN associative memories to recognize patterns

**Input:**  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{e}$  and the testing pattern  $\mathbf{u}$  in the motion equation

**Output:** Classification of the testing pattern  $\mathbf{u}$

Start:

Step 1: Set up initial output vector  $\mathbf{y}$ , its element values are all in  $[-1, 1]$  interval.

Step 2: Input the known  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{e}$ ,  $\mathbf{y}$  and testing pattern  $\mathbf{u}$  into the motion equation.

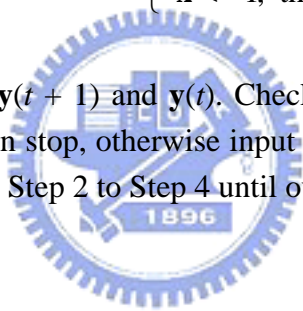
$$\mathbf{x}(t + 1) = \mathbf{A} \mathbf{y}(t) + \mathbf{B} \mathbf{u} + \mathbf{e}$$

Step 3: Input  $\mathbf{x}(t + 1)$  into activation function, then get new output  $\mathbf{y}(t + 1)$ .

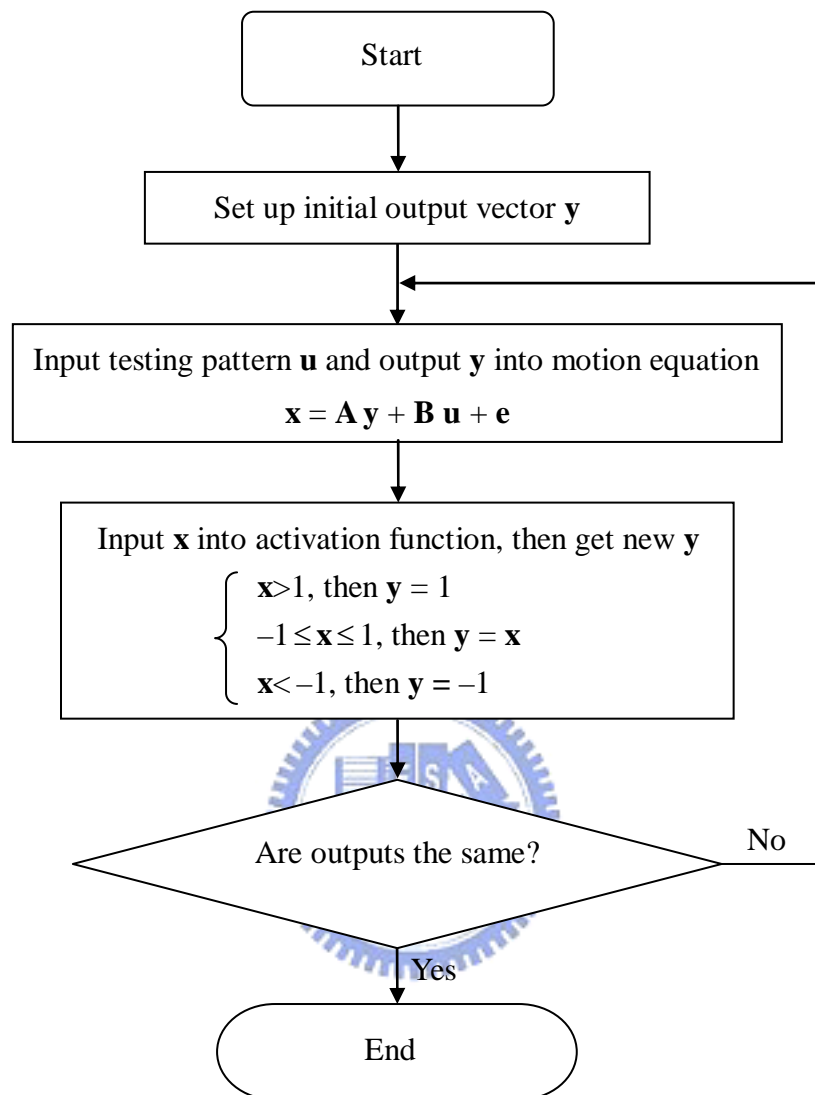
$$\text{activation function: } \begin{cases} \mathbf{x} > 1, \text{ then } \mathbf{y} = 1 \\ -1 \leq \mathbf{x} \leq 1, \text{ then } \mathbf{y} = \mathbf{x} \\ \mathbf{x} < -1, \text{ then } \mathbf{y} = -1 \end{cases}$$

Step 4: Compare new output  $\mathbf{y}(t + 1)$  and  $\mathbf{y}(t)$ . Check whether they are the same, if they are the same, then stop, otherwise input new output  $\mathbf{y}(t + 1)$  into motion equation again, repeat Step 2 to Step 4 until output  $\mathbf{y}$  is not changed.

End



**The flowchart of Algorithm 4.2:**



**4.5 Example**

We use an example to explain the process of pattern recognition with DT-CNNs. In this example, we suppose that there are two training patterns and one testing pattern, as Fig. 4.5 shows. We use “ 1 ” to represent white color and use “ -1 “ to represent black color.

Pattern size	$2 \times 3$
Number of training patterns	2
Neighborhood radius $r$	1
1-D template	[0.01 0.01 0.01]
$\alpha$	3

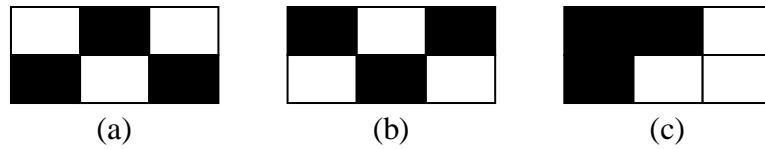


Fig. 4.5 (a) The first training pattern; (b) The second training pattern; (c) Testing pattern.

The first and second training patterns and testing pattern can be represented by three vectors  $\mathbf{u}^1$ ,  $\mathbf{u}^2$ , and  $\mathbf{t}$  respectively.

$$\mathbf{u}^1 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{u}^2 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

The process of pattern recognition:

Training:

Step 1: Calculate matrix  $\mathbf{U}$  from known  $\mathbf{u}^i$ .

$$\mathbf{U} = [\mathbf{u}^1 \quad \mathbf{u}^2] = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Step 2: Establish matrix  $\mathbf{Y} = \text{matrix } \mathbf{U}$ .

$$\mathbf{Y} = \mathbf{U} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Step 3: Establish matrix  $\mathbf{S}$ .

$$\mathbf{S} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}_{6 \times 6}$$

Step 4: Design matrix  $\mathbf{A}$  as a circulant matrix which satisfies globally asymptotically stable condition.

Globally asymptotically stable condition:

$$\left| \sum_{h=-r}^r a(h) e^{-j2\pi hq/n} \right| < 1, q = 0, 1, 2, \dots, n-1$$

$q = 0$ :

$$\left| \sum_{h=-1}^1 a(h) e^{-j2\pi hq/6} \right| = |0.01 + 0.01 + 0.01| = 0.03 < 1$$

$q = 1$ :

$$\left| \sum_{h=-1}^1 a(h) e^{-j2\pi hq/6} \right| = |0.02| = 0.02 < 1$$

$q = 2$ :

$$\left| \sum_{h=-1}^1 a(h) e^{-j2\pi hq/6} \right| = |5.2042 \times 10^{-18}| = 5.2042 \times 10^{-18} < 1$$

$q = 3:$

$$\left| \sum_{h=-1}^1 a(h)e^{-j2\pi hq/6} \right| = |-0.01| = 0.01 < 1$$

$q = 4:$

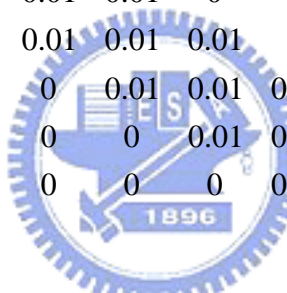
$$\left| \sum_{h=-1}^1 a(h)e^{-j2\pi hq/6} \right| = |-8.6736 \times 10^{-18}| = 8.6736 \times 10^{-18} < 1$$

$q = 5:$

$$\left| \sum_{h=-1}^1 a(h)e^{-j2\pi hq/6} \right| = |0.02| = 0.02 < 1$$

So 1-D template [0.01 0.01 0.01] satisfies globally asymptotically stable condition.

Design matrix  $\mathbf{A}$  as a circulant matrix:

$$\mathbf{A} = \begin{bmatrix} 0.01 & 0.01 & 0 & 0 & 0 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0.01 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0 & 0 & 0 & 0.01 & 0.01 \end{bmatrix}_{6 \times 6}$$


Step 5: Calculate  $\mathbf{X} = \alpha \mathbf{Y}$ .

$$\mathbf{X} = \alpha \mathbf{Y} = 3 \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & -3 \\ -3 & 3 \\ 3 & -3 \\ -3 & 3 \\ 3 & -3 \\ -3 & 3 \end{bmatrix}$$



Step 6: Calculate  $\mathbf{A}_y = \mathbf{A}\mathbf{Y}$ .

$$\mathbf{A}_y = \mathbf{A}\mathbf{Y} = \begin{bmatrix} 0.01 & 0.01 & 0 & 0 & 0 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0.01 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0 & 0 & 0 & 0.01 & 0.01 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}$$

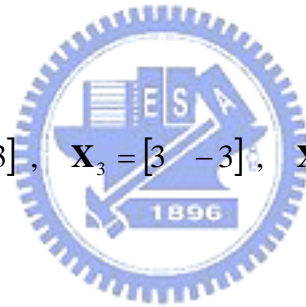
$$= \begin{bmatrix} -0.01 & 0.01 \\ 0.01 & -0.01 \\ -0.01 & 0.01 \\ 0.01 & -0.01 \\ -0.01 & 0.01 \\ 0.01 & -0.01 \end{bmatrix}$$

Step 7: for ( $j=1$  to 6) do:

Calculate  $\mathbf{X}_j$  by  $\mathbf{X}$ :

$$\mathbf{X}_1 = [3 \ -3], \quad \mathbf{X}_2 = [-3 \ 3], \quad \mathbf{X}_3 = [3 \ -3], \quad \mathbf{X}_4 = [-3 \ 3], \quad \mathbf{X}_5 = [3 \ -3],$$

$$\mathbf{X}_6 = [-3 \ 3].$$



Calculate  $\mathbf{A}_{y,j}$  by  $\mathbf{A}_y$ :

$$\mathbf{A}_{y,1} = [-0.01 \ 0.01], \quad \mathbf{A}_{y,2} = [0.01 \ -0.01], \quad \mathbf{A}_{y,3} = [-0.01 \ 0.01],$$

$$\mathbf{A}_{y,4} = [0.01 \ -0.01], \quad \mathbf{A}_{y,5} = [-0.01 \ 0.01], \quad \mathbf{A}_{y,6} = [0.01 \ -0.01].$$

Calculate  $\mathbf{R}$ ,  $\mathbf{R} = [\mathbf{U}^T \mathbf{h}]$ :

$$\mathbf{R} = [\mathbf{U}^T \mathbf{h}] = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}$$

Establish matrix  $\tilde{\mathbf{R}}_j$  from matrix  $\mathbf{S}$  and matrix  $\mathbf{R}$ :

$$\tilde{\mathbf{R}}_1 = \begin{bmatrix} 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 & 1 \end{bmatrix}, \quad \tilde{\mathbf{R}}_2 = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix},$$

$$\tilde{\mathbf{R}}_3 = \begin{bmatrix} -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 \end{bmatrix}, \quad \tilde{\mathbf{R}}_4 = \begin{bmatrix} 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 & 1 \end{bmatrix},$$

$$\tilde{\mathbf{R}}_5 = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}, \quad \tilde{\mathbf{R}}_6 = \begin{bmatrix} -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 \end{bmatrix}.$$

Calculate inverse matrix  $\tilde{\mathbf{R}}_j^+$  of  $\tilde{\mathbf{R}}_j$ :

$$\tilde{\mathbf{R}}_1^+ = \begin{bmatrix} 0.125 & -0.125 \\ -0.125 & 0.125 \\ -0.125 & 0.125 \\ 0.125 & -0.125 \\ 0.5 & 0.5 \end{bmatrix}, \quad \tilde{\mathbf{R}}_2^+ = \begin{bmatrix} 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.5 & 0.5 \end{bmatrix}, \quad \tilde{\mathbf{R}}_3^+ = \begin{bmatrix} -0.125 & 0.125 \\ 0.125 & -0.125 \\ 0.125 & -0.125 \\ -0.125 & 0.125 \\ 0.5 & 0.5 \end{bmatrix},$$

$$\tilde{\mathbf{R}}_4^+ = \begin{bmatrix} 0.125 & -0.125 \\ -0.125 & 0.125 \\ -0.125 & 0.125 \\ 0.125 & -0.125 \\ 0.5 & 0.5 \end{bmatrix}, \quad \tilde{\mathbf{R}}_5^+ = \begin{bmatrix} 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.5 & 0.5 \end{bmatrix}, \quad \tilde{\mathbf{R}}_6^+ = \begin{bmatrix} -0.125 & 0.125 \\ 0.125 & -0.125 \\ 0.125 & -0.125 \\ -0.125 & 0.125 \\ 0.5 & 0.5 \end{bmatrix}.$$

Calculate  $\tilde{\mathbf{w}}_j^T$ ,  $\tilde{\mathbf{w}}_j^T = \tilde{\mathbf{R}}_j^+ (\mathbf{X}_j^T - \mathbf{A}_{y,j}^T)$ :

$$\tilde{\mathbf{w}}_1^T = \tilde{\mathbf{R}}_1^+ (\mathbf{X}_1^T - \mathbf{A}_{y,1}^T) = \begin{bmatrix} 0.125 & -0.125 \\ -0.125 & 0.125 \\ -0.125 & 0.125 \\ 0.125 & -0.125 \\ 0.5 & 0.5 \end{bmatrix} \cdot \left( \begin{bmatrix} 3 \\ -3 \end{bmatrix} - \begin{bmatrix} -0.01 \\ 0.01 \end{bmatrix} \right) = \begin{bmatrix} 0.7525 \\ -0.7525 \\ -0.7525 \\ 0.7525 \\ 0 \end{bmatrix},$$

$$\tilde{\mathbf{w}}_2^T = \tilde{\mathbf{R}}_2^+ (\mathbf{X}_2^T - \mathbf{A}_{y,2}^T) = \begin{bmatrix} 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.5 & 0.5 \end{bmatrix} \cdot \left( \begin{bmatrix} -3 \\ 3 \end{bmatrix} - \begin{bmatrix} 0.01 \\ -0.01 \end{bmatrix} \right) = \begin{bmatrix} -0.5017 \\ 0.5017 \\ -0.5017 \\ 0.5017 \\ -0.5017 \\ 0.5017 \\ 0 \end{bmatrix},$$

$$\begin{aligned}
\tilde{\mathbf{w}}_3^T &= \tilde{\mathbf{R}}_3^+ (\mathbf{X}_3^T - \mathbf{A}_{y,3}^T) = \begin{bmatrix} -0.125 & 0.125 \\ 0.125 & -0.125 \\ 0.125 & -0.125 \\ -0.125 & 0.125 \\ 0.5 & 0.5 \end{bmatrix} \left( \begin{bmatrix} 3 \\ -3 \end{bmatrix} - \begin{bmatrix} -0.01 \\ 0.01 \end{bmatrix} \right) = \begin{bmatrix} -0.7525 \\ 0.7525 \\ 0.7525 \\ -0.7525 \\ 0 \end{bmatrix}, \\
\tilde{\mathbf{w}}_4^T &= \tilde{\mathbf{R}}_4^+ (\mathbf{X}_4^T - \mathbf{A}_{y,4}^T) = \begin{bmatrix} 0.125 & -0.125 \\ -0.125 & 0.125 \\ -0.125 & 0.125 \\ 0.125 & -0.125 \\ 0.5 & 0.5 \end{bmatrix} \cdot \left( \begin{bmatrix} -3 \\ 3 \end{bmatrix} - \begin{bmatrix} 0.01 \\ -0.01 \end{bmatrix} \right) = \begin{bmatrix} -0.7525 \\ 0.7525 \\ 0.7525 \\ -0.7525 \\ 0 \end{bmatrix}, \\
\tilde{\mathbf{w}}_5^T &= \tilde{\mathbf{R}}_5^+ (\mathbf{X}_5^T - \mathbf{A}_{y,5}^T) = \begin{bmatrix} 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.0833 & -0.0833 \\ 0.0833 & -0.0833 \\ -0.0833 & 0.0833 \\ 0.5 & 0.5 \end{bmatrix} \left( \begin{bmatrix} 3 \\ -3 \end{bmatrix} - \begin{bmatrix} -0.01 \\ 0.01 \end{bmatrix} \right) = \begin{bmatrix} 0.5017 \\ -0.5017 \\ 0.5017 \\ -0.5017 \\ 0.5017 \\ 0.5017 \\ -0.5017 \\ 0 \end{bmatrix}, \\
\tilde{\mathbf{w}}_6^T &= \tilde{\mathbf{R}}_6^+ (\mathbf{X}_6^T - \mathbf{A}_{y,6}^T) = \begin{bmatrix} -0.125 & 0.125 \\ 0.125 & -0.125 \\ 0.125 & -0.125 \\ -0.125 & 0.125 \\ 0.5 & 0.5 \end{bmatrix} \left( \begin{bmatrix} -3 \\ 3 \end{bmatrix} - \begin{bmatrix} 0.01 \\ -0.01 \end{bmatrix} \right) = \begin{bmatrix} 0.7525 \\ -0.7525 \\ -0.7525 \\ 0.7525 \\ 0 \end{bmatrix}
\end{aligned}$$

Recover  $\mathbf{w}_j$  from  $\tilde{\mathbf{w}}_j^T$ :

$$\begin{aligned}
\mathbf{w}_1 &= [0.7525 \quad -0.7525 \quad 0 \quad -0.7525 \quad 0.7525 \quad 0 \quad 0], \\
\mathbf{w}_2 &= [-0.5017 \quad 0.5017 \quad -0.5017 \quad 0.5017 \quad -0.5017 \quad 0.5017 \quad 0], \\
\mathbf{w}_3 &= [0 \quad -0.7525 \quad 0.7525 \quad 0 \quad 0.7525 \quad -0.7525 \quad 0], \\
\mathbf{w}_4 &= [-0.7525 \quad 0.7525 \quad 0 \quad 0.7525 \quad -0.7525 \quad 0 \quad 0], \\
\mathbf{w}_5 &= [0.5017 \quad -0.5017 \quad 0.5017 \quad -0.5017 \quad 0.5017 \quad -0.5017 \quad 0], \\
\mathbf{w}_6 &= [0 \quad 0.7525 \quad -0.7525 \quad 0 \quad -0.7525 \quad 0.7525 \quad 0].
\end{aligned}$$

$$\text{So } \mathbf{B} = \begin{bmatrix} 0.7525 & -0.7525 & 0 & -0.7525 & 0.7525 & 0 \\ -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 \\ 0 & -0.7525 & 0.7525 & 0 & 0.7525 & -0.7525 \\ -0.7525 & 0.7525 & 0 & 0.7525 & -0.7525 & 0 \\ 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 \\ 0 & 0.7525 & -0.7525 & 0 & -0.7525 & 0.7525 \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Recognition:

Step 1: Set up initial output vector  $\mathbf{y}$ , its element values are all in  $[-1, 1]$  interval.

We use a random value vector as initial output vector  $\mathbf{y}$ .

$$\mathbf{y} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

Step 2: Input the known  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{e}$ ,  $\mathbf{y}$  and testing pattern  $\mathbf{t}$  as  $\mathbf{u}$  into the motion equation.

$\mathbf{x} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{u} + \mathbf{e}$

$$= \begin{bmatrix} 0.01 & 0.01 & 0 & 0 & 0 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0.01 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0 & 0 & 0 & 0.01 & 0.01 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 0.7525 & -0.7525 & 0 & -0.7525 & 0.7525 & 0 \\ -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 \\ 0 & -0.7525 & 0.7525 & 0 & 0.7525 & -0.7525 \\ -0.7525 & 0.7525 & 0 & 0.7525 & -0.7525 & 0 \\ 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 \\ 0 & 0.7525 & -0.7525 & 0 & -0.7525 & 0.7525 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.495 \\ -1.0133 \\ 1.515 \\ -1.495 \\ 1.0133 \\ -1.515 \end{bmatrix}$$

Step 3: Input  $\mathbf{x}$  into activation function  $\mathbf{f}$ , then get new output  $\mathbf{y}_n$ .

$$\mathbf{y}_n = \mathbf{f}(\mathbf{x}) = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

Step 4: Compare new output  $\mathbf{y}_n$  and  $\mathbf{y}$ . Check whether they are the same, if they are the same, then stop, otherwise input new output  $\mathbf{y}_n$  as  $\mathbf{y}$  into motion equation again, repeat Step 2 to Step 4 until output  $\mathbf{y}$  is not changed.

$$\mathbf{y}_n = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \Rightarrow \mathbf{y}_n \neq \mathbf{y}, \text{ set } \mathbf{y} = \mathbf{y}_n.$$



$$\mathbf{x} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{u} + \mathbf{e}$$

$$= \begin{bmatrix} 1.495 \\ -1.0033 \\ 1.495 \\ -1.495 \\ 1.0033 \\ -1.495 \end{bmatrix}$$

$$\mathbf{y}_n = \mathbf{f}(\mathbf{x}) = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{y}_n = \mathbf{y}, \text{ stop.}$$

$\mathbf{y}_n = \mathbf{u}^1$ , so the testing pattern belongs to the class of the first training pattern.

We use the energy equation [20]:

$$E(t) = -\frac{1}{2} \sum_i \sum_j a_{ij} y_i(t) y_j(t) + \frac{1}{2} \sum_i y_i(t)^2 - \sum_i \sum_j b_{ij} y_i(t) u_j - \sum_i I y_i(t)$$

Compute the energy after each iteration:

Iteration 0

:

$$\mathbf{y} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \Rightarrow E(0) = 6.0000$$

Iteration 1:

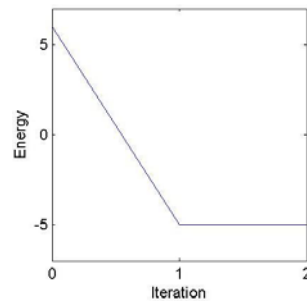
$$\mathbf{y} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \Rightarrow E(1) = -4.9967$$



Iteration 2:

$$\mathbf{y} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \Rightarrow E(2) = -4.9967$$

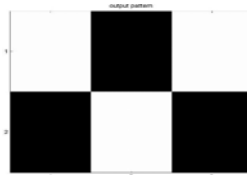
Energy curve:



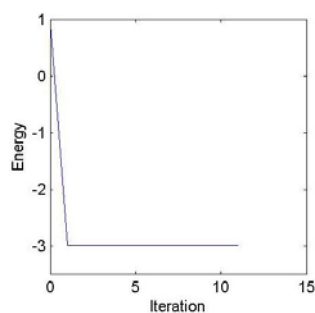
We use pseudoinverse technique to compute matrix  $\mathbf{B}$ . But pseudoinverse matrix of  $\mathbf{R}$  is not unique, so matrix  $\mathbf{B}$  is not unique. We compute matrix  $\mathbf{B}$  without using matrix  $\mathbf{S}$  to indicate the interconnecting structure of network inputs now. Then matrix  $\mathbf{B}$  is shown in the following:

$$\mathbf{B} = \begin{bmatrix} 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 \\ -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 \\ 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 \\ -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 \\ 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 \\ -0.5017 & 0.5017 & -0.5017 & 0.5017 & -0.5017 & 0.5017 \end{bmatrix}$$

The interconnecting structure of network inputs is fully connected. This is not the same with the general CNN structure. The result of this example with the fully connected CNN is the same with the locally connected CNN and is shown in the following:



The energy curve is shown in the following:



## 4.6 Experimental Results

In Experiment 1, we use two values for  $\alpha$  and several different feedback templates to observe the influences of  $\alpha$  and the feedback template for the network performance. In Experiment 2, 3, 4 and 5, we apply DT-CNN associative memory with matrix  $\mathbf{S}$ , DT-CNN associative memory without matrix  $\mathbf{S}$  and Hopfield associative memory to simulated seismic patterns and compare performances of these three systems. In Experiment 6, we apply the DT-CNN associative memory with matrix  $\mathbf{S}$  to real seismic patterns. In Experiment 7, we use an example to introduce the case of indefinite cells.

### Experiment 1.

To observe the effect of  $\alpha$  and the effect of feedback template, we use two different values of  $\alpha$  and different feedback templates to do the same experiment.

Pattern size	$7 \times 7$
The number of training patterns	3
Hamming distance(HD)	6
Neighborhood radius r	1, 2
$\alpha$	3, 100
1-D feedback template	[0.01 0.01 0.01], [0.0001 0.0001 0.0001], [0.01 0.01 0.01 0.01 0.01], [0.0001 0.0001 0.0001 0.0001 0.0001]



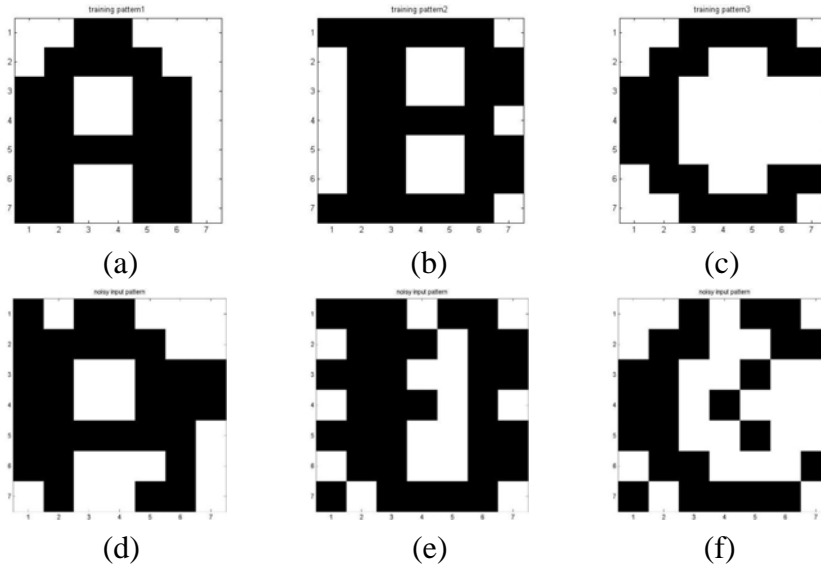


Fig. 4.6 (a) The 1st training pattern; (b) The 2nd training pattern; (c) The 3rd training pattern; (d) The 1st noisy input pattern; (e) The 2nd noisy input pattern; (f) The 3rd noisy input pattern.

Case 1:  $\alpha = 3$ ,  $r = 1$ , 1-D feedback template: [0.01 0.01 0.01]

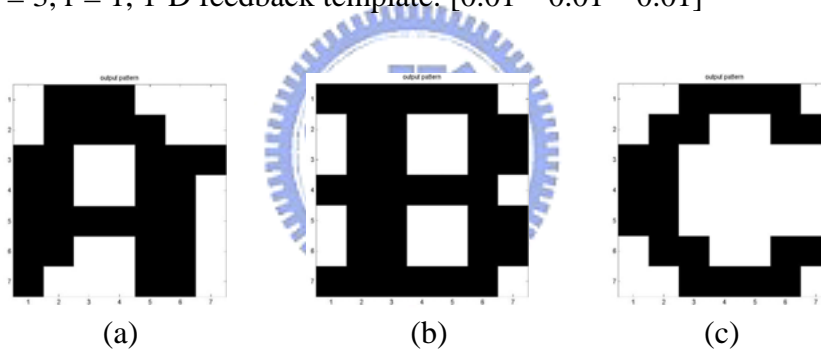


Fig. 4.7 (a) Output of Fig. 4.6(d); (b) Output of Fig. 4.6(e); (c) Output of Fig. 4.6(f).

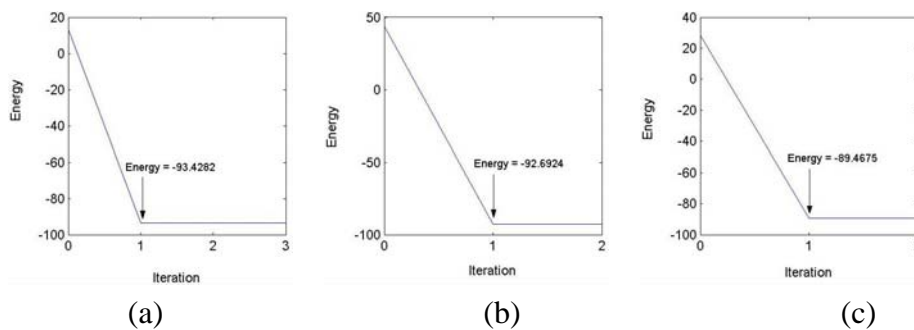


Fig. 4.8 (a) Energy curve of Fig. 4.6(d); (b) Energy curve of Fig. 4.6(e); (c) Energy curve of Fig. 4.6(f).

Case 2:  $\alpha = 100, r = 1$ , 1-D feedback template: [0.01 0.01 0.01]

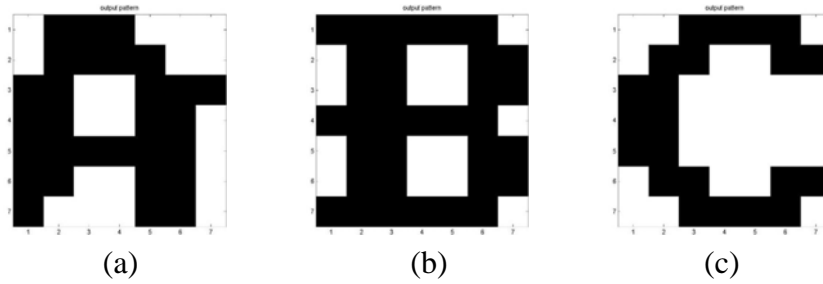


Fig. 4.9 (a) Output of Fig. 4.6(d); (b) Output of Fig. 4.6(e); (c) Output of Fig. 4.6(f).

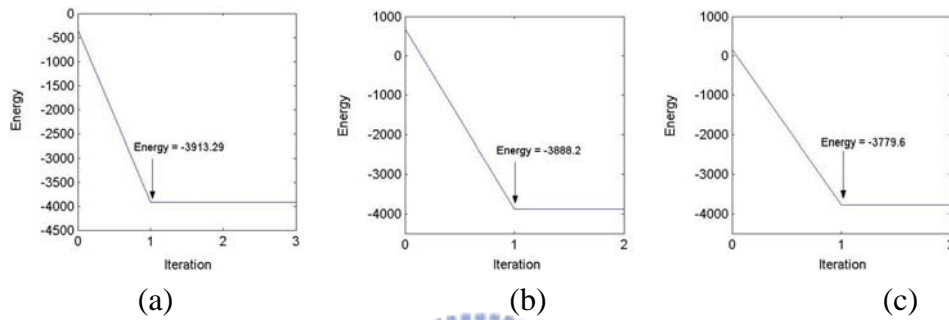


Fig. 4.10 (a) Energy curve of Fig. 4.6(d); (b) Energy curve of Fig. 4.6(e); (c) Energy curve of Fig. 4.6(f).

Case 3:  $\alpha = 3, r = 2$ , 1-D feedback template: [0.01 0.01 0.01 0.01 0.01]

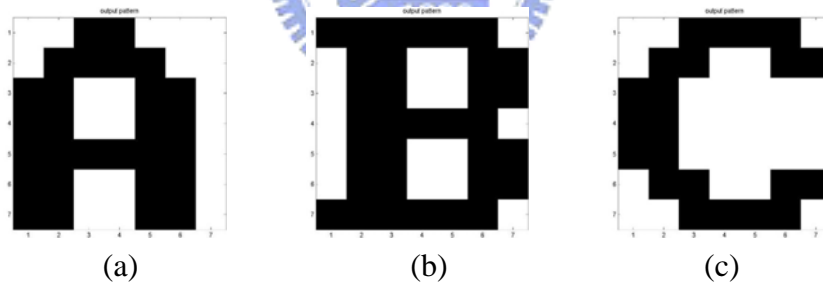


Fig. 4.11 (a) Output of Fig. 4.6(d); (b) Output of Fig. 4.6(e); (c) Output of Fig. 4.6(f).

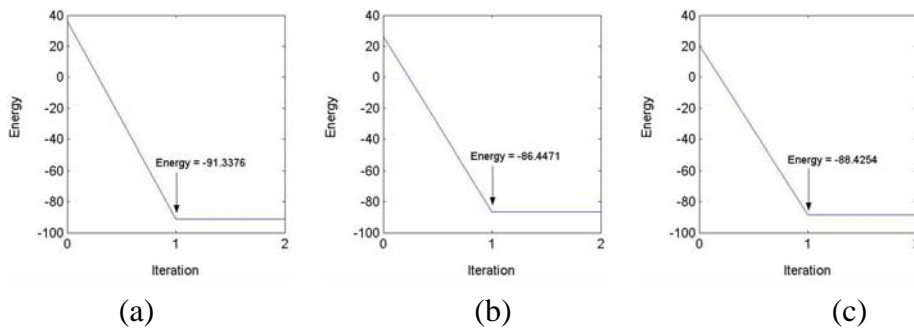


Fig. 4.12 (a) Energy curve of Fig. 4.6(d); (b) Energy curve of Fig. 4.6(e); (c) Energy curve of Fig. 4.6(f).

Case 4:  $\alpha = 100$ ,  $r = 2$ , 1-D feedback template: [0.01 0.01 0.01 0.01 0.01]

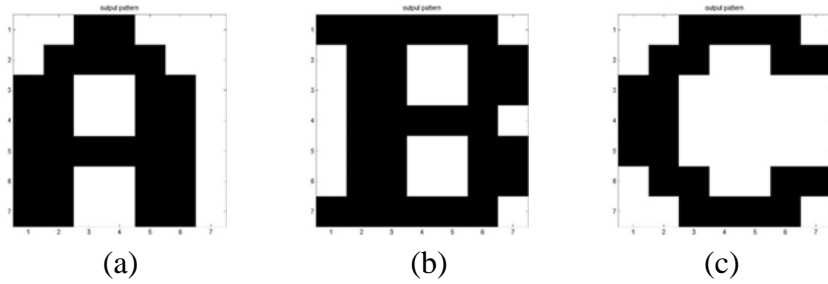


Fig. 4.13 (a) Output of Fig. 4.6(d); (b) Output of Fig. 4.6(e); (c) Output of Fig. 4.6(f).

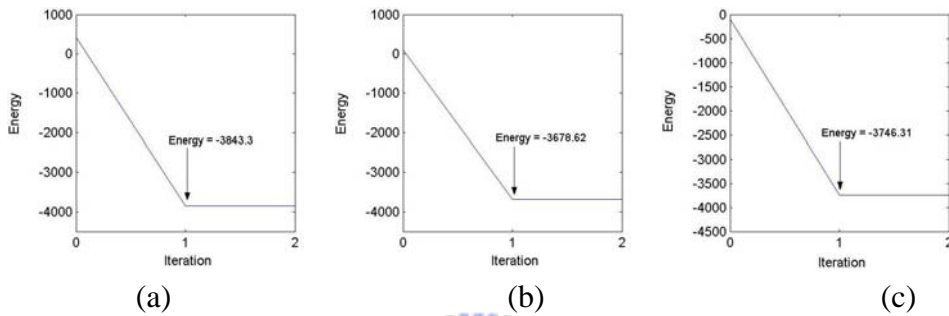


Fig. 4.14 (a) Energy curve of Fig. 4.6(d); (b) Energy curve of Fig. 4.6(e); (c) Energy curve of Fig. 4.6(f).

The outputs of Case 1 are equal to the outputs of Case 2. And the outputs of Case 3 are equal to the outputs of Case 4. So network performances are not affected by the choice of  $\alpha$ .

Case 5:  $\alpha = 3$ ,  $r = 1$ , 1-D feedback template: [0.0001 0.0001 0.0001]

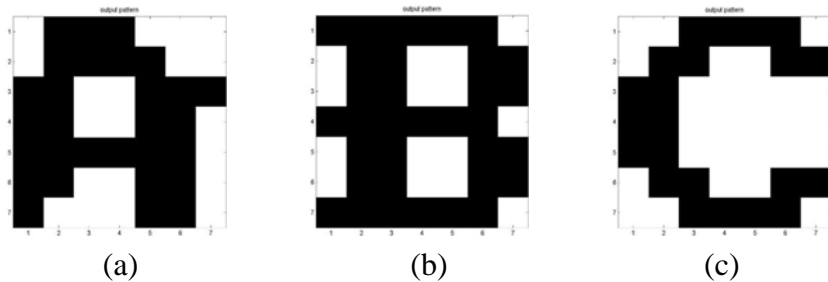


Fig. 4.15 (a) Output of Fig. 4.6(d); (b) Output of Fig. 4.6(e); (c) Output of Fig. 4.6(f).

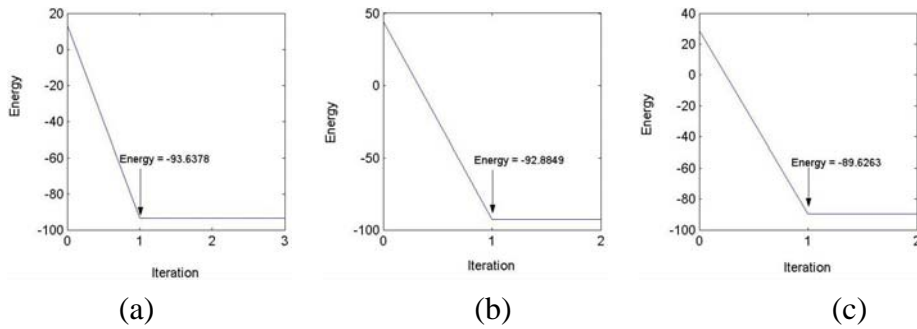


Fig. 4.16 (a) Energy curve of Fig. 4.6(d); (b) Energy curve of Fig. 4.6(e); (c) Energy curve of Fig. 4.6(f).

Case 6:  $\alpha = 3$ ,  $r = 2$ , 1-D feedback template: [0.0001 0.0001 0.0001 0.0001 0.0001]

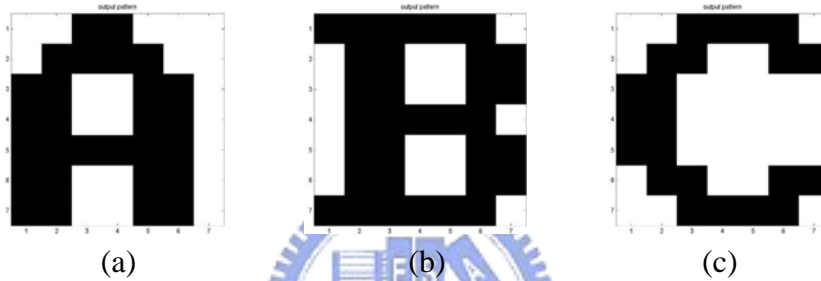


Fig. 4.17 (a) Output of Fig. 4.6(d); (b) Output of Fig. 4.6(e); (c) Output of Fig. 4.6(f).

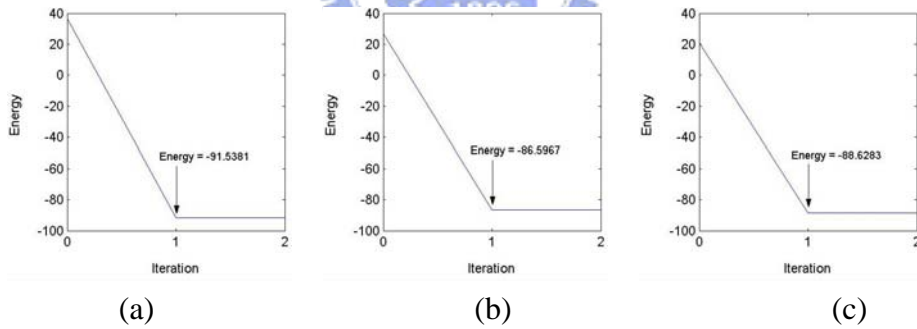


Fig. 4.18 (a) Energy curve of Fig. 4.6(d); (b) Energy curve of Fig. 4.6(e); (c) Energy curve of Fig. 4.6(f).

The outputs of Case 1 are equal to the outputs of Case 5. And the outputs of Case 3 are equal to the outputs of Case 6. So network performances are not affected by the choice of 1-D feedback template.

## Experiment 2.

In this experiment, we store two simulated seismic patterns and recognize a noisy input pattern.

Pattern size	$8 \times 12$
The number of training patterns	2
Hamming distance(HD)	6

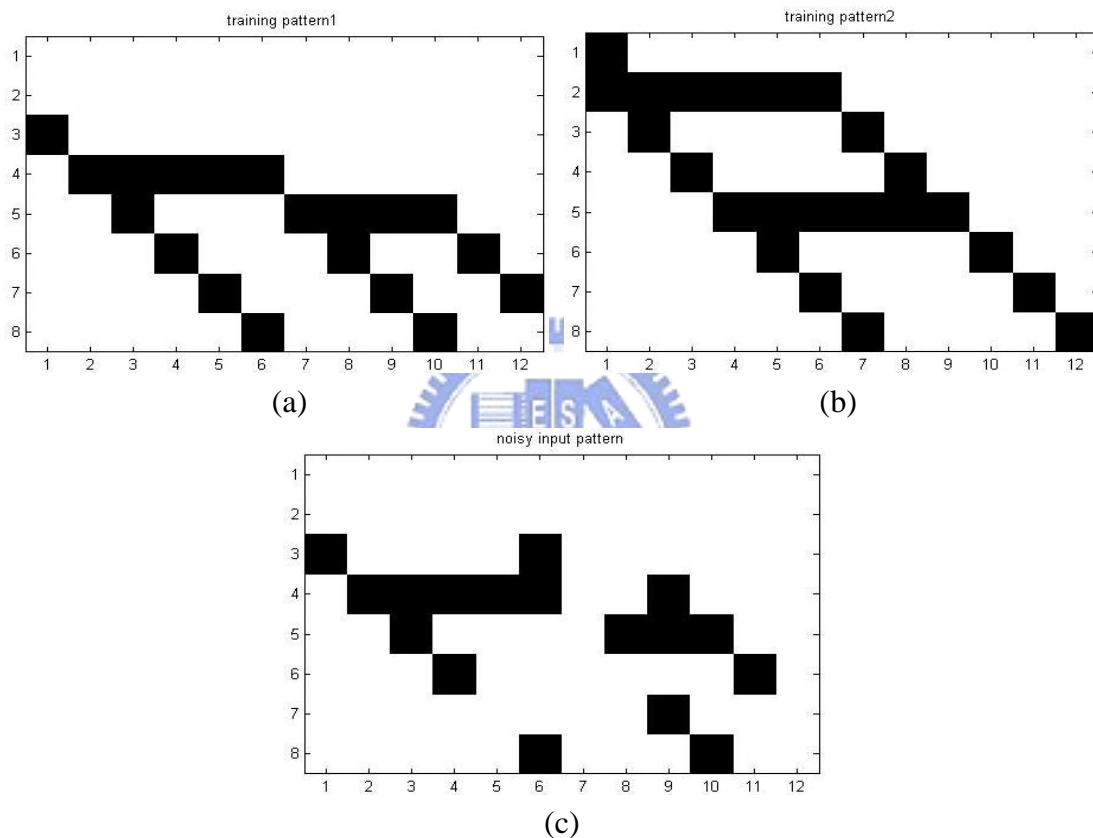


Fig. 4.19 (a) The 1st training pattern: sealevel fall; (b) The 2nd training pattern; (c) The noisy input pattern.

We apply this DT-CNN associative memory with matrix  $\mathbf{S}$  to simulated seismic patterns first. We set  $\alpha = 3$  and neighborhood radius  $r = 1$ . The result is shown in Fig. 4.20.

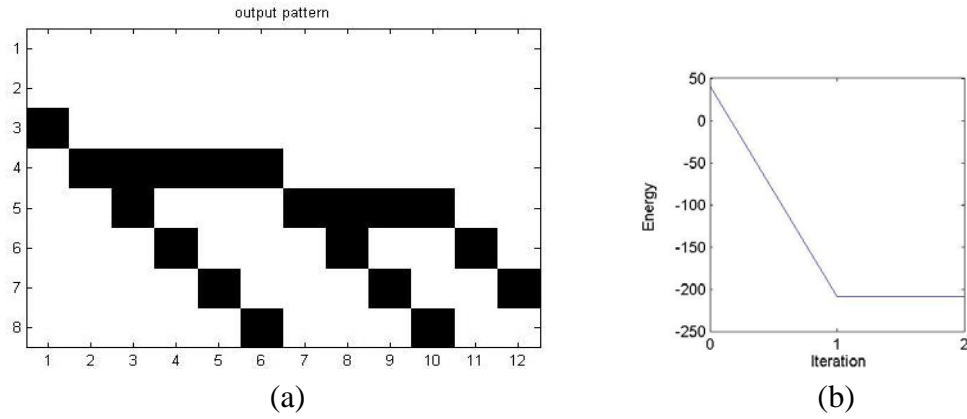


Fig. 4.20 (a) Output of Fig. 4.19(c); (b) Energy curve.

Next, we apply DT-CNN associative memory without matrix  $\mathbf{S}$  to this experiment. We set  $\alpha = 3$  and neighborhood radius  $r = 1$ . The output pattern is the same with Fig. 4.20(a).

And then we apply Hopfield associative memory to this experiment. The output pattern is also the same with Fig. 4.20(a). The comparison of these three systems for this experiment is shown in Table 4.1.

	DT-CNN with $\mathbf{S}$	DT-CNN without $\mathbf{S}$	Hopfield
	$r = 1$	$r = 1$	
Recognition	Success	Success	Success
Training time (second)	0.141	0.047	0.0003
Recognition time (second)	0.312	0.281	0.156

Table. 4.1 Comparison of three systems for Experiment 2.

### Experiment 3.

In this experiment, we store two simulated seismic patterns and recognize two noisy input patterns.

Pattern size	$19 \times 29$
The number of training patterns	2
Hamming distance(HD)	100, 90

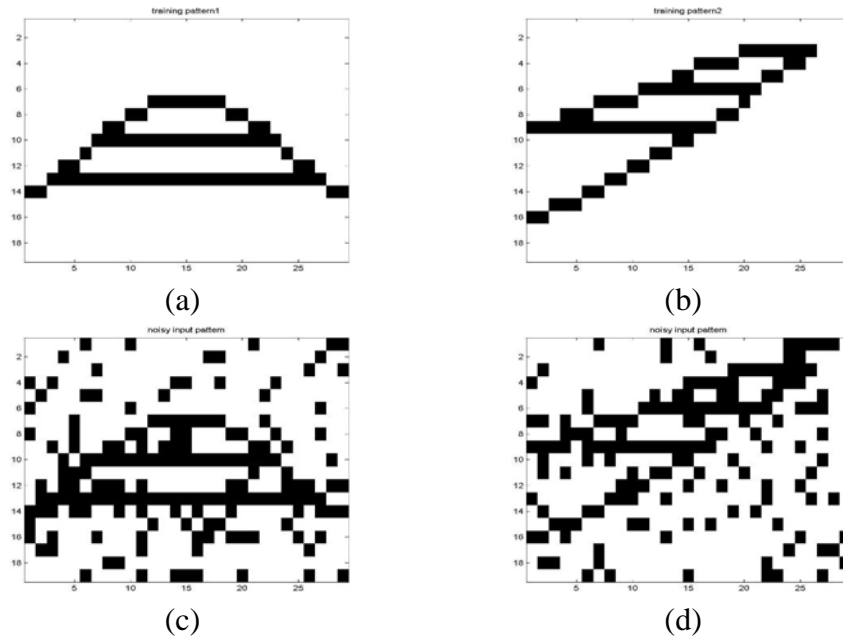


Fig. 4.21 (a) The 1st training pattern: bright-spot; (b) The 2nd training pattern: pinch-out; (c) The 1st noisy input pattern (HD = 100); (d) The 2nd noisy input pattern (HD = 90).

We apply this DT-CNN associative memory with matrix  $\mathbf{S}$  to simulated seismic patterns first. We set  $\alpha = 3$  and neighborhood radius  $r = 1, 2$  and  $3$ . The results are shown in Fig. 4.22, 4.23 and 4.24.

$r = 1$ :

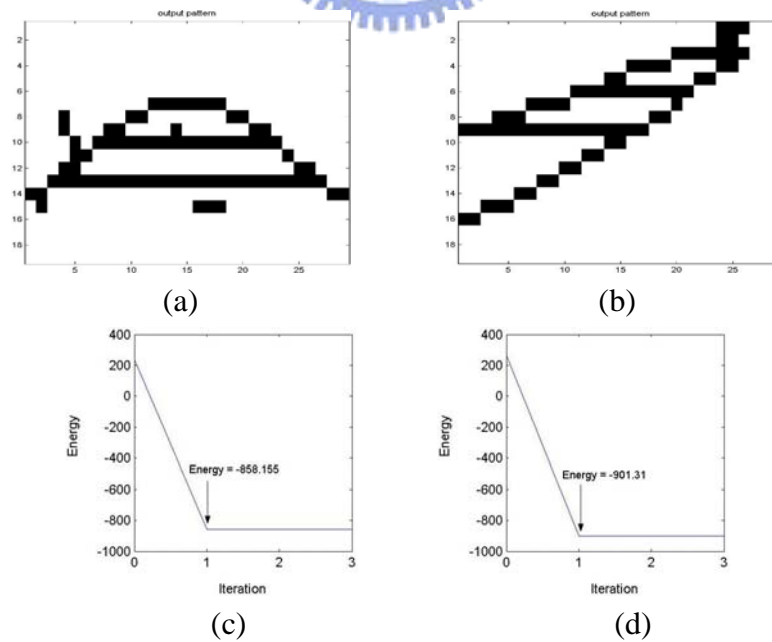


Fig. 4.22 (a) Output of Fig. 4.21(c); (b) Output of Fig. 4.21(d); (c) Energy curve of Fig. 4.21(c); (d) Energy curve of Fig. 4.21(d).

Fig. 4.22(a) and (b) are uncorrected output patterns, so we set neighborhood radius to 2 and try again.

$r = 2$ :

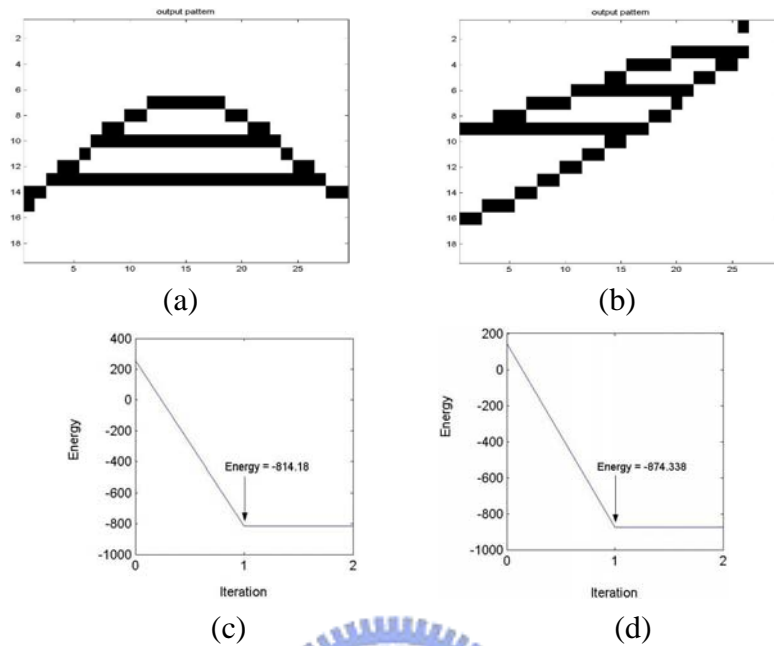


Fig. 4.23 (a) Output of Fig. 4.21(c); (b) Output of Fig. 4.21(d); (c) Energy curve of Fig. 4.21(c); (d) Energy curve of Fig. 4.21(d).

Fig. 4.23(a) and (b) are uncorrected output patterns, so we set neighborhood radius to 3 and try again.

$r = 3$ :

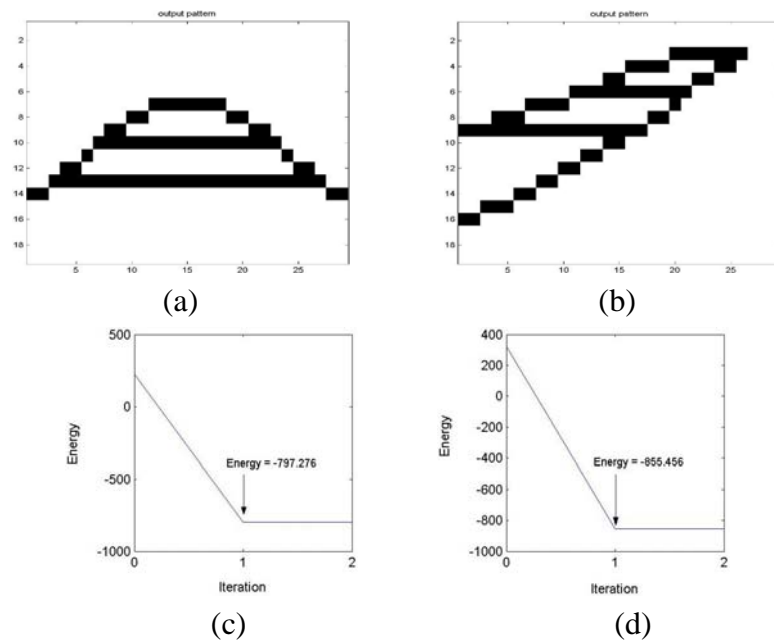


Fig. 4.24 (a) Output of Fig. 4.21(c); (b) Output of Fig. 4.21(d); (c) Energy curve of Fig. 4.21(c); (d) Energy curve of Fig. 4.21(d).



Two output patterns in Fig. 4.24 are all correct. Next, we apply DT-CNN associative memory without matrix  $\mathbf{S}$  to this experiment. We set  $\alpha = 3$  and neighborhood radius  $r = 1$ . The output patterns are the same with Fig. 4.24(a) and (b). And then we apply Hopfield associative memory to this experiment. The output patterns are also the same with Fig. 4.24(a) and (b). The comparison of these three systems for this experiment is shown in Table 4.2.

	DT-CNN with $\mathbf{S}$			DT-CNN without $\mathbf{S}$	Hopfield
	$r = 1$	$r = 2$	$r = 3$	$r = 1$	
Recognition	Failure	Failure	Success	Success	Success
Average training time (second)	8.5			7.25	0.047
Average recognition time (second)	0.3205			0.328	0.172

Table. 4.2 Comparison of three systems for Experiment 3.

#### Experiment 4.

In this experiment, we store three simulated seismic patterns and recognize three noisy input patterns.



Pattern size	$12 \times 48$
The number of training patterns	3
Hamming distance(HD)	107, 118, 118

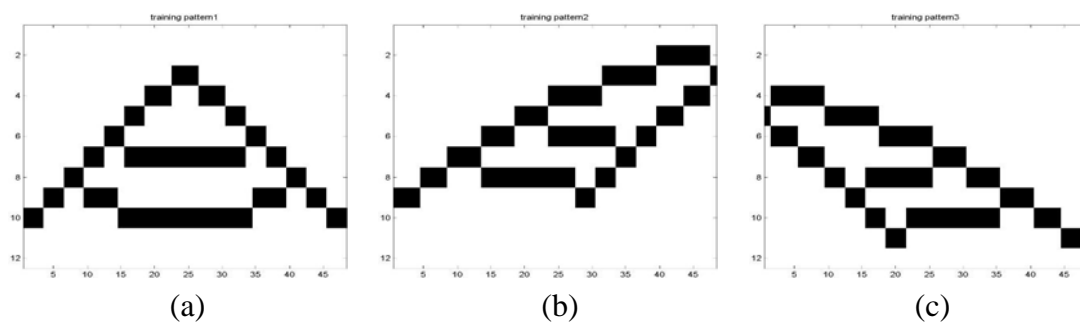


Fig. 4.25 (a) The 1st training pattern: bright-spot; (b) The 2nd training pattern: right pinch-out; (c) The 3rd training pattern: left pinch-out.

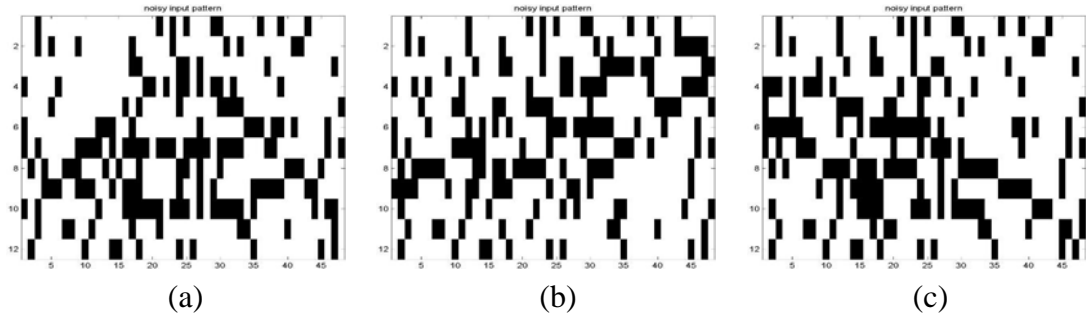


Fig. 4.26 (a) ~ (c) Three noisy input patterns with HD = 107, 118, 118 respectively.

We apply this DT-CNN associative memory with matrix  $S$  to simulated seismic patterns first. We set  $\alpha = 3$  and neighborhood radius  $r = 2, 3$  and  $4$ . The results are shown in Fig. 4.27, 4.28 and 4.29.

$r = 2$ :

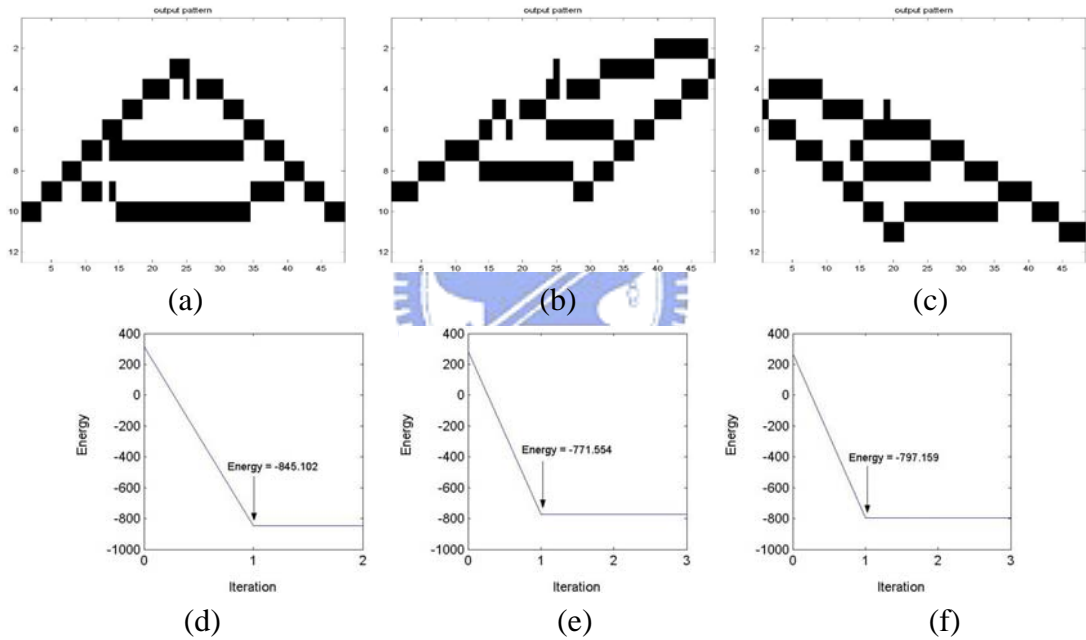


Fig. 4.27 (a) Output of Fig. 4.26(a); (b) Output of Fig. 4.26(b); (c) Output of Fig. 4.26(c); (d) Energy curve of Fig. 4.26(a); (e) Energy curve of Fig. 4.26(b); (f) Energy curve of Fig. 4.26(c).

Three output patterns in Fig. 4.27 are uncorrected output patterns, so we set neighborhood radius to 3 and try again.

$r = 3$ :

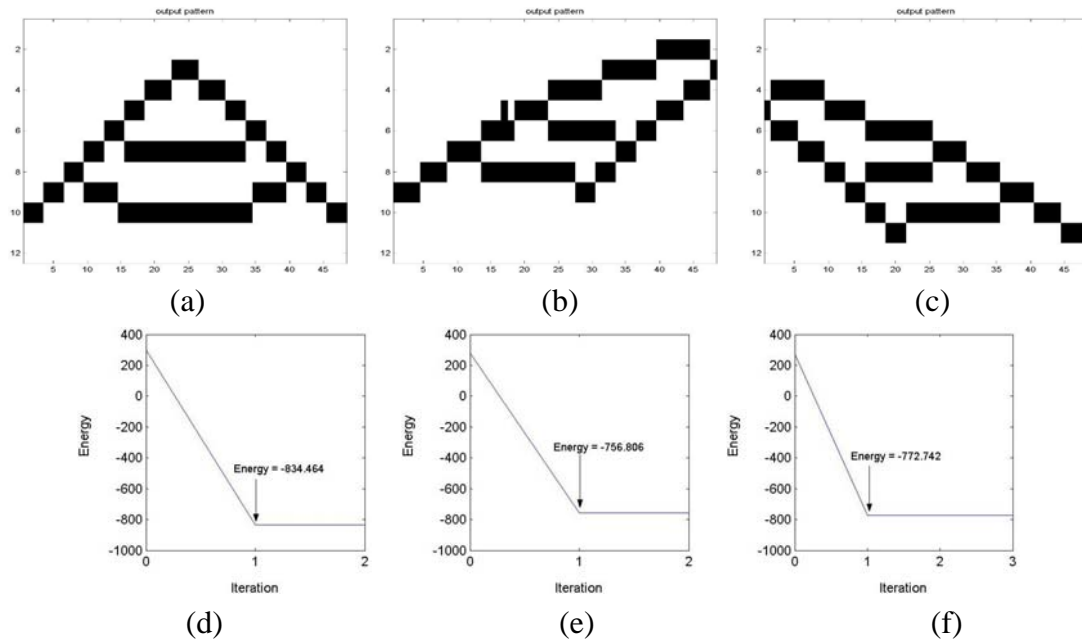


Fig. 4.28 (a) Output of Fig. 4.26(a); (b) Output of Fig. 4.26(b); (c) Output of Fig. 4.26(c); (d) Energy curve of Fig. 4.26(a); (e) Energy curve of Fig. 4.26(b); (f) Energy curve of Fig. 4.26(c).

Fig. 4.28(b) is an uncorrected output pattern, so we set neighborhood radius to 4 and try again.

$r = 4$ :

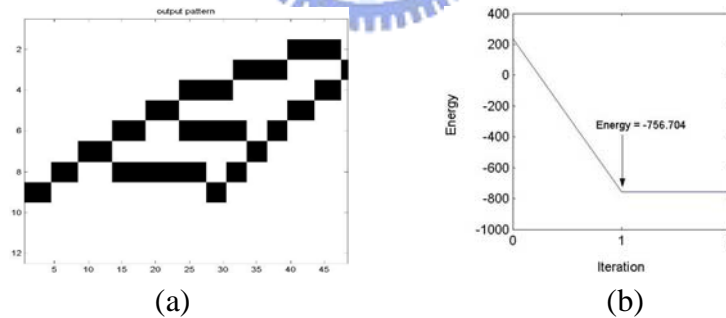


Fig. 4.29 (a) Output of Fig. 4.26(b); (b) Energy curve of Fig. 4.26(b).

Next, we apply DT-CNN associative memory without matrix  $S$  to this experiment. We set  $\alpha = 3$  and neighborhood radius  $r = 1$ . The output patterns are the same with Fig. 4.28(a), Fig. 4.29(a) and Fig. 4.28(c). And then we apply Hopfield associative memory to this experiment. The output patterns are shown in Fig. 4.30. The comparison of these three systems for this experiment is shown in Table 4.3.

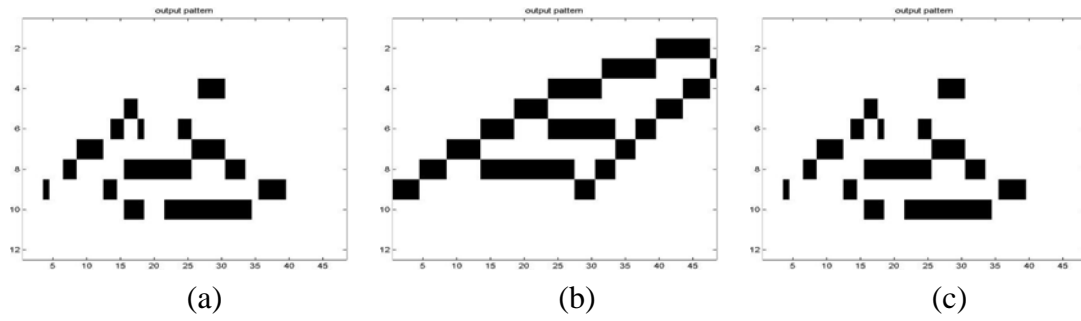


Fig. 4.30 (a) Output of Fig. 4.26(a) by Hopfield associative memory; (b) Output of Fig. 4.26(b) by Hopfield associative memory; (c) Output of Fig. 4.26(c) by Hopfield associative memory.

	DT-CNN with <b>S</b>			DT-CNN without <b>S</b>	Hopfield
	r = 2	r = 3	r = 4	r = 1	
Recognition	Failure	Failure	Success	Success	Failure
Average training time (second)	9.922			8.25	0.047
Average recognition time (second)	0.3177			0.3023	0.193

Table. 4.3 Comparison of three systems for Experiment 4.

### Experiment 5.

In this experiment, we store five simulated seismic patterns and recognize five noisy input patterns.

Pattern size	$19 \times 29$
The number of training patterns	5
Hamming distance(HD)	100, 90, 100, 100, 100

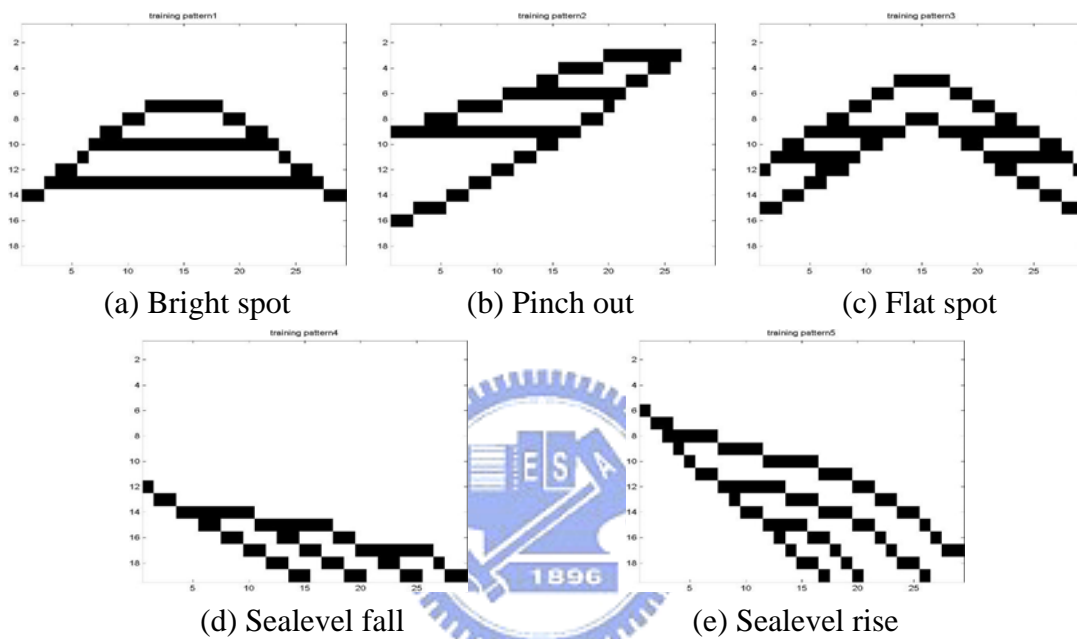


Fig. 4.31 (a) ~ (e) Five training patterns.

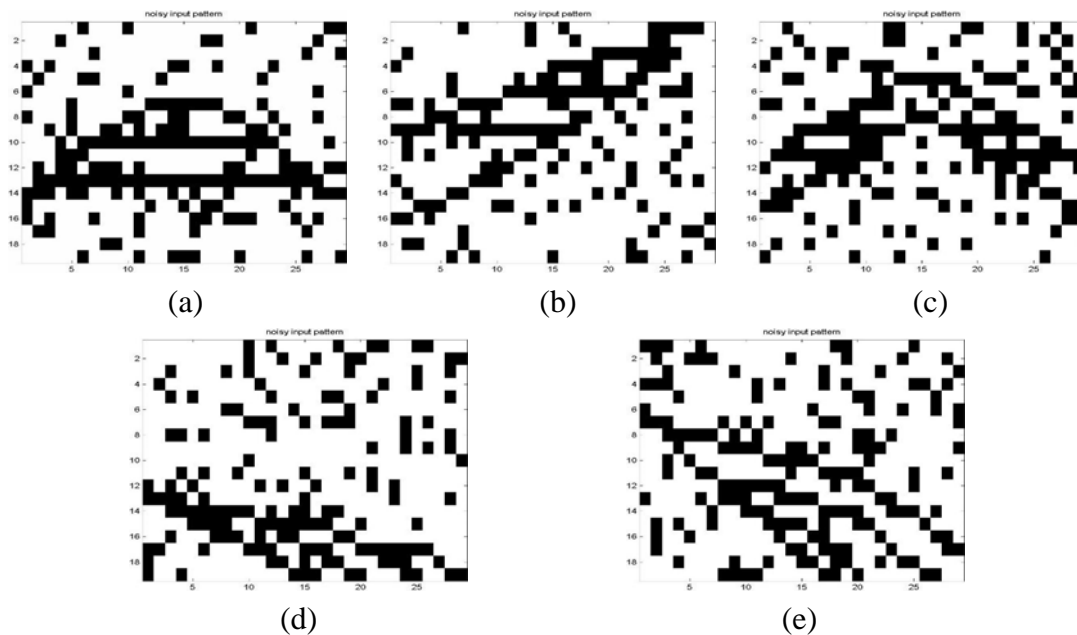


Fig. 4.32 (a) ~ (e) Five noisy input patterns with HD = 100, 90, 100, 100, 100 respectively.

We apply this DT-CNN associative memory with matrix  $\mathbf{S}$  to simulated seismic patterns first. We set  $\alpha = 3$  and neighborhood radius  $r = 3$  and 4. The results are shown in Fig. 4.33 and 4.34.

$r = 3$ :

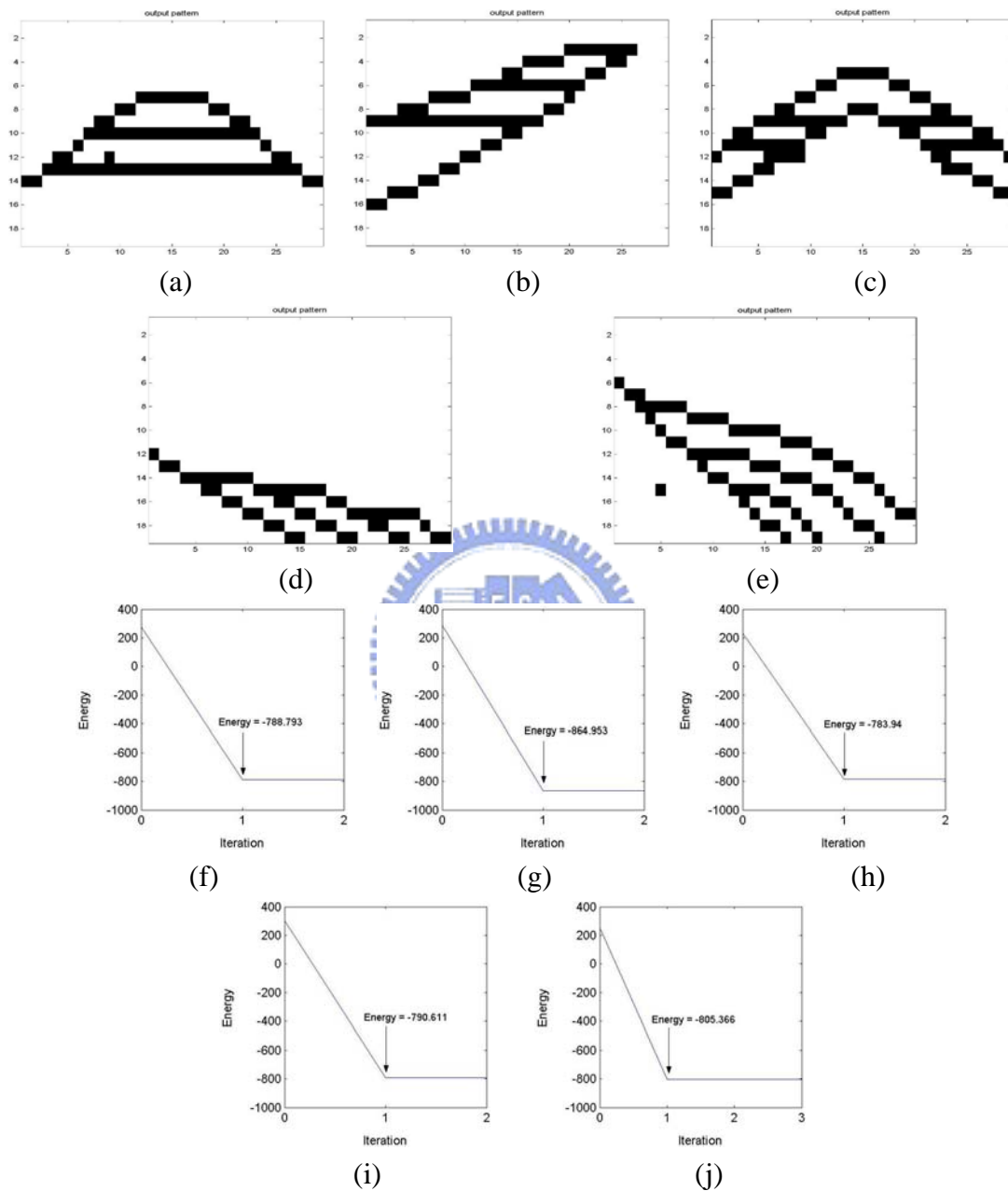


Fig. 4.33 (a) Output of Fig. 4.32(a); (b) Output of Fig. 4.32(b); (c) Output of Fig. 4.32(c); (d) Output of Fig. 4.32(d); (e) Output of Fig. 4.32(e); (f) Energy curve of Fig. 4.32(a); (g) Energy curve of Fig. 4.32(b); (h) Energy curve of Fig. 4.32(c); (i) Energy curve of Fig. 4.32(d); (j) Energy curve of Fig. 4.32(e).

Fig. 4.33(a), (c) and (e) are uncorrected output patterns, so we set neighborhood radius to 4 and try again.

$r = 4$ :

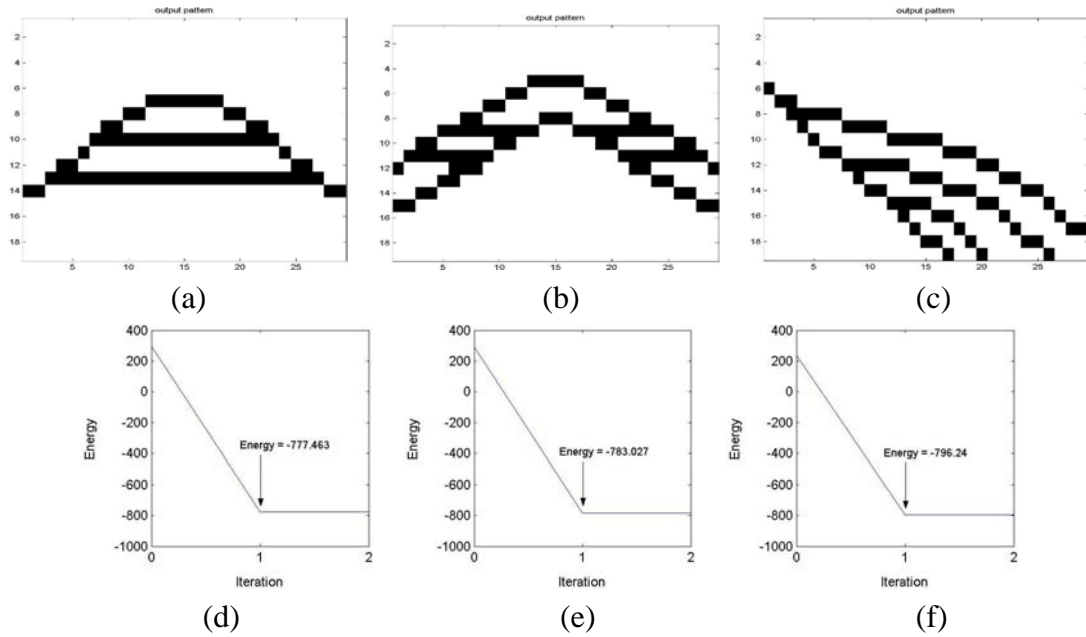


Fig. 4.34 (a) Output of Fig. 4.32(a); (b) Output of Fig. 4.32(c); (c) Output of Fig. 4.32(e); (d) Energy curve of Fig. 4.32(a); (e) Energy curve of Fig. 4.32(c); (f) Energy curve of Fig. 4.32(e).

Three output patterns in Fig. 4.34 are all correct. Next, we apply DT-CNN associative memory without matrix  $S$  to this experiment. We set  $\alpha = 3$  and neighborhood radius  $r = 1$ . The output patterns are the same with Fig. 4.34(a), Fig. 4.33(b), Fig. 4.34(b), Fig. 4.33(d) and Fig. 4.34(c). And then we apply Hopfield associative memory to this experiment. The output patterns are shown in Fig. 4.35. The comparison of these three systems for this experiment is shown in Table 4.4.

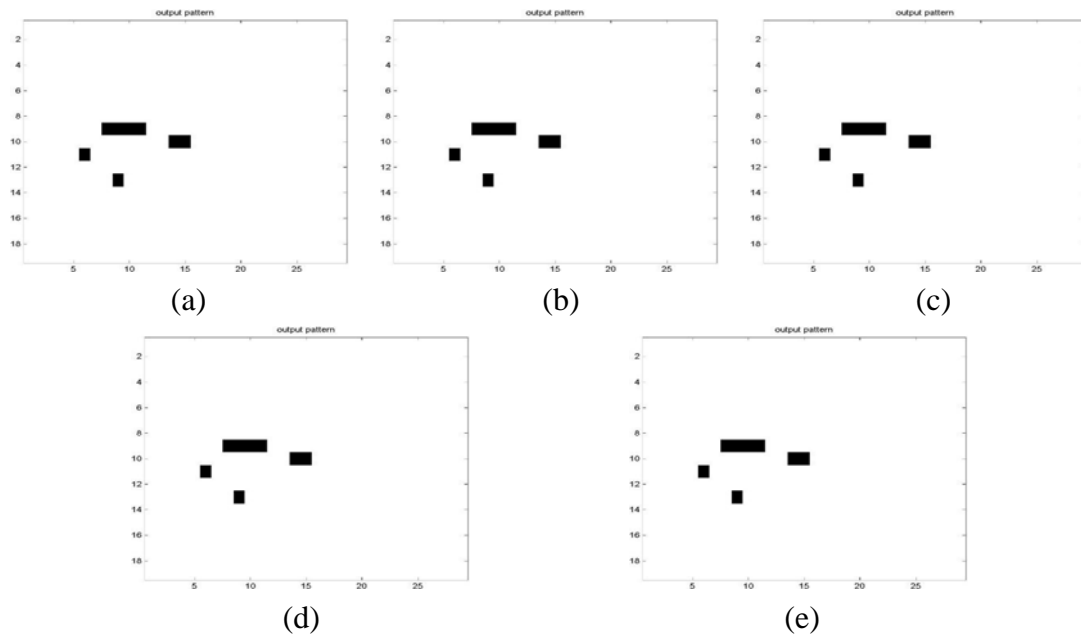


Fig. 4.35 (a) Output of Fig. 4.32(a) by Hopfield associative memory; (b) Output of Fig. 4.32(b) by Hopfield associative memory; (c) Output of Fig. 4.32(c) by Hopfield associative memory; (d) Output of Fig. 4.32(d) by Hopfield associative memory; (e) Output of Fig. 4.32(e) by Hopfield associative memory.

	CNN with <b>S</b>		CNN without <b>S</b>	Hopfield
	r = 3	r = 4	r = 1	
Recognition	Failure	Success	Success	Failure
Average training time (second)	9.281		7.5	0.079
Average recognition time (second)	0.294		0.2976	0.2622

Table. 4.4 Comparison of three systems for Experiment 5.



## Experiment 6.

We apply this DT-CNN associative memory with matrix  $\mathbf{S}$  to real seismic patterns. In this experiment, we store two training patterns and recognize three real seismic patterns.

Training pattern size	$16 \times 50$
Real seismic pattern size	$64 \times 64$
The number of training patterns	2
Neighborhood radius $r$	1, 3
$\alpha$	3

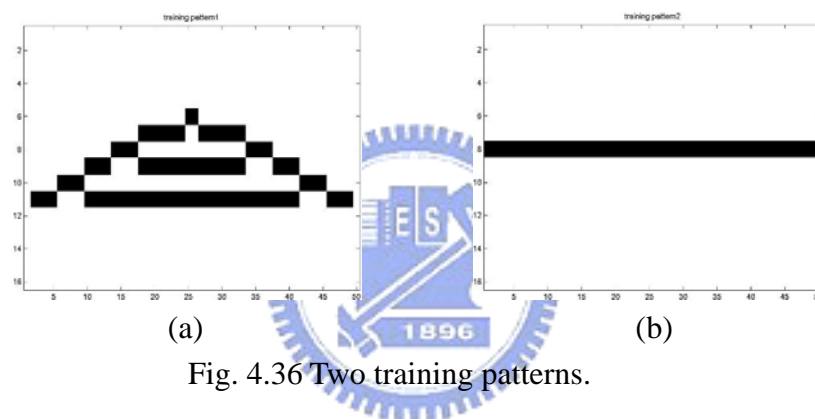


Fig. 4.36 Two training patterns.

The size of real seismic pattern is larger than the size of training pattern. We use a window to extract the testing pattern from real seismic pattern. The size of this window is equal to the size of training pattern. This window is shifted from left to right and top to bottom on the real seismic pattern. If the output pattern of the network is equal to one of training patterns. We record the coordinate of upper-left corner of the window. After the window is shifted to the last position on the real seismic pattern and all testing patterns are recognized, we calculate the coordinate of the center of all recorded coordinates which are about the same training pattern. And then we use the detected training pattern and the center coordinate to recover the pattern. We set neighborhood radius  $r = 1$  to process the first and second real seismic patterns and set  $r = 3$  to process the third real seismic pattern.

$r = 1$ :

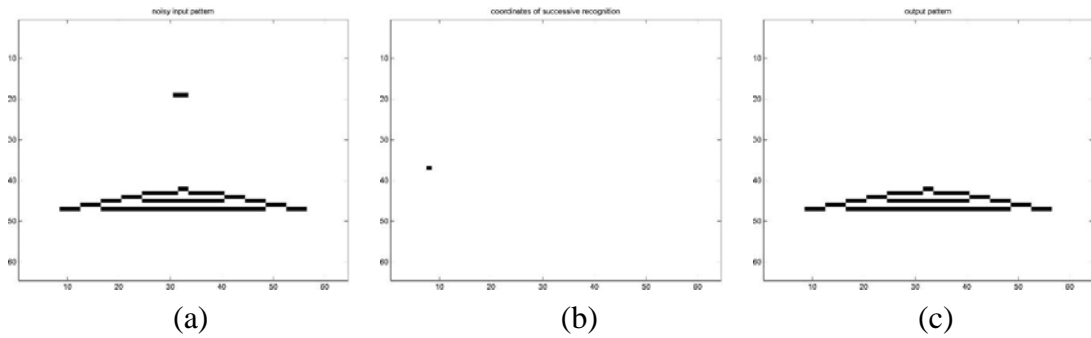


Fig. 4.37 (a) The first real seismic pattern; (b) Coordinates of successful recognition; (c) Output of (a).

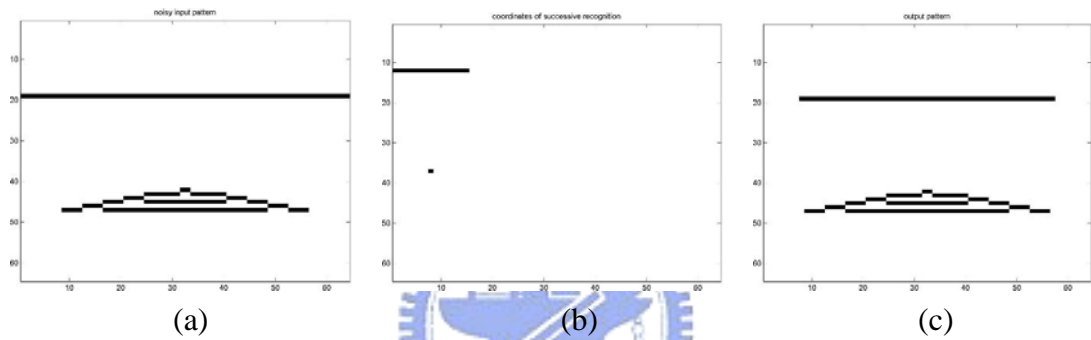


Fig. 4.38 (a) The second real seismic pattern; (b) Coordinates of successful recognition; (c) Output of (a).

$r = 3$ :

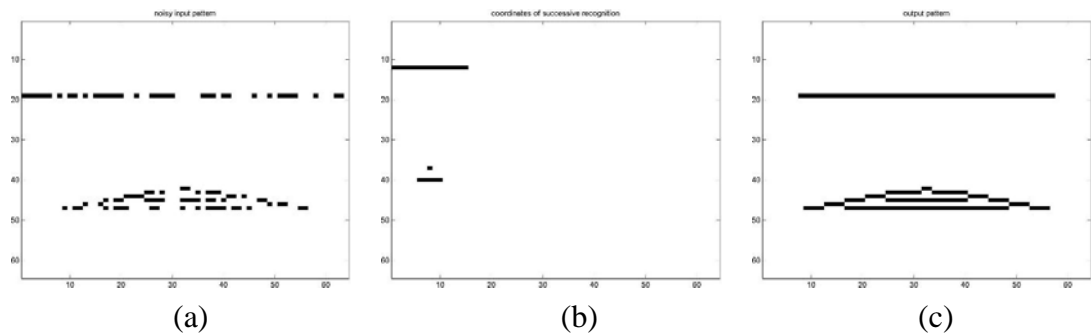


Fig. 4.39 (a) The third real seismic pattern; (b) Coordinates of successful recognition; (c) Output of (a).

## Experiment 7.

Pattern size	$4 \times 4$
The number of training patterns	2
Hamming distance(HD)	3
Neighborhood radius r	1, 2
$\alpha$	3

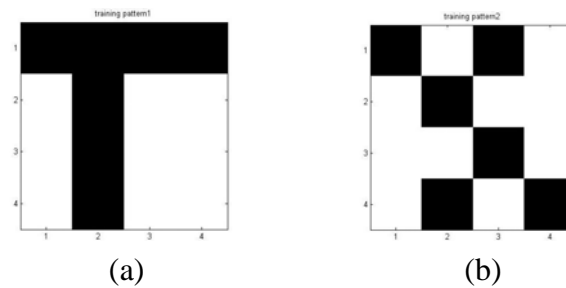


Fig. 4.40 Two training patterns.

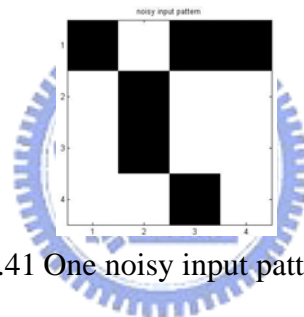


Fig. 4.41 One noisy input pattern.

$r = 1$ :

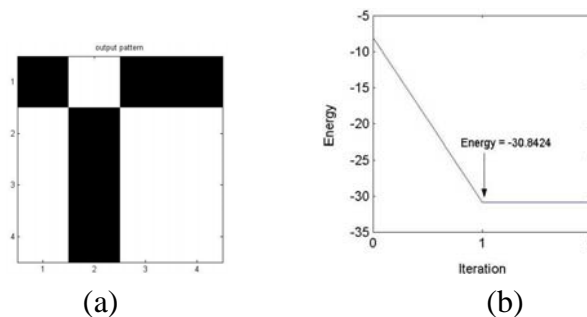


Fig. 4.42 (a) Output of Fig. 4.41 if  $r = 1$ ; (b) Energy curve of Fig. 4.41 if  $r = 1$ .

When  $r = 1$ , the output pattern shown in Fig. 4.42 is not a correct pattern. The pixel on the 1st row and the 2nd column should be a black pixel. Namely the output pattern should be equal to the training pattern 1 shown in Fig. 4.40(a). The pixel on the 1st row and the 2nd column is an indefinite cell. The definition of indefinite cells is explained after. The problem of indefinite cells can be solved by increasing neighborhood radius. So we increase neighborhood radius  $r$  to 2 and try again.

$r = 2$ :

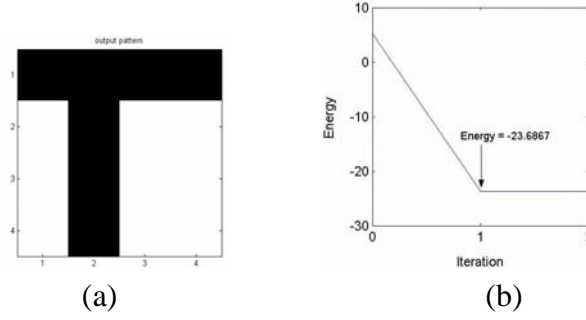


Fig. 4.43 (a) Output of Fig. 4.41 if  $r = 2$ ; (b) Energy curve of Fig. 4.41 if  $r = 2$ .

### Indefinite Cell

If more than two similar patterns are stored, the conventional synthesis procedure may generate the cells with only self-input from these patterns, which are so-called indefinite cells. As the outputs of indefinite cells are determined by only their inputs, DT-CNN can't always associate the expected memory pattern.

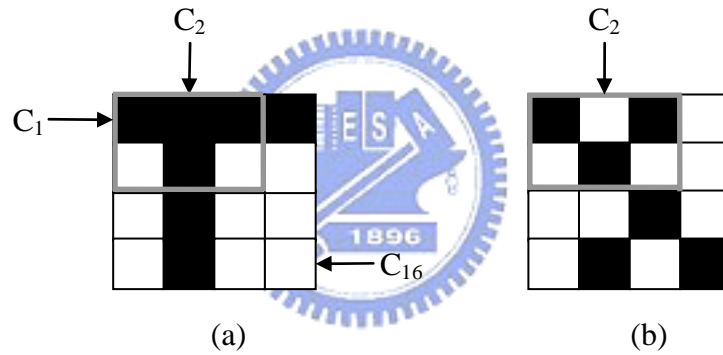


Fig. 4.44 Example of the indefinite cell.

Two patterns shown in Fig. 4.44 are overall different forms each other [21]. But, it is found that the  $r (=1)$ -neighborhood of  $C_2$  in Fig. 4.44(a) is the same pattern as one of  $C_2$  in Fig. 4.44(b) except  $C_2$ . Therefore, the conventional DT-CNN associative memory can't determine the output of  $C_2$  from its  $r$ -neighborhood.  $C_2$  is called an indefinite cell. We know the following equations for every cell.

$$\dot{x}_i = \frac{dx_i(t)}{dt} = g(x_i) + w_i = h(x_i; w_i)$$

where

$$g(x_i) = -x_i + a_i f(x_i) = \begin{cases} -x_i + a_i, & x_i \geq 1 \\ (a_i - 1)x_i, & |x_i| < 1 \\ -x_i - a_i, & x_i \leq -1 \end{cases}$$

$$w_i = I + \sum_{C_j \in S_i, j \neq i} a_j f(x_j) + \sum_{C_j \in S_i} b_j u_j$$

We observe  $w_i$  of  $C_2$  for two patterns in Fig. 4.44 and a testing pattern. For the pattern in Fig. 4.44, we note the  $w_i$  of (a) as  $w_2^1$  and the  $w_i$  of (b) as  $w_2^2$ . And we note the  $w_i$  of testing pattern as  $w_2^3$ . We plot the dynamic routes of the cell  $C_2$  of two patterns in Fig. 4.44 as Fig. 4.45.

$$w_2^i = I + \sum_{C_j \in S_2, j \neq 2} a_j f(x_j) + \sum_{C_j \in S_2} b_j u_j^i, i = 1, 2, 3$$

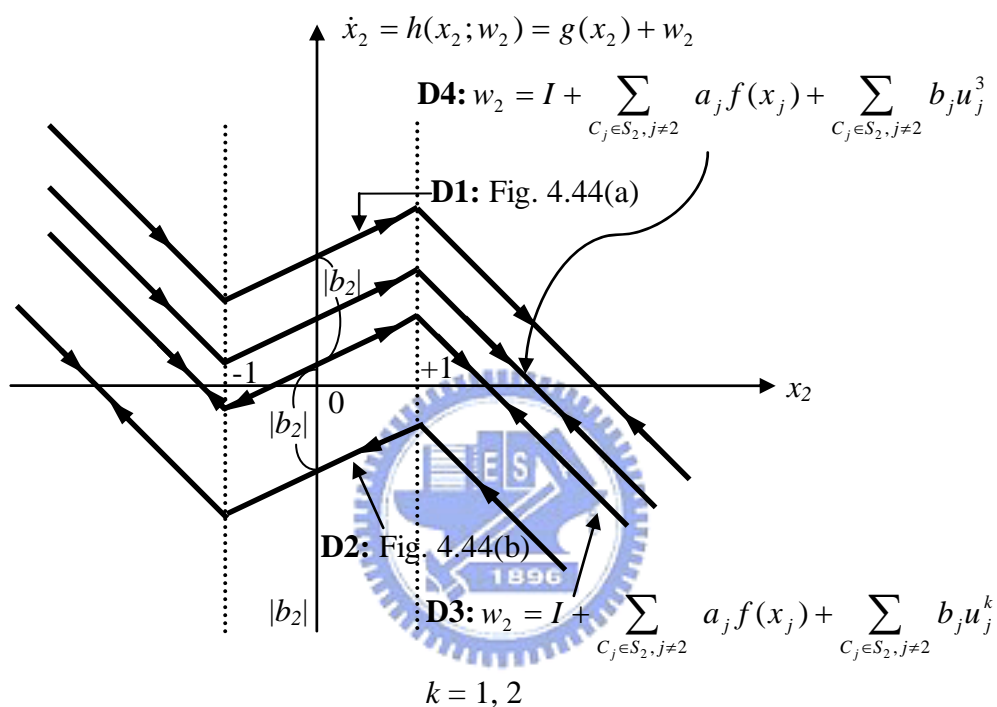


Fig. 4.45 The dynamic routes of cell  $C_2$ .

$$\begin{aligned} |w_2^3 - w_2^k| &= \left| I + \sum_{C_j \in S_2, j \neq 2} a_j f(x_j) + \sum_{C_j \in S_2} b_j u_j^3 - I - \sum_{C_j \in S_2, j \neq 2} a_j f(x_j) - \sum_{C_j \in S_2} b_j u_j^k \right| \\ &= \left| \sum_{C_j \in S_2} b_j u_j^3 - \sum_{C_j \in S_2} b_j u_j^k \right| \\ &= \left| \sum_{C_j \in S_2} b_j (u_j^3 - u_j^k) \right| \\ &= \left| \sum_{C_j \in S_2, j \neq 2} b_j (u_j^3 - u_j^k) + b_2 (u_2^3 - u_2^k) \right| \end{aligned}$$

$$k = 1, 2$$

Because the Hamming distance between the testing pattern and the two patterns in Fig. 4.44 can not be too large, so  $(u_j^3, C_j \in S_2, j \neq 2)$  are almost equal to  $(u_j^k, C_j \in S_2, j \neq 2, k = 1 \text{ or } 2)$ . Therefore  $\sum_{C_j \in S_2, j \neq 2} b_j(u_j^3 - u_j^k)$  is small, its absolute value should be

less than  $|b_2|$ .  $b_2(u_2^3 - u_2^k)$  is equal to 0 or  $\pm 2b_2$ .  
 $\left| \sum_{C_j \in S_2, j \neq 2} b_j(u_j^3 - u_j^k) \right| < \left| \sum_{C_j \in S_2, j \neq 2} b_j(u_j^3 - u_j^k) \pm 2b_2 \right|$ . So if  $u_2^3 = 1$ , then

$$|w_2^3 - w_2^1| = \left| \sum_{C_j \in S_2, j \neq 2} b_j(u_j^3 - u_j^k) \right| \quad \text{and} \quad |w_2^3 - w_2^2| = \left| \sum_{C_j \in S_2, j \neq 2} b_j(u_j^3 - u_j^k) + 2b_2 \right|,$$

$|w_2^3 - w_2^1| < |w_2^3 - w_2^2|$ . Therefore  $x_2$  will converge to a value greater than +1, namely the output will converge to +1. If  $u_2^3 = -1$ , then

$$|w_2^3 - w_2^1| = \left| \sum_{C_j \in S_2, j \neq 2} b_j(u_j^3 - u_j^k) - 2b_2 \right| \quad \text{and} \quad |w_2^3 - w_2^2| = \left| \sum_{C_j \in S_2, j \neq 2} b_j(u_j^3 - u_j^k) \right|,$$

$|w_2^3 - w_2^1| > |w_2^3 - w_2^2|$ . Therefore  $x_2$  will converge to a value less than -1, namely the output will converge to -1. So the outputs of indefinite cells are determined by only their input values.

Therefore, if such memory patterns exist, then the conventional DT-CNN associative memory has the indefinite cells and can't determine the outputs of indefinite cells from their r-neighborhoods.

T. Kamio and H. Asai proposed a DT-CNN system to overcome this problem [21]. The DT-CNN system for associative memory uses two-dimensional discrete Walsh transform. This DT-CNN system consists of two kinds of DT-CNN associative memories and two-dimensional DWT processor as shown in Fig. 4.46 [21]. One of the DT-CNNs is the conventional DT-CNN and the other is the DT-CNN with multilevel threshold function. The former works for the bipolar memory patterns and the latter remembers the 2D-DWT of these bipolar memory patterns.

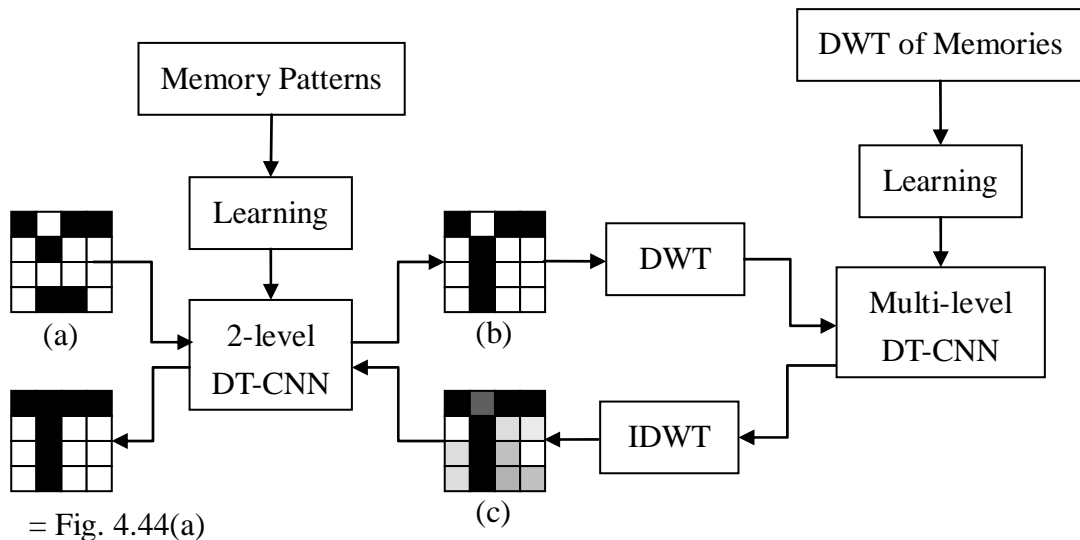


Fig. 4.46 DT-CNN for associative memory using two-dimensional discrete Walsh transform.

As a simple example, it is supposed that two DT-CNNs have the  $r = 1$  neighborhood and the system has two memory patterns shown in Fig. 4.44. If the noisy pattern shown in Fig. 4.46(a) is input to 2-level DT-CNN, then Fig. 4.46(b) is output. As a result, it is found that 2-level DT-CNN fails to associate the desired memory pattern (Fig. 4.44(a)) due to the indefinite cell  $C_2$ . Furthermore, if the DWT of Fig. 4.46(b) is input to the multilevel DT-CNN, then the output in Fig. 4.46(c) is obtained as the IDWT of the image from the multilevel DT-CNN. Fig. 4.46(c) shows that the output value of indefinite cell  $C_2$  is close to the correct value (+1). Finally, 2-level DT-CNN can completely derive the desired pattern shown in Fig. 4.44(a) from Fig. 4.46(c).

The other method that overcomes this problem is increasing the neighborhood radius of template **B**. It makes the neighborhood size larger, and then the same neighborhood patterns originally will be not the same. We observe the two training patterns in Fig. 4.44. If neighborhood radius  $r$  is increased to 2, then the neighborhood of  $C_2$  in Fig. 4.44(a) is not the same pattern as the neighborhood of  $C_2$  in Fig. 4.44(b), as Fig. 4.47 shows.

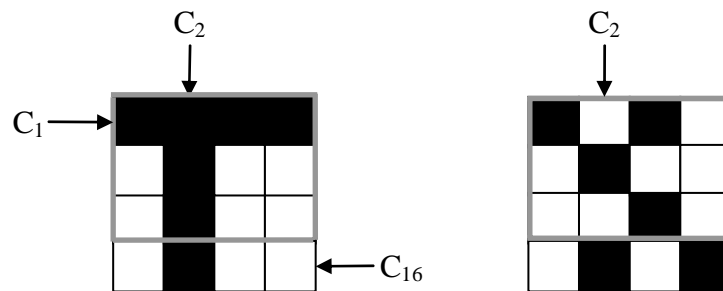


Fig. 4.47 If  $r = 2$ , the neighborhoods of  $C_2$  in two training patterns.

So  $C_2$  in Fig. 4.47 is not the indefinite cell.

The process of the above method is recognizing input pattern first. If the output pattern is different to all memory patterns and that is caused by some indeterminate cells, then increase neighborhood radius and recognize input pattern again until the output pattern is correct. The computation time of this method is much. M. Namba, S. Takatori, H. Kawabata and Z. Zhang proposed an algorithm which can reduce the computation time [22]. This algorithm searches indeterminate cells by matching every memory pattern in advance, and modifies the neighborhood of indeterminate cells. This algorithm can be shown as Fig. 4.48 [22].

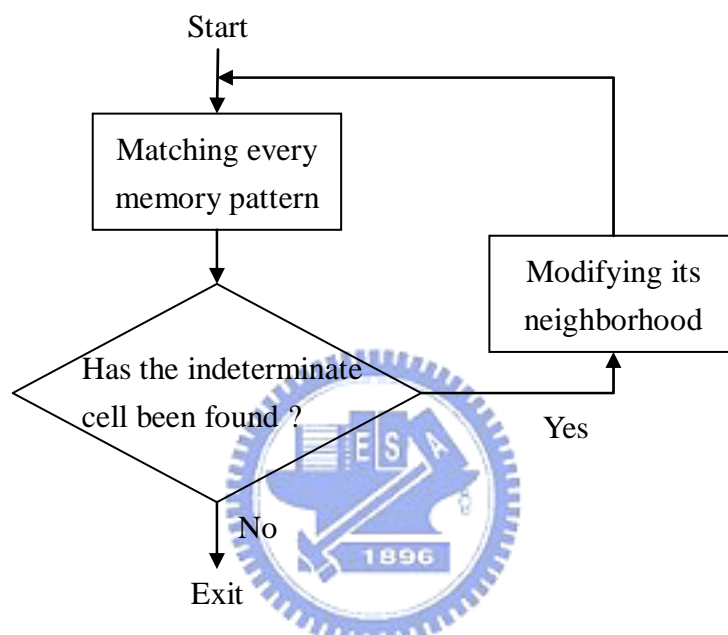


Fig. 4.48 The algorithm searching indeterminate cells.

## 4.7 Conclusions

In this chapter an application of autoassociative memories using DT-CNNs has been proposed. If each element of the 1-D space-invariant template is equivalent, then matrix  $\mathbf{A}$  is symmetric. The symmetry of the interconnection matrix  $\mathbf{A}$  assures that the synthesized network does not oscillate nor become chaotic. Moreover, all the stored patterns correspond to asymptotically stable equilibrium points.



## References

- [1] J. Anderson, "A Memory Storage Model Utilizing Spatial Correlation Functions", *Kybernetik*, 5, 113-119, 1986.
- [2] Amari, "Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements", *IEEE Trans. on Computers*, c 21, 1197-206, 1972.
- [3] T. Kohonen, "Correlation matrix memories", *IEEE Trans. Comput.*, vol. c 21, no. 4, pp. 353-359, April 1972.
- [4] B. Kosko, "Adaptive Bidirectional Associative Memories", *Applied Optics*, vol. 26, no. 23, Dec. 1987.
- [5] D. Liu and A. N. Michel, "Sparsely interconnected neural networks for associative memories with applications to cellular neural networks," *IEEE Trans. Circuits Syst. II*, vol. 41, pp. 295-307, Apr. 1994.
- [6] A. N. Michel and J. A. Farrell, "Associative memories via artificial neural networks," *IEEE Contr. Syst. Mag.*, vol. 10, pp. 6-17, Apr. 1990.
- [7] G. Seiler, A. J. Schuler, and J. A. Nossek, "Design of robust cellular neural networks," *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 358-364, May 1993.
- [8] G. Grassi, "A new approach to design cellular neural networks for associative memories," *IEEE Trans. Circuits Syst. I*, vol. 44, pp. 835-838, Sept. 1997.
- [9] G. Grassi, "On discrete-time cellular neural networks for associative memories," *IEEE Trans. Circuits Syst. I*, vol. 48, pp. 107-111, Jan. 2001.
- [10] J. Park, H.-Y. Kim, Y. Park, and S.-W. Lee, "A synthesis procedure for associative memories based on space-varying cellular neural networks," *Neural Networks*, vol. 14, no. 1, pp. 107-113, Jan. 2001.
- [11] S. Boyd and L. El Ghaoui, "Method of centers for minimizing generalized eigenvalues," *Linear Alg. Applicat.*, vol. 188, pp. 63-111, 1993.
- [12] Y. E. Yu. E. Nesterov and A. S. Nemirovskii, "An interior-point method for generalized linear-fractional programming," *Math. Prog.*, vol. 69, pp. 177-204, 1995.
- [13] L. Vandenberghe and V. Balakrishnan, "Algorithms and software for LMI problems in control," *IEEE Contr. Syst. Mag.*, vol. 17, pp. 89-95, Oct. 1997.
- [14] Liu, D., and Lu, Z., "A new synthesis approach for feedback neural networks based on the perceptron training algorithm," *IEEE Trans. Neural Networks*, 8, 1468-1482.
- [15] Chan, H. Y., and Zak, S. H., "Real-time synthesis of sparsely interconnected neural associative memories," *Neural Networks*, 11, 749-759.
- [16] Perfetti, R., "Dual-mode space-varying self-designing cellular neural networks

- for associative memory,” *IEEE Trans. Circuits Syst. I*, 46, 1281–1285.
- [17] R. Perfetti, “Frequency domain stability criteria for cellular neural networks,” *Int. J. Circuit Theory Appl.*, vol. 25, no. 1, pp. 55–68, 1997.
- [18] S. Arik, A. Kilinc and F. A. Savaci, “Global Asymptotic Stability of Discrete-Time Cellular Neural Networks,” *1998 Fifth IEEE International Workshop on Cellular Neural Networks and their Applications*, London, England, 1998, pp.52-55.
- [19] C. Guzelis and L. O. Chua, “Stability analysis of generalized cellular neural networks”, *Int. J. of Circ. Theory and Appl.*, 21, 1-33 (1993).
- [20] L. O. Chua and Lin Yang, “Cellular Neural Networks: Theory,” *IEEE Trans. on CAS*, vol. 35 no. 10, pp.1257-1272, 1988.
- [21] T. Kamio and H. Asai, “Improvement of Discrete-Time Cellular Neural Networks for Associative Memory Using 2-Dimensional Discrete Walsh Transform,” *Cellular Neural Networks and Their Applications Proceedings, 1998 Fifth IEEE International Workshop on*, 14-17 April 1998, Page(s):416 - 421.
- [22] M. Namba, S. Takatori, H. Kawabata and Z. Zhang, “The Variable Neighborhood CNN,” *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, Volume 3, 6-9 May 2001, Page(s):105 - 108 vol. 2.

