Neural, Parallel & Scientific Computations 4 (1996) 305-324

Theory and Applications of Sparsely Interconnected Feedback Neural Networks

Anthony N. Michel[1] and Derong Liu[2]

1. Department of Electrical Engineering
University of Notre Dame, Notre Dame, IN 46556
Email: Anthony.N.Michel@nd.edu
2. Department of Electrical Engineering and Computer Science
Stevens Institute of Technology, Hoboken, NJ 07030
Email: dliu@stevens-tech.edu

ABSTRACT: This paper presents some recent developments in the analysis and design of a class of feedback neural networks with sparse interconnecting structure. The analysis results presented make it possible to determine whether a given vector is a stable memory of a neural network and to what extent implementation errors are permissible. The design methods presented allow the synthesis of neural networks with predetermined sparse interconnecting structures with or without symmetry constraints on the interconnection weights. Two examples are included to demonstrate the applicability of the methodology advanced herein.

KEYWORDS - associative memories, sparsely interconnected neural networks, robustness analysis, pattern recognition

1 INTRODUCTION

The two basic categories of artificial neural networks encountered in the literature are feedforward and feedback neural nets. In the former, neurons are arranged on several layers and unidirectional (feedforward) connections are used between neurons of adjacent layers. In the latter, all neurons are arranged in a single layer and bidirectional connections (*i.e.*, connections from neuron i to neuron j and from neuron j to neuron i, where $i \neq j$) are used between various pairs of neurons in the network. In feedback neural networks, every neuron of the network is usually connected to all other neurons (including to itself), *i.e.*, neural nets are usually *fully connected.* Single-layer, fully connected neural networks are a special class of nonlinear dynamical systems which are endowed with many asymptotically stable equilibrium points (stable memories) as well as unstable equilibria. The study of

Received June 19, 1996

such systems has been of great interest to many researchers in recent years (see, e.g., [1], [6]–[15], [18]–[22]). These works are concerned with the qualitative analysis of neural networks (the existence and locations of equilibrium points, the qualitative properties of the equilibria, and the extent of the basins of attraction of asymptotically stable equilibria. (see, e.g., [1], [6], [7], [9]–[15], [18], [22])), and design methodologies for such networks (including *the outer product method* [12], *the projection learning rule* [20], [21], *the eigenstructure method* [15], and other methods [8], [19]). The study of single-layer, *sparsely interconnected* neural networks has also been of recent interest [2]–[5], [16], [17]. In this paper, we will present some analysis and synthesis results for a class of sparsely interconnected feedback neural networks.

We will concern ourselves primarily with the implementation of *associative memories* by means of a class of feedback neural networks. To this end, we will first introduce a result concerning the qualitative properties of the class of neural networks considered herein. This result yields conditions which can be used for testing whether a given vector is a stable memory. Next, we address a problem which is unavoidably encountered in hardware implementations of neural networks, implementation inaccuracies or parameter perturbations. This problem is of great practical interest, especially in VLSI implementations of neural networks, since one cannot realize *precisely* designed parameters. A robustness analysis result will be presented to deal with parameter perturbations.

One of the major difficulties encountered in VLSI implementations of fully connected neural networks is the realization of extremely large numbers of interconnections and the reduction of the number of connections in such networks is of great interest from a practical point of view. A desirable solution for solving the problem of interconnection constraints encountered in the hardware implementation of neural networks involves synthesis procedures which incorporate reductions in the number of interconnections. This has been one of the major motivations for considering the study of sparsely interconnected neural networks [16], [17]. In this paper, we will provide a synthesis procedure for such neural networks with sparse connection matrices in which the interconnecting structure is predetermined. Utilizing the robustness analysis result mentioned above, we will be in a position to design neural networks with predetermined interconnecting structure and with *symmetric* interconnections.

We will demonstrate the applicability of the methodology advanced herein by considering two specific examples.

2 THE NEURAL NETWORK MODEL CONSIDERED AND ASSOCIATIVE MEMORIES

We consider a class of neural networks described by equations of the form

$$\begin{cases} \dot{x} = -Ax + T\mathrm{sat}(x) + I \\ y = \mathrm{sat}(x) \end{cases} \tag{2.1}$$

where $x \in R^n$ is the state vector (representing neuron inputs), $\dot{x}$ denotes the derivative of x with respect to time t, $y = [y_1, \cdots, y_n]^T$ with $|y_i| \leq 1$ represents neuron outputs, $A = \text{diag}[a_1, \cdots, a_n]$ is the state transition matrix with $a_i > 0$ for $i = 1, \cdots, n$, $T = [T_{ij}] \in R^{n \times n}$ is the connection (or weight) matrix, $I = [I_1, \cdots, I_n]^T \in R^n$ is a (constant) bias vector, and $\text{sat}(x) = [\text{sat}(x_1), \cdots, \text{sat}(x_n)]^T$ represents the activation function (the neuron model), where

$$\text{sat}(x_i) = \begin{cases} 1, & x_i > 1 \\ x_i, & -1 \leq x_i \leq 1 \\ -1, & x_i < -1 \end{cases}$$

We assume that the initial states of (2.1) satisfy $|x_i(0)| \leq 1$ for $i = 1, \cdots, n$.

In the hardware implementation of artificial neural networks described by (2.1) (e.g., in VLSI implementations), one cannot realize *precisely* the synthesized parameters $\{A, T, I\}$. A bound for permissible parameter perturbations which guarantees to retain the desired performance of the network is therefore of great interest from a practical point of view. We will use a perturbation model of system (1) given by

$$\begin{cases} \dot{x} = -(A + \Delta A)x + (T + \Delta T)\text{sat}(x) + (I + \Delta I) \\ y = \text{sat}(x) \end{cases} \tag{2.2}$$

where A, T, I, and $\text{sat}(\cdot)$ are defined as in (1), $\Delta A = \text{diag}[\Delta a_1, \cdots, \Delta a_n]$ with $a_i + \Delta a_i > 0$ for $i = 1, \cdots, n$, $\Delta T \in R^{n \times n}$, and $\Delta I \in R^n$.

In this paper, we consider the implementation of *associative memories* via neural networks modeled by (2.1). The goal of associative memories is to store a set of desired patterns as memories such that a stored pattern can be recognized when the input pattern contains sufficient information about that pattern.

A vector α is said to be a *memory vector* (or simply, a *memory*) of system (2.1), if $\alpha = \text{sat}(\beta)$ and if β is an asymptotically stable equilibrium point of system (2.1). (Recall that an equilibrium x_e of system (1) is asymptotically stable if (i) it is *stable* in the sense of Lyapunov, *i.e.*, the state $x(t)$ of system (1) remains arbitrarily close to x_e for all $t \geq 0$ whenever the initial state $x(0)$ is sufficiently close to x_e, and (ii) $x(t)$ approaches x_e as t tends to infinity whenever $x(0)$ belongs to $B(x_e)$, where $B(x_e)$ is an open neighborhood of x_e in R^n.) In practice the desired memory patterns are usually represented by bipolar vectors (or binary vectors). We will not consider the case where desired memory patterns are not bipolar vectors.

Equation (1) has the same form as the *cellular neural network* model employed in [5] in which $n = M \times N$ neurons are arranged in an $M \times N$ array and only local interconnections are used. Sparsity constraints on the interconnecting structure of (1), in general, are expressed as prespecified zero elements in the connection matrix T at given locations and therefore, the cellular neural network model in [5] is a special case of the present neural network model. For other feedback neural network models such as the Hopfield model [12], neural networks described on hypercubes [15], and iterated-map neural networks [18], similar sparsity constraints

can be developed. We emphasize that (1) is used as an example in the present paper to demonstrate how results can be developed for neural networks with sparse interconnecting structures and that similar procedures can be used in the synthesis of neural nets with sparsity constraints involving the other models mentioned above. We note, however, that the realization of associative memories is only one of many possible applications of feedback neural networks for which interconnection constraints are of interest.

3 ANALYSIS

In this section, we first present conditions for testing whether a given bipolar vector is an asymptotically stable equilibrium point of system (1), and conditions under which neural networks are globally stable. We then present a result for the robustness analysis of (1).

3.1 Basic Analysis

We use B^n to denote the set of n-dimensional *bipolar vectors*, *i.e.*, $B^n \triangleq \{x \in \Re^n\colon\ x_i = 1 \text{ or } -1,\ i = 1,\cdots,n\}$. For $\alpha \in B^n$, we define $C(\alpha) = \{x \in \Re^n\colon\ x_i\alpha_i > 1,\ i = 1,\cdots,n\}$.

A result proved in [16] states that if $\alpha \in B^n$ and if

$$\beta = A^{-1}(T\alpha + I) \in C(\alpha) \tag{3.1}$$

then (α, β) is a pair of memory vector and asymptotically stable equilibrium point of (2.1). The proof of this result uses the fact that $\text{sat}(\gamma) = \alpha \in B^n$ for all $\gamma \in C(\alpha)$. For $x \in C(\alpha)$, the first equation of (2.1) can be written as

$$\dot{x} = -Ax + T\alpha + I \tag{3.2}$$

System (3.2) has a unique equilibrium at $x_e = A^{-1}(T\alpha + I)$, and $x_e = \beta \in C(\alpha)$ by assumption. Clearly, this equilibrium is also asymptotically stable, since in system (3.2) all eigenvalues $\lambda_i(-A)$ of $-A$ are negative (since $\lambda_i(A) = a_i > 0$).

This result constitutes the basis of our synthesis procedures to be presented in the next section. In our design problem, we will be given a set of desired bipolar patterns to be stored as memory vectors and our objective will be to determine A, T, and I so that (3.1) is true for every desired pattern.

In the study of global stability of neural networks in the form (1), we usually assume that T is symmetric. Recall that a neural network is globally stable if every initial state of the network converges to some asymptotically stable equilibrium. When this is the case, periodic and chaotic motions cannot exist in the network. With system (1), we associate an energy function of the form (see, e.g., [5], [16]),

$$E(y) = -\frac{1}{2}y^T Ty - I^T y + \frac{1}{2}y^T Ay \tag{3.3}$$

where y is defined as in (1). It is easily verified that the function $E(\cdot)$ in (3.3) is monotonically decreasing in time t along the solutions of (1) if T is *symmetric*. This, along with the fact that the set of asymptotically stable equilibria is discrete, can be used to show that the network is globally stable.

3.2 Perturbation Analysis

When implementing a synthesized neural network, one usually assumes that all the computed parameters are realized exactly, so that the network will exhibit a desired performance (such as having desired memory points). It is a fact of life that every implementation process will result in some inaccuracies. When considering such inaccuracies, it is important to have criteria that enable one to determine whether the implemented neural network will perform as expected.

In system (2.2), ΔA, ΔT, and ΔI may be considered as inaccuracies that are incurred during the implementation process and/or as a limited precision implementation of system (1). A possible upper bound for the permissible perturbations ΔA, ΔT, and ΔI in terms of the expression $\|A^{-1}\Delta A\|_\infty + \|A^{-1}\Delta T\|_\infty + \|A^{-1}\Delta I\|_\infty$ can be established, where $\|\cdot\|_\infty$ denotes the matrix norm induced by the l_∞ vector norm. Recall that the matrix norm induced by the l_∞ vector norm for a matrix $F = [f_{ij}] \in R^{m\times n}$ is defined by

$$\|F\|_\infty = \max_{1\le i\le m}\Big\{\sum_{j=1}^{n}|f_{ij}|\Big\}$$

We will make use of the notation $\delta(x) = \min_{1\le i\le n}\{|x_i|\}$ for $x \in R^n$. Suppose that $\alpha^1, \cdots, \alpha^m \in B^n$ are desired memory vectors of system (2.1), and suppose that $\beta^1, \cdots, \beta^m$ are asymptotically stable equilibrium points of system (2.1) corresponding to $\alpha^1, \cdots, \alpha^m$, respectively, *i.e.*, $\beta^i = A^{-1}(T\alpha^i + I)$ for $i = 1, \cdots, m$. Let

$$\nu = \min_{1\le i\le m}\{\delta(\beta^i)\} > 1 \tag{3.4}$$

Then, $\alpha^1, \cdots, \alpha^m$ are also memory vectors of system (2.2) provided that

$$\|A^{-1}\Delta A\|_\infty + \|A^{-1}\Delta T\|_\infty + \|A^{-1}\Delta I\|_\infty < \nu - 1 \tag{3.5}$$

The above robustness analysis result is proved in [17] using the results of Section 3.1.

Suppose that (α, β) is a pair of memory vector and asymptotically stable equilibrium point of (2.1). After perturbation, the new asymptotically stable equilibrium point $\overline{\beta}$ is given by

$$\overline{\beta} = (A + \Delta A)^{-1}[(T + \Delta T)\alpha + (I + \Delta I)] \tag{3.6}$$

When condition (3.5) is satisfied, it can be shown that $\overline{\beta} \in C(\alpha)$, which implies that α is still a memory vector of (2.1) after perturbation.

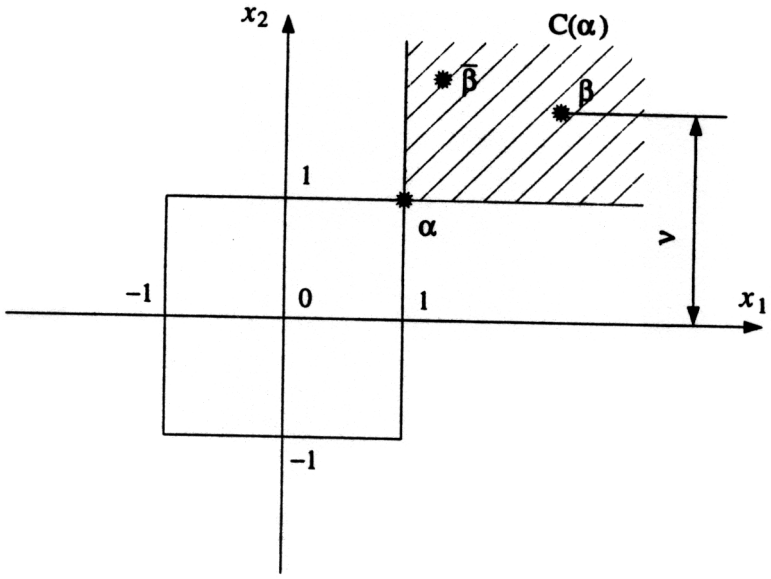

Figure 1: A geometric interpretation of the robustness analysis

To give a geometric interpretation, assume that $\alpha \in R^2$ is a (desired) memory of system (2.1) and its corresponding asymptotically stable equilibrium point is β. Then, $\beta = A^{-1}(T\alpha + I)$ *must* be in the region $C(\alpha)$ (cf. the crosshatched region in Figure 1), since $\nu = \min\{\delta(\beta)\} > 1$.

When we have perturbations ΔA, ΔT, and ΔI as in system (2.2), the vector β will be displaced from its original location to, say, $\overline{\beta}$. In order for α to remain a memory vector for system (2.1) after perturbation (*i.e.*, for α to be a memory vector for system (2.2)), we require that $\overline{\beta}$ also be in $C(\alpha)$. It is clear that as long as $\overline{\beta}$ is in $C(\alpha)$, α will be a memory vector of the perturbed system (2.2). The above robustness analysis result gives one of the *possible* upper bounds for the perturbations, specified by $\|A^{-1}\Delta A\|_\infty + \|A^{-1}\Delta T\|_\infty + \|A^{-1}\Delta I\|_\infty$, which will ensure that the perturbed vector $\overline{\beta}$ and the original vector β are within the same region given by $C(\alpha)$. This upper bound is given by $\nu - 1$ where ν is defined in (3.4).

In system (2.2), we have to require that $a_i + \Delta a_i > 0$ for each i. It is clear that a perturbation ΔA with $\Delta a_i < 0$ for $i = 1, \cdots, n$ will not change the desired memory vectors $\alpha^1, \cdots \alpha^m \in B^n$ of system (2.1) (cf. equation (3.6)).

When considering perturbations due to an implementation process, the focus is usually on the interconnection matrix T and not on the parameters A and I. Assuming $\Delta A = 0$ and $\Delta I = 0$, system (2.2) takes the form

$$\begin{cases} \dot{x} = -Ax + (T + \Delta T)\text{sat}(x) + I \\ y = \text{sat}(x) \end{cases} \tag{3.7}$$

and condition (3.5) assumes the simple form

$$\|A^{-1}\Delta T\|_\infty \leq \nu - 1$$

4 SYNTHESIS METHODS

In applications of neural networks to associative memories, a *desired* set of asymptotically stable equilibria determines a set of neuron output vectors that are used as *stable memories* to store information. The locations of the desired stable memories are determined by choosing the network parameters, given by A, T, and I, in an appropriate manner. We will call this process of selecting $\{A, T, I\}$ synthesis.

4.1 Overview of Existing Synthesis Methods

In the following, we will summarize three synthesis procedures for associative memories realized by neural networks (1).

The Outer Product Method [12]: We wish to store m desired bipolar patterns $\alpha^i \in B^n$, $1 \leq i \leq m$, which correspond to m asymptotically stable equilibria $\beta^i = \alpha^i$ of (1) (therefore, $\alpha^i = \text{sat}(\beta^i)$), as stable memories. A set of parameter choices determined by the Outer Product Method is given by

$$T = \sum_{j=1}^{m} \alpha^j (\alpha^j)^T, \; A = E \text{ and } I = 0$$

where E denotes the $n \times n$ identity matrix. The name of this method is motivated by the fact that T consists of the sum of outer products of the patterns that are to be stored as stable memories. This method requires that the α^i, $1 \leq i \leq m$, be mutually orthogonal (*i.e.*, $(\alpha^i)^T \alpha^j = 0$ when $i \neq j$). ∎

The Projection Learning Rule [20], [21]: When the desired prototype patterns $\alpha^j \in B^n$, $1 \leq j \leq m$, to be stored in (1) as memories are not mutually orthogonal, a method called the Projection Learning Rule can be used to synthesize the interconnection parameters for (1).

Let $\Sigma = [\alpha^1, \cdots, \alpha^m]$. Recall that for $\Sigma \in R^{n \times m}$, the *Moore-Penrose pseudo-inverse* Σ^I: $R^n \to R^m$ defines the linear mapping of any $b \in R^n$ to a unique $x \in R^m$ (*i.e.*, $x = \Sigma^I b$) which has the property that x is the vector that has the smallest Euclidean norm $\|x\|$ on the set $\{\gamma \in R^m : \|\Sigma\gamma - b\|^2$ is minimized $\}$. Then the interconnection matrix T for system (1) is given by

$$T = \Sigma \, \Sigma^I \tag{4.1}$$

(refer, e.g., to [20], [21]). We note that T determined by (4.1) satisfies the relation $T\Sigma = \Sigma$, which shows that T is an orthogonal projection of R^n onto the linear space spanned by α^j, $1 \leq j \leq m$ (hence, the name Projection Rule). It is easily

verified that when the α^j, $1 \leq j \leq m$, are mutually orthogonal, then the Projection Learning Rule and the Outer Product Method coincide. The Projection Learning Rule does not guarantee that an equilibrium of (1) corresponding to a given desired memory is asymptotically stable. ∎

The Eigenstructure Method [15]: Neural networks which are synthesized by this method are guaranteed to store desired sets of patterns as stable memories which need not be mutually orthogonal and which correspond to asymptotically stable equilibria of (1). Suppose that we are given a set of desired patterns $\alpha^1, \cdots, \alpha^m \in B^n$. We wish to design a system of form (2.1) which stores $\alpha^1, \cdots, \alpha^m$ as memories. Without loss of generality, we choose A as the $n \times n$ identity matrix and we choose $\beta^l = \mu\alpha^l$, for $l = 1, \cdots, m$, with $\mu > 1$ (hence, $\beta^l \in C(\alpha^l)$). T and I will be determined by the relations

$$A\beta^l = \mu\alpha^l = T\alpha^l + I, \; l = 1, \cdots, m$$

Solutions of (4.2) for T and I will always exist. To see this, we let $Y = [\alpha^1 - \alpha^m, \cdots, \alpha^{m-1} - \alpha^m]$. We need to solve T from

$$TY = \mu Y$$

and set $I = \mu\alpha^m - T\alpha^m$. This will guarantee that system (2.1) stores the desired patterns $\alpha^1, \cdots, \alpha^m$ as memories and that it will store $\beta^1, \cdots, \beta^m$ as corresponding asymptotically stable equilibrium points. Solutions of (4.3) (for T) always exist since

$$\text{rank}[Y] = \text{rank}\begin{bmatrix} Y \\ \cdots \\ \mu Y \end{bmatrix}$$

T can be determined from (4.3) using the *singular value decomposition method* (see, e.g., [15]) as follows. Performing a singular value decomposition ot Y, we obtain

$$Y = [U_1 \vdots U_2]\begin{bmatrix} D & \vdots & 0 \\ \cdots & \vdots & \cdots \\ 0 & \vdots & 0 \end{bmatrix}\begin{bmatrix} V_1^T \\ \cdots \\ V_2^T \end{bmatrix}$$

where $D \in R^{p \times p}$ is a diagonal matrix with the nonzero singular values of matrix Y on its diagonal and $p = \text{rank}[Y]$. Then,

$$T = \mu U_1 U_1^T + W U_2^T$$

where W is an arbitrary $n \times (n - p)$ real matrix. We note that by special choice of matrix W, e.g., $W = \tau U_2$ with τ a scalar, (4.5) can result in a symmetric matrix T [16].

The above steps constitute a design procedure of neural networks with *no constraints* on the interconnecting structure. This procedure usually results in a *fully connected* neural network. The consequence of the above design is that $\alpha^1, \cdots, \alpha^m$

will be stored as stable memory vectors in system (2.1), that the states β^i corresponding to α^i, $i = 1, \cdots, m$, will be asymptotically stable equilibrium points of system (2.1), and that all vectors in $L_\alpha \cap B^n$, including $\alpha^1, \cdots, \alpha^m$, will be stored as memory vectors of system (2.1), where $L_\alpha = \text{Aspan}(\alpha^1, \cdots, \alpha^m) \triangleq \text{Span}(\alpha^1 - \alpha^m, \cdots, \alpha^{m-1} - \alpha^m) + \alpha^m$ and $\text{Span}(\gamma^1, \cdots, \gamma^n)$ denotes the linear subspace of R^n generated by $\gamma^1, \cdots, \gamma^n$. ∎

4.2 Synthesis of Sparsely Interconnected Neural Networks

Based on the eigenstructure method summarized above, we provide in the following a design procedure for artificial neural networks which have predetermined interconnecting structures, and which do not require that the interconnection matrix be symmetric. In the next subsection, we will consider symmetry constraints on the interconnection matrix.

We begin by introducing some necessary terminology.

A matrix $S = [S_{ij}] \in R^{n \times n}$ is said to be an *index matrix,* if it satisfies $S_{ij} = 1$ or 0. The restriction of matrix $W = [W_{ij}] \in R^{n \times n}$ to an index matrix S, denoted by $W|S$, is defined by $W|S = [h_{ij}]$, where

$$h_{ij} = \begin{cases} W_{ij}, & \text{if } S_{ij} = 1 \\ 0, & \text{otherwise} \end{cases}$$

We will say that (2.1) is a *neural network with a sparse coefficient matrix* if $T = T|S$ for some given index matrix S. Thus, the index matrix S specifies a (sparse) interconnecting structure for a neural network.

Sparse Design Problem: Given an $n \times n$ index matrix $S = [S_{ij}]$ with $S_{ii} \neq 0$ for $i = 1, \cdots, n$, and m vectors $\alpha^1, \cdots, \alpha^m$ in B^n, choose $\{A, T, I\}$ with $T = T|S$ in such a manner that $\alpha^1, \cdots, \alpha^m$ are memory vectors of system (2.1). ∎

A solution for the above sparse design problem is as follows (cf. [16]).

Sparse Design Procedure: Suppose we are given an $n \times n$ index matrix $S = [S_{ij}]$ with $S_{ii} \neq 0$ for $i = 1, \cdots, n$, and m vectors $\alpha^1, \cdots, \alpha^m$ in B^n which are to be stored as memory vectors for (2.1). We proceed as follows:

1) Choose matrix A as the identity matrix.

2) Choose a real number $\mu > 1$ and set $\beta^i = \mu\alpha^i$ for $i = 1, \cdots, m$.

3) Compute the $n \times (m-1)$ matrices $Y = [y^1, \cdots, y^{m-1}] = [\alpha^1 - \alpha^m, \cdots, \alpha^{m-1} - \alpha^m]$, and $Z = [z^1, \cdots, z^{m-1}] = [\beta^1 - \beta^m, \cdots, \beta^{m-1} - \beta^m]$. We let $y^i = [y_1^i, \cdots, y_n^i]^T$ and $z^i = [z_1^i, \cdots, z_n^i]^T$ for $i = 1, \cdots, m-1$.

4) Denote the i^{th} row of the index matrix S by $S_i = [S_{i1}, \cdots, S_{in}]$. For each $i = 1, \cdots, n$, construct two sets M_i and N_i, such that $M_i \cup N_i = \{1, \cdots, n\}$, $M_i \cap N_i = \phi$, and $S_{ij} = 1$ if $j \in M_i$, $S_{ij} = 0$ if $j \in N_i$. Let $M_i = \{\sigma_i(1), \cdots, \sigma_i(m_i)\}$, where $m_i = \sum_{j=1}^n S_{ij}$ and $\sigma_i(k) < \sigma_i(l)$ if $1 \leq k < l \leq m_i$. (Note that m_i is the number of nonzero elements in the i^{th} row of matrix S.)

5) For $i = 1, \cdots, n$, and $l = 1, \cdots, m-1$, let $y_{Ii}^l = [y_{\sigma(1)}^l, \cdots, y_{\sigma(m_i)}^l]^T$.

6) For $i = 1, \cdots, n$, compute the $m_i \times (m-1)$ matrices $Y_i = [y_{Ii}^1, \cdots, y_{Ii}^{m-1}]$, and the $1 \times (m-1)$ vectors $Z_i = [z_i^1, \cdots, z_i^{m-1}]$.

7) For $i = 1, \cdots, n$, perform singular value decompositions of Y_i, and obtain

$$Y_i = [U_{i1} \vdots U_{i2}] \begin{bmatrix} D_i & \vdots & 0 \\ \cdots & \vdots & \cdots \\ 0 & \vdots & 0 \end{bmatrix} \begin{bmatrix} V_{i1}^T \\ \cdots \\ V_{i2}^T \end{bmatrix}$$

where $D_i \in R^{p_i \times p_i}$ is a diagonal matrix with the nonzero singular values of Y_i on its diagonal and $p_i = \text{rank}(Y_i)$.

8) Compute for $i = 1, \cdots, n$, $G_i = [G_{i1}, \cdots, G_{im_i}] = Z_i V_{i1} D_i^{-1} U_{i1}^T + W_i U_{i2}^T$, where W_i is an arbitrary $1 \times (m_i - p_i)$ real vector.

9) The matrix $T = [T_{ij}]$ is computed as follows:

$$T_{ij} = \begin{cases} 0, & \text{if } S_{ij} = 0 \\ G_{ik}, & \text{if } S_{ij} \neq 0 \text{ and if } j = \sigma_i(k) \end{cases} \tag{4.6}$$

10) Compute the bias vector $I = \beta^m - T\alpha^m$.

Then, $\alpha^1, \cdots, \alpha^m$ will be stored as memory vectors for system (2.1) with A, T, and I determined as above. The states β^i corresponding to α^i, $i = 1, \cdots, m$, will be asymptotically stable equilibrium points of the synthesized system. ∎

The idea used in the above sparse design procedure is to solve the matrix T on a row-by-row basis. The following observations pertain to the above design procedure: (i) solutions for the sparse design problem always exist if $S_{ii} = 1$ for $i = 1, \cdots, n$; (ii) the Sparse Design Procedure guarantees that $T = T|S$; and (iii) the Sparse Design Procedure guarantees that all vectors in $L_\alpha \cap B^n$, including $\alpha^1, \cdots, \alpha^m$, are stored as stable memory vectors of system (2.1).

It is readily seen from the robustness analysis result of Section 3.2 that the synthesis procedures presented above guarantee that $\alpha^1, \cdots, \alpha^m$ are also memory vectors of system (2) provided that (refer to step 2 above and equations (3.4) and (3.5))

$$\|A^{-1}\Delta A\|_\infty + \|A^{-1}\Delta T\|_\infty + \|A^{-1}\Delta I\|_\infty = \|\Delta A\|_\infty + \|\Delta T\|_\infty + \|\Delta I\|_\infty < \mu - 1 \tag{4.7}$$

The above enables us to specify *an upper bound* for the parameter inaccuracies encountered in the implementation of a given network design to store a desired set of bipolar patterns in system (1). This bound is chosen by the designer during the initial phase of the design procedure. This type of flexibility does not appear to have been achieved before (e.g., [8]–[15], [19]–[21]). Specifically, the synthesis procedure advocated above incorporates two features which are very important in the VLSI implementation of artificial neural networks: (i) it allows the VLSI designer to choose a suitable interconnecting structure for the neural network; and (ii) it takes

into account inaccuracies which arise in the realization of the neural network by hardware.

Modified Sparse Design Procedure: If we wish that the above design procedure results in a system of form (2.1) with $I = 0$, we can modify the Sparse Design Procedure as follows:

a) In step 3, take $Y = [\alpha^1, \cdots, \alpha^m]$ and $Z = [\beta^1, \cdots, \beta^m]$.

b) In step 10, take $I = 0$.

Then all conclusions will still be true. In particular, all vectors in $\text{Span}(\alpha^1, \cdots, \alpha^m) \cap B^n$ including $\pm\alpha^1, \cdots, \pm\alpha^m$ are stored as memory vectors of the synthesized system (2.1) [16]. ∎

4.3 Synthesis of Neural Networks with Sparsity and Symmetry Constraints

In this subsection, a synthesis procedure for associative memories which results in *sparse and symmetric* interconnection matrices T for system (2.1) will be introduced.

For the A, T, and I determined by the Sparse Design Procedure with $\mu > 1$, let us choose

$$\Delta T = (T^T - T)/2 \tag{4.8}$$

Then, $T_s \triangleq T + \Delta T = (T + T^T)/2$ is a symmetric matrix. From the robustness analysis result, we note that if

$$\|A^{-1}\Delta T\|_\infty = \|T^T - T\|_\infty/2 < \mu - 1 \tag{4.9}$$

the neural network (3.7) will also store all the desired patterns as memories, with a symmetric connection matrix $T + \Delta T = T_s$.

The above observation gives rise to the possibility of designing a neural network (2.1) with *prespecified interconnection structure* and with *a symmetric interconnection matrix.* (Note that in this case, we require that $S = S^T$.) Such capability is of *great* interest since neural network (2.1) will be *globally stable* when T is symmetric (cf. Section 3.1). It appears that (4.9) might always be satisfied by choosing μ sufficiently large. However, large μ will usually result in large absolute values of the components of T which in turn may result in a large $\|T^T - T\|_\infty$. Therefore, it is not always possible for (4.9) to be satisfied by choosing μ large. From (4.9), we see that if our original synthesized matrix T is sufficiently close to its symmetric part $(T + T^T)/2$, or equivalently, if $\|T^T - T\|_\infty$ is sufficiently small, then (4.9) is satisfied and we are able to design a neural network of form (2.1) with the following properties: (*i*) the network stores $\alpha^1, \cdots, \alpha^m$ as memory vectors; (*ii*) the network has a predetermined (full or sparse) interconnecting structure; and (*iii*) the connection matrix T of the network is symmetric.

An *iterative algorithm* (design procedure) can be utilized to achieve this.

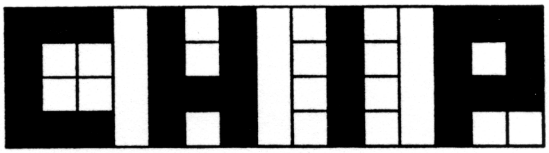

Figure 2: The four desired memory patterns

Symmetric Design Procedure: Let ΔT be defined as in (4.8). For the given $\mu > 1$, suppose that $\|\Delta T\|_\infty \geq \mu - 1$. We can find a λ, $0 < \lambda < 1$, such that $\lambda\|\Delta T\|_\infty < \mu - 1$, and we let $T_1 = T + \lambda\Delta T$. We use this T_1 as the *new* connection matrix for our neural network (2.1). According to the robustness analysis result, we see that $\alpha^1, \cdots, \alpha^m$ are still memory vectors of system (2.1) with coefficient matrix T_1, and we can compute the corresponding asymptotically stable equilibrium points as $\overline{\beta}^l = A^{-1}(T_1\alpha^l + I)$ for $l = 1, \cdots, m$. Clearly $\overline{\beta}^l \in C(\alpha^l)$. We can determine the upper bound ν for the permissible perturbation ΔT as in (3.4) and (3.5), where we use $\overline{\beta}^l$ instead of β^l. We *repeat* the above procedure, until we determine a symmetric coefficient matrix T or until we arrive at $\nu \leq 1 + \eta$ (where η is a small positive number, e.g., $\eta = 0.001$). ∎

If we end up with $T = T^T$, we have found a solution for our symmetric design problem. If we end up with $\nu \leq 1 + \eta$ and $T \neq T^T$, our design procedure is not successful in solving a symmetric T for the given problem. Experimental results indicated that this procedure will frequently succeed in determining a symmetric matrix T.

5 APPLICATIONS

Two specific examples will be considered in this section to demonstrate the applicability of the results presented in the preceding sections.

Simple Pattern Recognition Problem

We consider a neural network with 12 neurons ($n = 12$) with the objective of storing the four patterns shown in Figure 2 as memories. As indicated in this figure, twelve boxes are used to represent each pattern (in R^{12}), with each box corresponding to a vector component which is allowed to assume values between -1 and 1. For purpose of visualization, -1 will represent white, 1 will represent black, and the intermediate values will correspond to appropriate grey levels, as shown in Figure 3.

In all cases, we seek to design a neural network described by (1) with the interconnection configuration given in Figure 4 (in which interconnections are restricted

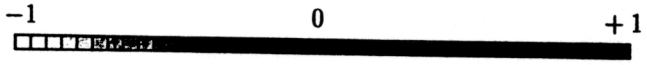

Figure 3: Grey levels

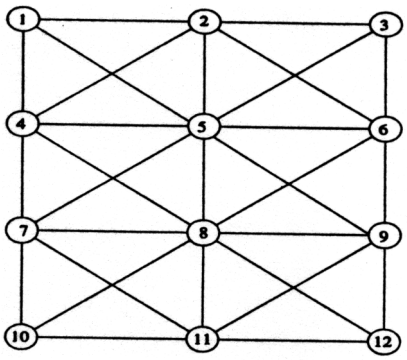

Figure 4: Interconnecting structure of a cellular neural network

to a radius $r = 1$, *i.e.*, a cellular neural network with $M = 4$, $N = 3$, and $r = 1$. cf. [5]). The index matrix for this interconnecting structure is as follows, where "0" represents no connection and "1" represents a connection.

$$S = S^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{5.1}$$

Case I: Nonsymmetric Design. We utilize the Sparse Design Procedure summarized in Section 4.2 to design a non-symmetric cellular neural network with the the index matrix given in (5.1). We obtain A as the identity matrix, and we obtain

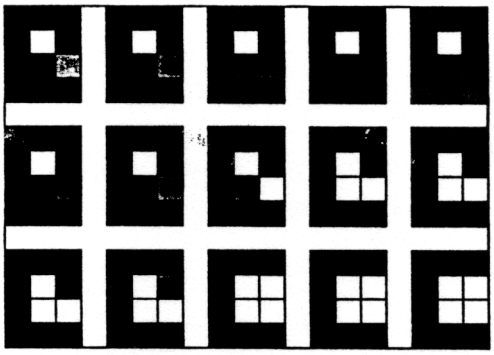

Figure 5: A typical evolution of pattern No. 1 of Figure 2

$$T = \begin{bmatrix}
0.333 & -0.000 & 0 & 0.333 & -14.333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3.500 & 15.000 & -3.500 & -3.500 & -10.500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.500 & 0 & -14.500 & 0.000 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.250 & 0 & 0 & 0.250 & -14.250 & 0 & 0.250 & 0 & 0 & 0 & 0 & 0 \\
-5.941 & -0.000 & -5.941 & -5.941 & -8.059 & 0.353 & -5.941 & -0.706 & 0.353 & 0 & 0 & 0 \\
0 & 0.000 & -5.125 & 0 & -8.875 & 5.625 & 0 & 3.750 & 5.625 & 0 & 0 & 0 \\
0 & 0 & 0 & -2.111 & -11.889 & 0 & -2.111 & -9.444 & 0 & -2.111 & -9.444 & 0 \\
0 & 0 & 0 & -7.158 & -6.842 & 0.211 & -7.158 & 0.368 & 0.211 & -7.158 & -14.211 & -0.000 \\
0 & 0 & 0 & 0 & -3.143 & -0.714 & 0 & 3.143 & -0.714 & 0 & -13.286 & 0.000 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.800 & -11.400 & 0 & 1.800 & -11.400 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -5.375 & -10.750 & -9.125 & -5.375 & -4.875 & -0.000 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.000 & -7.000 & 0 & -7.000 & 15.000
\end{bmatrix}$$

and

$$I = [0,\ 0.000,\ -0.000,\ -0.000,\ 0.706,\ -3.750,\ 9.444,\ 14.632,\ -3.143,\ 11.400,\ 10.750,\ 0.000]^T.$$

In the above computations, we chose $\mu = 15$ for the Sparse Design Procedure. From the robustness analysis result, we see that the upper bound for the admissible perturbation $\|\Delta T\|_\infty$ is $\mu - 1 = 14$. (For simplicity, in our examples, we consider $\Delta A = 0$ and $\Delta I = 0$. For the case when they are not zero, we can make similar conclusions and give similar examples.)

The performance of this network is illustrated by means of a typical simulation run of equation (2.1), shown in Figure 5. In this figure, the desired memory pattern is depicted in the lower right corner. The initial state, shown in the upper left corner, is generated by adding to the desired pattern zero-mean Gaussian noise with a standard deviation SD=1. The iteration of the simulation evolves from left to right in each row and from the top row to the bottom row. The desired pattern is recovered in 13 steps with a step size $h = 0.06$ in the simulation of equation (2.1). All simulations for the present paper were performed on a Sun SPARC Station using MATLAB.

Case II: Symmetric Design. Using the Symmetric Design Procedure outlined

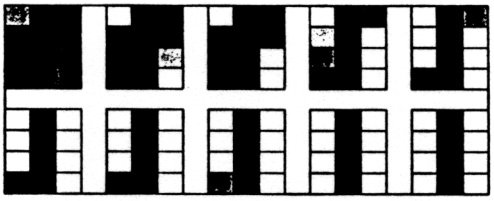

Figure 6: A typical evolution of pattern No. 3 of Figure 2

in Section 4.3 we can easily determine a symmetric matrix T for the present design. Starting with matrix T obtained in Case I, we determine that $\nu = 14$. Using our iterative algorithm for symmetric design summarized in Section 4.3, we find a symmetric matrix T_1 in four iterations as

$$T_1 = \begin{bmatrix}
-0.746 & -2.918 & 0 & 0.795 & -9.150 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2.918 & 15.671 & -1.980 & -3.410 & -5.805 & 0.607 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1.980 & 0.436 & 0 & -9.849 & -1.827 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.795 & -3.410 & 0 & 2.089 & -10.875 & 0 & -0.708 & -2.971 & 0 & 0 & 0 & 0 \\
-9.150 & -5.805 & -9.849 & -10.875 & -7.145 & -4.282 & -9.086 & -4.519 & -1.640 & 0 & 0 & 0 \\
0 & 0.607 & -1.827 & 0 & -4.282 & 7.226 & 0 & 2.629 & 3.043 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.708 & -9.086 & 0 & -2.955 & -9.039 & 0 & -1.431 & -8.339 & 0 \\
0 & 0 & 0 & -2.971 & -4.519 & 2.629 & -9.039 & -1.274 & 1.641 & -10.205 & -12.411 & -1.602 \\
0 & 0 & 0 & 0 & -1.640 & 3.043 & 0 & 1.641 & -1.539 & 0 & -10.976 & -3.569 \\
0 & 0 & 0 & 0 & 0 & 0 & -1.431 & -10.205 & 0 & 2.924 & -9.015 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -8.339 & -12.411 & -10.976 & -9.015 & -4.517 & -3.630 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.602 & -3.569 & 0 & -3.630 & 15.821
\end{bmatrix}$$

It can be verified that $\alpha^1, \alpha^2, \alpha^3$, and α^4 are also memories for system (2.1) with the symmetric matrix T_1 given above. From the robustness analysis result, we can verify that the allowable upper bound for the perturbation for system (2.1) with the above symmetric matrix T_1 is given by $\|\Delta T\|_\infty < 6.2807 - 1 = 5.2807$.

A typical simulation run for system (2.1) with T_1 given above is shown in Figure 6. In this case, the noisy pattern is generated by adding the same noise as in Case I (*i.e.*, Gaussian noise $N(0,1)$) to the desired pattern. Convergence occurs in 8 steps with $h = 0.06$.

5.2 Chinese Character Recognition

In this subsection, we propose to solve a Chinese character recognition problem using sparsely interconnected neural networks.

To demonstrate our method, we consider a small set of basic Chinese characters as our desired memory patterns which are represented by bipolar vectors. We consider the 25 desired memory patterns $\alpha^1, \cdots, \alpha^{25}$ shown in Figure 7. (The upper left pattern is denoted by α^1 and the lower right pattern is denoted by α^{25}.) These patterns constitute modules which represent individually or in combination, Chinese characters. They are coded in R^{81} in a similar manner as was done in Section 5.1 (including usage of grey levels).

We wish to synthesize a neural network (1) with $n = 81$ ($M = N = 9$) and with local interconnections restricted to a radius $r = 3$ (*i.e.*, a cellular neural network of the form (1) with $M = N = 9$ and $r = 3$), which will "remember" these modules. As mentioned above, some of the Chinese characters can be represented by two modules. In particular, the patterns given in Figure 7 can be used to generate at least 50 commonly used Chinese characters. To demonstrate this, we add one more vector, α^{26}, with every entry equal to 1 (black), to the set of desired memory patterns. In doing so, we can generate desired combinations for Chinese characters which are made up of some of the basic modules given in Figure 7. For instance, the character corresponding to α^6 means "sun" and the character corresponding to α^{14} means "moon". A new Chinese character can be generated as $\alpha^{27} = \alpha^6 + \alpha^{14} + \alpha^{26} \in \text{Span}(\alpha^1, \cdots, \alpha^{26}) \cap B^{81}$, which means "bright" (see Figure 8). Using the Modified Sparse Design Procedure (with $I = 0$) as discussed in Section 4.2, we only need to synthesize a system (1), in which the 81 neurons are arranged in a 9×9 array and the interconnections are restricted to local neighborhoods of radius $r = 3$, by employing these basic patterns. The resulting system will automatically "remember" all possible combinations of these basic components, which include the 50 commonly used Chinese characters mentioned above. This system has 2601 total interconnections, while a fully connected neural network with $n = 81$ will have a total of 6561 interconnections. By using the cellular neural network of the present example, we are able to reduce the total number of required interconnections to less than 40%.

A typical simulation run, involving the pattern $\alpha^{27} = \alpha^6 + \alpha^{14} + \alpha^{26}$ is depicted in Figure 9. The noisy initial pattern in Figure 9 (upper left corner) is generated by adding to α^{27} zero-mean Gaussian noise with a standard deviation SD=1. The desired pattern α^{27} is recovered in 24 steps with a step size $h = 0.227$ (lower right corner in Figure 9).

Simulation results showed that all the other vectors corresponding to the aforementioned 50 commonly used Chinese characters are indeed stored as memory vectors of the synthesized cellular neural network.

For the same initial noisy pattern shown in Figure 9, the desired pattern is recovered in 8 steps, with the same step size, when using a fully connected neural network (1) designed using the eigenstructure method summarized in Section 4.1 for the same desired set of memory patterns $\alpha^1, \cdots, \alpha^{26}$. One of the reasons for the lower convergence speed of cellular neural networks is that we only use local interconnections in such systems.

The most commonly used Chinese characters are roughly 6700 in number. More than half of these consist of approximately 500 basic characters (modules) or combinations of these characters, as described above. This example can be expanded by designing a cellular neural network which will store these 500 basic Chinese characters as well as combinations of these characters. In doing so, we will have stored

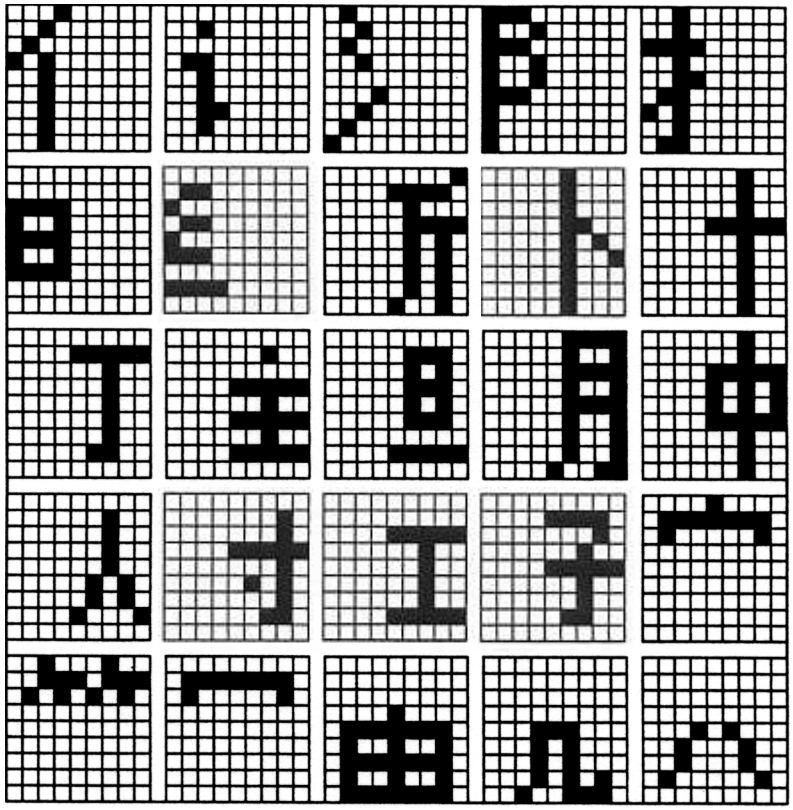

Figure 7: The twenty-five desired memory patterns

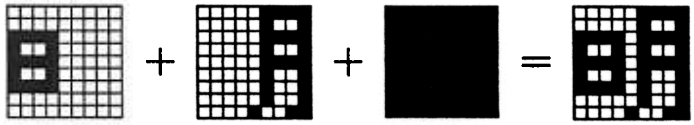

Figure 8: The Chinese character composed of patterns No. 6 and No. 14

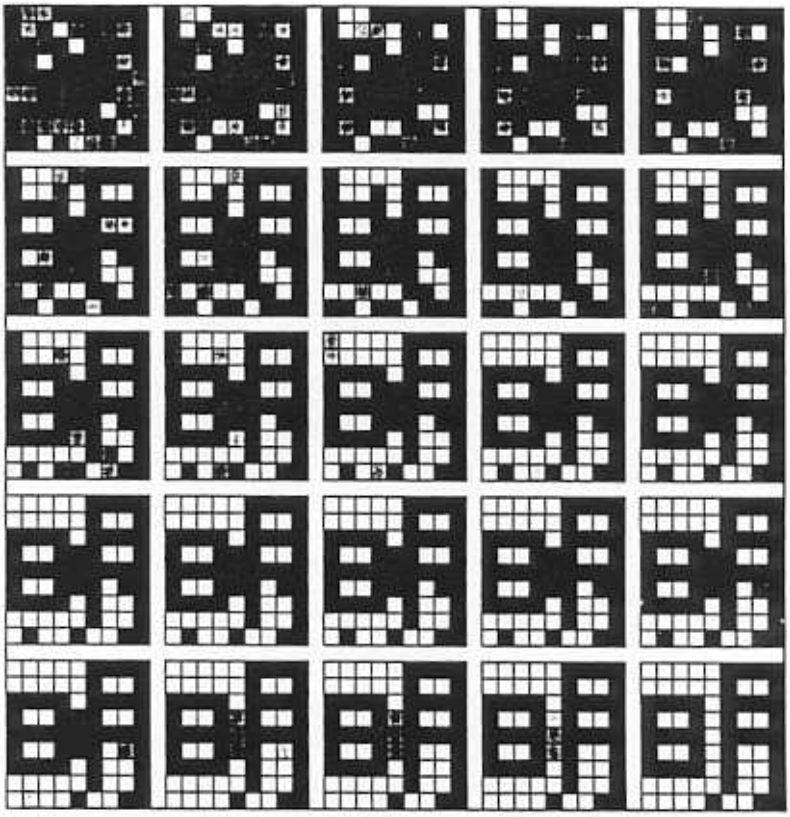

Figure 9: A typical evolution of the Chinese character composed of patterns No. 6 and No. 14

over one half of the 6700 commonly used Chinese characters. The remaining commonly used characters (numbering about 3000), will have to be stored separately, using the design procedure described in Section 4.

6 CONCLUDING REMARKS

In the present paper we considered a class of artificial neural networks which have the basic structure of analog Hopfield neural networks [12] and which use the (piecewise linear) saturation function to model the neurons. This model is closely related to the cellular neural networks introduced in [5] and to the neural networks defined on hypercubes in [15].

We provide upper bounds for the perturbations of parameters under which desired memories stored in a neural network (2.1) are preserved. This type of information is of great practical interest during the implementation process of such networks.

For system (2.1), a synthesis procedure for sparsely interconnected neural networks is provided. This procedure results in neural networks which satisfy a *prespecified* interconnecting structure and which do not require symmetric interconnections. Finally, a design procedure which enables us to design artificial neural networks with prespecified interconnecting structure and with symmetric interconnection matrix for storing a given set of desired bipolar patterns as memories is presented

Results presented herein are applicable to many other neural network models such as neural networks described on a closed hypercube [15], iterated-map neural networks [18], and discrete-time Hopfield model [11].

References

[1] S. V. B. Aiyer, M. Niranjan, and F. Fallside, "A theoretical investigation into the performance of the Hopfield model," *IEEE Trans. Neural Networks*, vol. 1, pp. 204–215, June 1990.

[2] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Trans. Circuits Syst.-I*, vol. 40, pp. 147–156, Mar. 1993.

[3] L. O. Chua and P. Thiran, "An analytic method for designing simple cellular neural networks," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 1332–1341, Nov. 1991.

[4] L. O. Chua and C. W. Wu, "On the universe of stable cellular neural networks," *Int. J. Circuit Theory App.*, vol. 20, pp. 497–572, 1992.

[5] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257–1272, Oct. 1988.

[6] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 13, pp. 815–826, Sep./Oct. 1983.

[7] M. Cottrell, "Stability and attractivity in associative memory networks," *Biol. Cybern.*, vol. 58, pp. 129–139, 1988.

[8] S. R. Das, "On the synthesis of nonlinear continuous neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, pp. 413–418, March/Apr. 1991.

[9] R. M. Golden, "The 'brain-state-in-a-box' neural model is a gradient descent algorithm," *J. of Math. Psych.*, vol. 30, pp. 73–80, March 1986.

[10] A. Guez, V. Protopopsecu, and J. Barhen, "On the stability, storage capacity, and design of nonlinear continuous neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 80–87, Jan./Feb. 1988.

[11] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA,* vol. 79, pp. 2554–2558, Apr. 1982.

[12] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. USA,* vol. 81, pp. 3088–3092, May 1984.

[13] S. Hui and S. H. Żak, "Dynamical analysis of the brain-state-in-a-box (BSB) neural models," *IEEE Trans. Neural Networks,* vol. 3, pp. 86–94, Jan. 1992.

[14] J. D. Keeler, "Basins of attraction of neural network models," *AIP Conf. Proc. 151,* Snowbird, UT, pp. 259–264, 1986.

[15] J.-H. Li, A. N. Michel, and W. Porod, "Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1405–1422, Nov. 1989.

[16] 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. 295–309, Apr. 1994.

[17] D. Liu and A. N. Michel, *Dynamical Systems with Saturation Nonlinearities: Analysis and Design,* New York: Springer-Verlag, 1994.

[18] C. M. Marcus and R. M. Westervelt, "Dynamics of iterated-map neural networks," *Physical Review A,* vol. 40, pp. 501–504, July 1989.

[19] R. Perfetti, "A neural network to design neural networks," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 1099–1103, Sept. 1991.

[20] L. Personnaz, I. Guyon, and G. Dreyfus, "Information storage and retrieval in spin-glass like neural networks," *J. Physique Lett.*, vol. 46, pp. L359–365, Apr. 1985.

[21] L. Personnaz, I. Guyon, and G. Dreyfus, "Collective computational properties of neural networks: New learning mechanisms," *Physical Review A,* vol. 34, pp. 4217–4228, Nov. 1986.

[22] F. M. A. Salam, Y. Wang, and M.-R. Choi, "On the analysis of dynamic feedback neural nets," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 196–201, Feb. 1991.