Second-Order Asymmetric BAM Design With a Maximal Basin of Attraction

Jyh-Yeong Chang and Chien-Wen Cho

***Abstract*—Bidirectional associative memory (BAM) generalizes the associative memory (AM) to be capable of performing two-way recalling of pattern pairs. Asymmetric bidirectional associative memory (ABAM) is a variant of BAM relaxed with connection weight symmetry restriction and enjoys a much better performance than a conventional BAM structure. Higher-Order associative memories (HOAMs) are reputed for their higher memory capacity than the first-order counterparts, yet there are few HOAMs design schemes proposed up to date. To this end, we are concerned in this paper with designing a second-order asymmetric bidirectional associative memory (SOABAM) with a maximal basin of attraction, whose extension to a HOABAM is possible and straightforward. First, a necessary and sufficient condition is derived for the connection weight matrix of SOABAM that can guarantee the recall of all prototype pattern pairs. To respect the complete recall theorem, an adaptive local training rule, which is adaptive in the learning step size and updates only the entries in the connection weight related to the most needful bit of a prototype, is formulated and it leads to better results and faster design. Then derived is a theorem, designing a SOABAM further enlarging the quantities required to meet the complete recall theorem will enhance the capability of evolving a noisy pattern to converge to its association pattern vector without error. Based on this theorem, our algorithm is also modified to ensure each training pattern is stored with a basin of attraction as large as possible. Computer simulations over the color graphics adapter (CGA) fonts have demonstrated the superiority of the proposed local training rule over other prevailing BAM schemes.**

***Index Terms*—Asymmetric BAM, basin of attraction, Hebbian learning, second-order associative memory.**

I. Introduction

THERE are wide-spread applications for associative memories (AMs), which include pattern recognition, data storage and retrieval, noise removal, and/or error correction [1], [2]. The bidirectional associative memory (BAM) [3] is a generalization of Hopfield AM model [4]. A BAM model consists of neurons in two layers, the X-layer and the Y-layer, and a connection weight matrix W between the X-layer and the Y-layer. A conventional BAM model admits a logical symmetry restriction: the connection weight matrix from the Y-layer to the X-layer is the transpose of the connection weight matrix from the X-layer to the Y-layer. A BAM performs a two-way associative search for stored bipolar vector pairs by forward and backward iterations between these two layers. For example, a BAM model is designed to store the following M bipolar pattern pairs $(X^1, Y^1), (X^2, Y^2), \ldots, (X^M, Y^M)$, with two-way retrieval capability as $X^m \longleftrightarrow Y^m$, where $X^m \in \{-1, +1\}^n$ and $Y^m \in \{-1, +1\}^p$; $m = 1, 2, \ldots, M$. The model iterates $X^{\text{new}} = \text{sgn}(WY^{\text{old}})$ and $Y^{\text{new}} = \text{sgn}(W^T X^{\text{old}})$ alternately until the network reaches a stable pattern pair (X^*, Y^*), which should be one of the pattern pairs stored in the BAM system. In the above, if there are just one set of pattern vectors, i.e., $Y^m = X^m$, to be stored in the BAM, then the model reduces to the Hopfield network model, which is only autoassociative and unidirectional in pattern memorization and recall.

In general, BAMs suffer from the following two drawbacks: low storage capacity and weak error-correcting capability. Many research efforts have been made to reduce these drawbacks. These efforts can mainly be classified into the following two directions. The first attempt deals with algorithmic improvements of the encoding schemes. In this context, multiple training method [5], Ho-Kashyap learning [6], Hamming stability learning [7], optimal stability training [8], and adaptive relaxation method [9] were introduced recently. The other attempt involves structure modification of the BAM network. These include threshold vector augmentation [10], multilayer BAM's [11], feedforward BAM's [12], and higher-order connections [13] and [14]. Also in this line of architecture modifications, the logic symmetry of the weight matrix in a BAM model is relaxed to pursue a higher performance of a BAM and leads to what is known as the asymmetric BAM (ABAM) [15]. Embedded in the ABAM structure, enlarging supporting function approach [16] and optimization-based design [17] are proposed to our attentions.

The rationale behind the most encoding designs above is essentially based on making the most use of the correlation existing among the bits of corresponding pattern pairs. Therefore, certain degrees of bit error correcting capability are inherited to all these methods, although different for each method. In this regard, a BAM design scheme that can efficiently utilize or create additional bit's correlation would be more likely to outperform others. In line with generating a very large additional bit's correlation, the HOAM approach exploits the quadratic bit's correlation of pattern pairs and is highly reputed for its excessively higher memory capacity than the first-order counterpart [18]. The easy implementation alternative of HOAM by optical holograms [18] and [19] constitutes another advantage of HOAM's. Moreover, we can easily build translation, rotation, and scale invariances into a HOAM network, which is

Manuscript received July 26, 2002; revised January 23, 2003. This work was supported in part by the National Science Council of Taiwan, R.O.C, under Grant NSC 90-2213-E-009-107 and in part by the Ministry of Education under Grant EX-91-E-FA06-4-4, the program for promoting university academic excellence, Taiwan, R.O.C. This paper was recommended by Associate Editor C.T. Lin.

The authors are with the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: jychang@cc.nctu.edu.tw).

Digital Object Identifier 10.1109/TSMCA.2003.811505

1083-4427/03$17.00 © 2003 IEEE

useful in invariant pattern recognition problems [20]. Stability and statistical property of SOAMs [13], [14] is discussed in the literature, yet there are few HOAM design schemes proposed up to date. Enjoying the degrees of freedom relieved by the second-order connections and logical asymmetry, we will propose in this paper a SOABAM design method with high performance. This new SOABAM system not only can guarantee the perfect recall of all pattern pairs but also can maximize the basin of attraction of each pattern pair. The extension from designing a SOABAM to a HOABAM is possible and straightforward.

II. Complete Recall Design of SOABAMs

For the SOABAM, storing pattern pairs (X^m, Y^m), $m = 1, 2, \ldots, M$, where $X^m \in \{-1, +1\}^n$ and $Y^m \in \{-1, +1\}^p$, as Kosko's outer product rule [3] leads to the following expression for connection weight matrix $U = [u_{ijk}]$

$$u_{ijk} = \sum_{m=1}^{M} y_i^m y_j^m x_k^m, \qquad \forall\, i \,\&\, j = 1, 2, \ldots, p \text{ and } k = 1, 2, \ldots, n.$$

Matrix U is of dimensions $p \times p \times n$ and holds the second-order connections from the Y-layer to the X-layer. Likewise, another matrix $W = [w_{ijk}]$, being of dimensions $n \times n \times p$ and holding the second-order connections from the X-layer to the Y-layer, will be given by

$$w_{ijk} = \sum_{m=1}^{M} x_i^m x_j^m y_k^m, \qquad \forall\, i \,\&\, j = 1, 2, \ldots, n \text{ and } k = 1, 2, \ldots, p.$$

It is known that the following necessary and sufficient conditions for the correlation matrices U and W are needed to guarantee the recall of all training patterns [5].

Lemma 1: The training pattern pair (X^m, Y^m), $m = 1, 2, \ldots, M$, is a stable pattern pair of a SOABAM if and only if

$$F_{x_k}^m = \left(\sum_{i=1}^{p} \sum_{j=1}^{p} y_i^m y_j^m u_{ijk} \right) \cdot x_k^m > 0 \quad k = 1, 2, \ldots, n, \tag{1}$$

and

$$F_{y_k}^m = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} x_i^m x_j^m w_{ijk} \right) \cdot y_k^m > 0 \quad k = 1, 2, \ldots, p. \tag{2}$$

We now derive the necessary and sufficient conditions for the correlation matrices U and W to guarantee the recall of all training patterns.

Theorem 1 (Complete Recall Theorem): Let

$$\boldsymbol{F}_x = \begin{pmatrix} F_{x_1}^1, & F_{x_1}^2 & \cdots & F_{x_1}^M \\ F_{x_2}^1, & \vdots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ F_{x_n}^1, & F_{x_n}^2 & \cdots & F_{x_n}^M \end{pmatrix} \tag{3}$$

and

$$\boldsymbol{F}_y = \begin{pmatrix} F_{y_1}^1, & F_{y_1}^2 & \cdots & F_{y_1}^M \\ F_{y_2}^1, & \vdots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ F_{y_p}^1, & F_{y_p}^2 & \cdots & F_{y_p}^M \end{pmatrix}. \tag{4}$$

All prototype pattern pairs can be recalled if and only if the correlation matrices U and W satisfy

$$\boldsymbol{F}_x > 0 \tag{5}$$

and

$$\boldsymbol{F}_y > 0. \tag{6}$$

Thus, if inequalities (1) and (2) have a solution, then there exist positive constants b_x and b_y so that the following two linear inequalities hold:

$$F_{x_k}^m = \left(\sum_{i=1}^{p} \sum_{j=1}^{p} y_i^m y_j^m \cdot u_{ijk} \right) \cdot x_k^m \geq b_x > 0 \quad k = 1, 2, \ldots, n \tag{7}$$

and

$$F_{y_k}^m = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} x_i^m x_j^m \cdot w_{ijk} \right) \cdot y_k^m \geq b_y > 0 \quad k = 1, 2, \ldots, p \tag{8}$$

where $m = 1, 2, \ldots, M$.

The complete recall theorem above provides the necessary and sufficient conditions for the correlation matrices U and W to ensure the recall of all prototype pattern pairs. The theorem is an extension of the complete recall condition in [5]. Note that the relaxation of abandoning logical symmetry in ABAM design admits the *independent design* of the weight matrix U from the Y- to X-layer and the weight matrix W from the X- to Y-layer. As a consequence, the training rule and properties drawn from the Y-layer to the X-layer can be applied equivalently to the case from the X-layer to the Y-layer. For brevity, we will only describe in detail the interesting context drawn from the Y-layer to the X-layer. The body of interests that can be drawn similarly from the X-layer to the Y-layer will not be explicitly stated unless necessary.

To respect the above inequalities for all the M pattern pairs, we have to train the connection weight matrix U starting from pattern pair one up to pattern pair M repeatedly. Let the current training pattern pair be denoted as (X^{m_c}, Y^{m_c}), e.g., $m = m_c$; and let $\text{net}(x_{k_c}^{m_c}) = \sum_{i=1}^{p} \sum_{j=1}^{p} y_i^{m_c} y_j^{m_c} u_{ijk_c}$. Stating from $k = 1$ to n, if for some k, denoted as k_c, such that $F_{x_{k_c}}^{m_c} < 0$, i.e., $x_{k_c}^{m_c} \neq \text{sgn}(\text{net}(x_{k_c}^{m_c}))$, then the following updating rule at iteration t is executed to train the sublattice weights [13] of U related to $x_{k_c}^{m_c}$ as given by

$$u_{ijk_c}(t) = u_{ijk_c}(t-1) + \alpha \cdot y_i^{m_c} y_j^{m_c} x_{k_c}^{m_c} \quad \forall\, i \,\&\, j = 1, 2, \ldots, p \tag{9}$$

where α is learning constant. Otherwise, for those k's, denoted as k_o, such that $x_{k_o}^{m_c} = \text{sgn}(\text{net}(x_{k_o}^{m_c}))$, leave sublattice weights of U related to $x_{k_o}^{m_c}$ unchanged. The updated weights according to (9) by the Rosenblatt's perceptron learning rule [21] is only guaranteed to increase the value of $F_{x_{k_c}}^{m_c}$. It is not necessarily

good to recall the k_cth bit for other pattern vectors. Namely, the updated U could make some other pattern's $F^m_{x_{k_c}}$, $m \neq m_c$ values smaller, and thus, such learning hinders this pattern's recall capability in the k_cth bit. As a result, sequentially learning prototype pattern vector pairs according to (9) may result in updating entries in weight matrix U back and forth, and such learning procedure is thus apt to be oscillatory and becomes difficult to converge. It usually takes hundreds or even thousands of epochs to complete a learning task.

A more efficient way to update U is to choose the smallest $F^m_{x_k}$ among all possible m and k combinations, which is denoted as $F^{m_s}_{x_{k_s}}$ here. We then update only its corresponding sublattice entries involved in weight matrix U as given by

$$u_{ijk_s}(t) = u_{ijk_s}(t-1) + \alpha \cdot y_i^{m_s} y_j^{m_s} x_{k_s}^{m_s} \quad \forall i \,\&\, j = 1, 2, \ldots, p. \tag{10}$$

The smallest $F^m_{x_k}$ learning procedure will not stop until $\boldsymbol{F}_x > 0$. In this way, in every learning step we change only the sublattice weights in U related to the smallest, namely, the most needful, $F^m_{x_k}$ to change. This updating rule will be referred to as *local training rule*, because every time the learning changes only the entries of connection weight U related to the single entry, $F^{m_s}_{x_{k_s}}$, in $\boldsymbol{F}_x$. Such local training rule will improve the performance of the SOABAM learning. In the reverse direction, connection weight matrix W follows similar learning rule without further explanation.

III. Maximizing the Basin of Attraction in SOABAM Design

For the SOABAM to be noise tolerant, the following theorem will present the relationship between error-correcting capability and the stability margin, b_x, and b_y, of a SOABAM.

Theorem 2: Suppose that $H_y = H(A, Y^m)$ is the Hamming distance between an input pattern A and a stored pattern Y^m, that $F^m_{x_k} \geq b_x > 0$ for $k = 1, 2, \ldots, n$, and that $|u_{ijk}| \leq D$. If

$$H_y = H(A, Y^m) \leq -\frac{p}{2} + \frac{1}{2}\sqrt{p^2 + \frac{b_x}{D}} \tag{11}$$

then the input A converges to its association pattern X^m in one recall iteration.

Proof: See the Appendix.

Theorem 2 states that for the given b_x, the pattern Y^m will have at least the basin of attraction H_y, in one step. Similarly, we can show that for the given b_y and bounded w_{ijk}, the pattern X^m will have at least the basin of attraction H_x

$$-\frac{n}{2} + \frac{1}{2}\sqrt{n^2 + \frac{b_y}{D}}$$

in one step. Under bounded constraint of u_{ijk}, i.e., $|u_{ijk}| \leq D$, inequality (11) implies that the larger the value b_x is, the larger the basin of attraction, H_y, of the Y-layer is for one iteration recall. In the absence of an analytical method for calculating exactly the basin of attraction for more than one iteration, we proceed on the heuristic that the size of the overall basin of attraction is also larger if b_x is larger. For convenience, b_x will be called, in an indirect sense, the basin of attraction of the Y-layer hereafter since H_y depends on b_x governed by (11). Because we want to form a basin of attraction that is as large as possible, we can choose the maximal value of b_x such that local training rule does converge. A practical way of designing U having maximal b_x value can be proceed as follows. Starting from the connection weight matrix U encoded by Kosko's BAM correlation rule, then derive new matrix U that can increase b_x gradually until the training rule (10) fails to converge. To make every updated u_{ijk} fulfill the constraint $|u_{ijk}| \leq D$, the updated u_{ijk} has to go through the following hard limiter-type function $\Phi(x)$ as defined by

$$\Phi(x) = \begin{cases} -D, & \text{if } x < -D \\ x, & \text{if } -D \leq x \leq D \\ D, & \text{if } x > D. \end{cases}$$

With the introduction of $\Phi(x)$ to limit the magnitude of the updated weight if necessary, we can respect the bounded weight constraint. For quicker convergence of learning procedure, the proposed local training rule is further improved to *adaptive local training rule*, whose details will be discussed in the following.

IV. Adaptive Local Training Rule for SOABAM Design

In this section, we will derive one lemma and one theorem concerning with the quantitative trend of the behavior of the local training rule, but the hard limiter-type function $\Phi(x)$ above to bound the weight will be ignored in the derivation. From our simulation experience, the limiting function is seldom activated for bounding enforcement during the course of learning. The frequency to activate limiting function $\Phi(x)$ increases as the learning iteration increases, but this frequency still remains few even in the final stage of a learning task. Therefore, it is reasonable to neglect this random occurrence of activating $\Phi(x)$ in the theoretical derivation below.

The increment of $F^{m_s}_{x_{k_s}}$ after the local training rule, (9), is constant for a constant learning rate α. This fact is illustrated by the following lemma.

Lemma 2: If $F^{m_s}_{x_{k_s}}(t-1) \leq b_x$, we use the local training rule (10), then

$$F^{m_s}_{x_{k_s}}(t) = F^{m_s}_{x_{k_s}}(t-1) + \alpha \cdot p^2. \tag{12}$$

Proof: If $F^{m_s}_{x_{k_s}}(t-1) \leq b_x$, then the local training rule (10) is adopted. We have

$$F^{m_s}_{x_{k_s}}(t) = \left(\sum_{i=1}^{p} \sum_{j=1}^{p} y_i^{m_s} y_j^{m_s} \cdot u_{ijk_s}(t) \right) \cdot x_{k_s}^{m_s}.$$

Substituting $u_{ijk_s}(t)$ from (10) produces

$$\begin{aligned} F^{m_s}_{x_{k_s}}(t) &= \left(\sum_{i=1}^{p} \sum_{j=1}^{p} y_i^{m_s} y_j^{m_s} \cdot u_{ijk_s}(t-1) \right) \cdot x_{k_s}^{m_s} + \alpha \cdot p^2 \\ &= F^{m_s}_{x_{k_s}}(t-1) + \alpha \cdot p^2. \end{aligned}$$

Q.E.D.

The lemma shows that the local training rule can move $F^{m_s}_{x_{k_s}}(t-1)$ at a constant step size toward the positive direction if learning rate α is positive.

Borrowed the relaxation rule of Oh and Kothari [9] to find a connection matrix U that can attain a maximal b_x at a faster convergence rate, we propose that the learning rate be changed

adaptively to a relaxation factor λ and the step difference between b_x and $F_{x_{k_s}}^{m_s}(t-1)$. The local training rule can thus be modified as

If $F_{x_{k_s}}^{m_s}(t-1) \le b_x$, then u_{ijk_s} is updated by

$$u_{ijk_s}(t) = \Phi\big(u_{ijk_s}(t-1) + \alpha_x(t)\cdot y_i^{m_s} y_j^{m_s} x_{k_s}^{m_s}\big) \qquad (13)$$

where

$$\alpha_x(t) = \lambda \cdot \frac{b_x - F_{x_{k_s}}^{m_s}(t-1)}{p^2}.$$

We denoted (13) above as the *adaptive local training rule*. For a given positive constant b_x, if the training rule (10) admits a solution of $F_{x_{k_s}}^{m_s} \ge b_x > 0$, then the training rule (13) will also be convergent and can converge faster. Since a larger difference between b_x and $F_{x_{k_s}}^{m_s}$ yields a larger learning step size, the improvement of convergence rate and capability is expected. The rationale behind how to choose the adaptive learning rate, $\alpha_x(t)$, in view of the relaxation factor λ is illustrated by the following theorem.

Theorem 3: If $F_{x_{k_s}}^{m_s}(t-1) \le b_x$ and we use the above local training rule, then

$$F_{x_{k_s}}^{m_s}(t) - b_x = (1-\lambda)\Big(F_{x_{k_s}}^{m_s}(t-1) - b_x\Big). \qquad (14)$$

Proof: If $F_{x_{k_s}}^{m_s}(t-1) \le b_x$, applying the adaptive local training rule (13) leads to

$$\begin{aligned} F_{x_{k_s}}^{m_s}(t) - b_x &= F_{x_{k_s}}^{m_s}(t-1) + \alpha_x(t)\cdot p^2 - b_x \\ &= F_{x_{k_s}}^{m_s}(t-1) + \lambda \cdot \frac{b_x - F_{x_{k_s}}^{m_s}(t-1)}{p^2}\cdot p^2 - b_x \\ &= (1-\lambda)\Big(F_{x_{k_s}}^{m_s}(t-1) - b_x\Big). \end{aligned}$$

Q.E.D.

It follows from Theorem 3 that if $\lambda < 1$, then $F_{x_{k_s}}^{m_s}(t)$ is still less than b_x; while if $\lambda > 1$, then $F_{x_{k_s}}^{m_s}(t)$ will be greater than b_x. The training rule (13) above is similar in spirit to the relaxation rule in [9]. In general, we restrict λ to the range $0 < \lambda < 2$; a better selection of λ is usually made slightly larger than 1.5 [9]. Exploiting the relaxation parameter λ could not only converge to the target value easier but also could lead to several multiples of ten times reduction in the number of epochs to converge compared to a constant step size learning one, for example, perceptron learning [9].

As mentioned previously, the proposed adaptive local perceptron learning rule is only guaranteed to increase the value of $F_{x_{k_s}}^{m_s}$. It is not necessarily good to $F_{x_k}^{m}$, $m \ne m_s$ or $k \ne k_s$. With this concept in mind, we will incorporate *pocket algorithm* [22] into our adaptive local training rule during the learning procedure. Perceptron learning is quite appropriate for separable samples to be dichotomized, i.e., problems for which some set of weights is available that correctly classifies linearly separable training patterns after a finite number of mistakes. Nonseparable problems are a different story. The fact that no set of weights can correctly classify all training patterns implies that a set of weights which correctly classifies as large a fraction of the training patterns as possible is preferred. Pocket algorithm is developed in perceptron learning to determine a set of weight in this optimal sense [22]. Adapting the concept behind the pocket algorithm, we save "in the pocket" of the weight matrix GU, the Global connection matrix U that produces the largest $F_{x_k}^{m}$ value during the course of adaptive local training algorithm. To improve the learning speed and accuracy at every learning step, we generate the increment of b_x, Δb_x, randomly. Moreover, the magnitude of Δb_x will decrease as the number of training iteration increases. Gradually decreasing the learning step size, and hence learning rate, is helpful in improving training convergence and is widely adopted in neural and adaptive learning routines [2].

To conclude, the following algorithm is presented for SOABAM design with maximal basin of attraction.

Algorithm 1 (Maximal Basin of Attraction b_x Algorithm):

Step 1) Initialize. $b_x = 0$, $D =$ constant.

$$u_{ijk}(0) = \Phi\left(\sum_{m=1}^{M} y_i^m y_j^m x_k^m\right), \qquad i \,\&\, j = 1, 2, \ldots, p;\quad k = 1, 2, \ldots, n.$$

$$\begin{aligned} b_x^o &= F_{x_{k_s}}^{m_s}(0) = \min_{1\le m\le M}\min_{1\le k\le n} F_{x_k}^{m}(0) \\ &= \min_{1\le m\le M}\min_{1\le k\le n}\left(\sum_{i=1}^{p}\sum_{j=1}^{p} y_i^m y_j^m u_{ijk}(0)\right)\cdot x_k^m. \end{aligned}$$

$g\min \leftarrow F_{x_{k_s}}^{m_s}(0)$, $GU(0) \leftarrow U(0)$, and $cnt \leftarrow 0$. Generate Δb_x and compute $b_x = b_x^o + \Delta b_x$.

Step 2) Train U for the pattern bit $x_{k_s}^{m_s}$ using the adaptive local training rule (13). Compute

$$\begin{aligned} b_x^o &= F_{x_{k_s}}^{m_s}(t) = \min_{1\le \mathrm{m}\le \mathrm{M}} \min_{1\le \mathrm{k}\le \mathrm{n}} \mathrm{F}_{\mathrm{x_k}}^{\mathrm{m}}(\mathrm{t}) \\ &= \min_{1\le m\le M}\min_{1\le k\le n}\left(\sum_{i=1}^{p}\sum_{j=1}^{p} y_i^m y_j^m u_{ijk}(t)\right)\cdot x_k^m \end{aligned}$$

then $cnt \leftarrow cnt + 1$. If $F_{x_{k_s}}^{m_s}(t) > g\min$, then $g\min \leftarrow F_{x_{k_s}}^{m_s}(t)$, $GU \leftarrow U$ and $cnt \leftarrow 0$.

Step 3) Generate Δb_x randomly and compute $b_x = b_x^o + \Delta b_x$. If $cnt < thd$ (maximal number of iteration), then go to Step 2. Otherwise, go to Step 4.

Step 4) Output the solution: $U \leftarrow GU$, $b_x \leftarrow g\min$, and stop.

V. Simulation Results for ABAM Designs

To validate the effectiveness of our proposed SOABAM schemes, the benchmark set of IBM PC color graphic adapter (CGA) character pairs of pixel sizes 7×7 were utilized. Twenty-six uppercase English letter, X-layer together with the corresponding lowercase counterparts, Y-layer, constitutes the prototype pattern pairs for BAM storage. For processing convenience, each one of the 52 prototypes is vectorized row by row, in which the black pixels are coded by $+1$ while the white pixels are coded by -1. Thus, either X- or Y-layer is composed of 26 bipolar vectors of dimension 49, i.e., $M = 26$ and $n = p = 49$ in our notation used.

A. Learning Scheme Comparisons via FOABAM Design

The improvement verification of various learning schemes mentioned above can be done either on first-order or second-order ABAM design. For efficiency purpose, the

TABLE I
COMPARISON OF ATTAINABLE b_x AND b_y, CPU TIME REQUIRED, AND NUMBER OF ENTRY UPDATES OF FOABAM DESIGN BY CONVENTIONAL PERCEPTRON AND PROPORTIONAL PERCEPTRON LEARNING RULES (CPU TIME AND NUMBER OF ENTRY UPDATES ARE COMPUTED FORWARD AND BACKWARD DESIGN LOOPS ALTOGETHER HEREAFTER)

	b_x	b_y	CPU Time (min)	Number of Entry Updates ($\times 10^6$)
(Conventional) Perceptron	23.75	46.52	18.04	11.92
Proportional Perceptron	29.24	63.78	5.51	2.28

improvement experiment will be made over the first-order ABAM (FOABAM) design since the design time required by the SOABAM is very long and thus time consuming, which will be seen in the later of this section. These learning schemes are concerned with gradually enlarging b_x and b_y of a FOABAM through iterative learning. They are basically perceptron learning or its variants and are verified in the following way. At each learning epoch, the increment of b_x is normally kept constant of ten, i.e., $\Delta b_x = 10$. If the training iteration cannot converge within 20 learning epochs, then a new learning iteration is invoked by reducing Δb_x by half. If a learning iteration can converge within 20 epochs, then Δb_x will be resumed to its normal value ten again. The learning procedure will not stop until $\Delta b_x \leq 1$.

The performance comparisons of the different learning schemes are evaluated by the following three experiments. First, we will demonstrate the superiority of the *proportional perceptron learning*, the training step size in every entry update of the connection weight matrix U is in proportion to the magnitude of the increment of b_x, over the conventional (nonproportional) perceptron learning in the FOABAM design. For the currently presented patterns X^m and Y^m, if $b_x < \boldsymbol{F}_x(t-1)$, the perceptron learning rule updates the ith row of the connection matrix U by

$$u_{ij}(t) = \Phi\left(u_{ij}(t-1) + \alpha_0 \cdot x_i^m y_j^m\right), \qquad \forall\, i = 1, 2, \ldots, p, \quad j = 1, 2, \ldots, n;\ m = 1, 2, \ldots, M. \tag{15}$$

Otherwise, leave the weight unchanged. If the learning constant α_0 is set to be $1/p$, a necessary update of the presented pattern's bit y_j^m will lead to a unitary increment of b_x. A more efficient update of u_{ij} may be proceeded by allowing the effective learning constant to be in proportion to the step size of $\Delta b_x(t)$, $b_x - \boldsymbol{F}_x(t-1)$, as

$$u_{ij}(t) = \Phi\left(u_{ij}(t-1) + \alpha_0 \cdot \Delta b_x(t) \cdot x_i^m y_j^m\right) \quad \forall\, i = 1, 2, \ldots, p,\ j = 1, 2, \ldots, n \quad m = 1, 2, \ldots, M. \tag{16}$$

We call above the *proportional perceptron learning rule*. For comparison, the number of entry updates of u_{ij} and w_{ij}, CPU time, running on an IBM Pentium PC-800 MHz, required to finish a design, and reachable b_x and b_y, were used as the performance index. To have a whole picture, the performance index commented below is usually given as the one counting both the forward and backward design loops altogether, unless stated otherwise. On the other hand, the numerical values listed in the table are generally given by the forward and backward designs separately to have more details. As such, Table I displays the results of these two learning rules to design CGA pattern pairs by FOABAM models. From the table, the design by conventional perceptron learning needs 11.92×10^6 entry updates and 18.04 mins CPU time, whereas by proportional perceptron needs 2.28×10^6 entry updates and 5.51 mins CPU time, respectively. As to reachable basin of attraction, the conventional perceptron learning rule converges to $b_x = 23.75$ and $b_y = 46.52$, respectively, and the proportional perceptron learning rule attains $b_x = 29.24$ and $b_y = 63.78$, respectively. Proportional learning increases more than 32% attraction radius, counting b_x and b_y altogether hereafter, compared with the conventional perceptron learning, while needs only about one-fifth of the number of entry updates by the conventional one. Moreover, the improvement of the proportional perceptron learning can be further enhanced by introducing the relaxation factor λ into the proportional perceptron learning rule as given by

TABLE II
COMPARISON OF ATTAINABLE b_x AND b_y, CPU TIME REQUIRED, AND NUMBER OF ENTRY UPDATES OF FOABAM DESIGN BY ADAPTIVE LEARNING RULES WITH DIFFERENT RELAXATION FACTORS λ

	b_x	b_y	CPU Time (min)	Number of Entry Updates ($\times 10^6$)
λ=1.6	28.49	63.88	3.53	1.50
λ=0.32	26.66	57.53	8.31	50.97
λ=8.0	7.96	32.24	1.66	0.36

TABLE III
COMPARISON OF ATTAINABLE b_x AND b_y OF FOABAM DESIGN BY ADAPTIVE LOCAL AND ADAPTIVE LEARNING RULES

	b_x	b_y
Adaptive Local ($\lambda = 1.6$)	59.69	130.19
Adaptive ($\lambda = 1.6$)	28.49	63.88

$$u_{ij}(t) = \Phi\left(u_{ij}(t-1) + \alpha_x(t) \cdot x_i^m y_j^m\right) \quad \forall\, i = 1, 2, \ldots, p, j = 1, 2, \ldots, n;\ m = 1, 2, \ldots, M \tag{17}$$

where $\alpha_x(t) = \lambda \cdot \alpha_0 \cdot \Delta b_x(t) = \lambda \cdot \Delta b_x(t)/p$. The proportional perceptron learning together with relaxation factor inclusion is called the *adaptive learning rule*. As noted in the previous section and [9], λ should be slightly larger than 1.5 for good performance; here we choose $\lambda = 1.6$ in comparison to other two selections: $\lambda = 0.32$ and $\lambda = 8.0$. The learning performances of these three relaxation values were summarized in Table II. From the table, the design by adaptive learning at $\lambda = 1.6$ needs 1.50×10^6 entry updates and 3.53 min CPU time, whereas adaptive learning at $\lambda = 0.32$ and 8.0 needs 50.97×10^6 and 0.36×10^6 entry updates and 8.31 and 1.66 min CPU time, respectively. The proportional perceptron learning rule listed in Table I is equivalent to adaptive learning with $\lambda = 1.0$. For attainable basin of attraction, the better relaxed adaptive perceptron learning rule, $\lambda = 1.6$, converges to $b_x = 28.49$ and $b_y = 63.88$, which is almost identical to those obtained by the proportional perceptron learning rule at relaxation factor $\lambda = 1.0$, but the entry updates are reduced to 67% of $\lambda = 1.0$. Small relaxation $\lambda = 0.32$ can reach about 91% of attraction

TABLE IV
AVERAGE ERROR CORRECTION PROBABILITY $\mathrm{prob}(r)$ VERSUS HAMMING DISTANCE r OF VARIOUS BAM DESIGN SCHEMES

	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$	$r=9$	$r=10$
SOABAM	1.000	1.000	0.999	0.998	0.993	0.985	0.974	0.952	0.938	0.900	0.849
FEEDF. [12]	1.000	1.000	1.000	1.000	0.998	0.986	0.957	0.899	0.814	0.689	0.539
FOABAM	1.000	0.973	0.833	0.674	0.536	0.398	0.317	0.251	0.220	0.158	0.130
GBAM [16]	1.000	0.944	0.829	0.702	0.581	0.466	0.369	0.337	0.281	0.206	0.175
OABAM [17]	1.000	0.889	0.566	0.422	0.306	0.275	0.248	0.231	0.214	0.193	0.161
ABAM [15]	1.000	0.804	0.527	0.294	0.186	0.167	0.151	0.136	0.129	0.122	0.116

TABLE V
COMPARISON OF ATTAINABLE b_x AND b_y AND CPU TIME REQUIRED OF SOABAM DESIGN BY ADAPTIVE LOCAL AND ADAPTIVE LEARNING RULES

	b_x	b_y	CPU Time (min)
Adaptive Local ($\lambda = 1.6$)	13002	18614	243.1
Adaptive ($\lambda = 1.6$)	4596	6665	12037.2

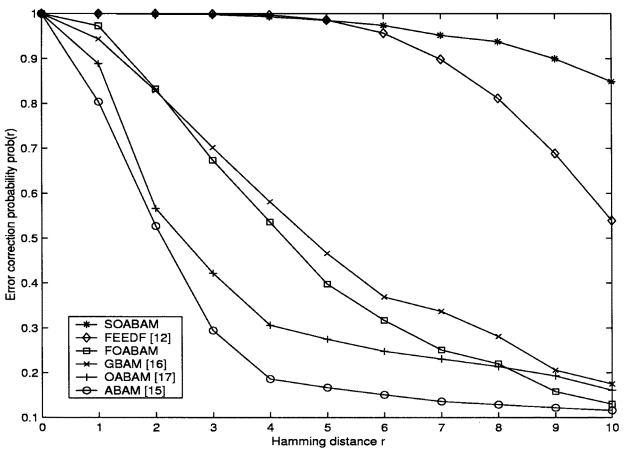


Fig. 1. Average error correction probability $\mathrm{prob}(r)$ versus hamming distance r of various BAM design schemes.

radius of good relaxation $\lambda = 1.6$, but requires 34 times larger in the number of entry updates. On the contrary, large relaxation $\lambda = 8.0$ needs 76% less entry updates of $\lambda = 1.6$ but can reach only about 44% of attraction radius obtained by $\lambda = 1.6$. Too large λ, 8.0 for example, leads to oscillatory in the learning steps; it could lead to fewer entry updates but becomes harder to converge to a good attraction radius, and even worse cannot converge at all sometimes. On the other hand, too small λ, 0.32 for example, leads to very small, i.e., conservative, step size in the learning steps; it requires larger entry updates in number but still suffers from the problem of unable to converge to a good attraction radius. Moreover, introducing local learning feature into the adaptive learning rule leads to the proposed *adaptive local learning rule* as given by

$$u_{ij_s}(t) = \Phi\left(u_{ij_s}(t-1) + \alpha_x(t) \cdot x_i^{m_s} y_{j_s}^{m_s}\right) \quad \forall\, i = 1, 2, \ldots, p \tag{18}$$

where $\alpha_x(t) = \lambda \cdot \alpha_0 \cdot \Delta b_x(t) = \lambda \cdot \Delta b_x(t)/p$, and m_s and j_s are the numbers that produce smallest $F_{x_j}^m$, $F_{x_{j_s}}^{m_s}(t)$, among all possible j and m; $1 \le j \le n$ and $1 \le m \le M$, at iteration t. The adaptive local learning can further improve the learning performance in the following two folds: the easiness and fastness to converge during the learning phase and further enlarging the attainable attraction radius. Table III shows the attraction radius is further enlarged to its twice owing to the addition of local learning to the adaptive training rule. The easier and faster to converge is eminent especially to a difficult (design) problem, as will be demonstrated in the SOABAM design below. In the FOABAM design here, this nature is not evident from our empirical experiment. For such a problem not too difficult to design, the expected advantage may perhaps have been offset by the additional minimum finding of $F_{x_i}^m$ and pocket algorithm.

B. Performance Comparisons of Various BAM Designs

1) Recall Rate Comparisons: Using the proposed adaptive local training rule to design the connection matrix U of an SOABAM from the Y- to X-layer having maximal attraction radius b_y. In a similar manner, an independent design loop can produce the connection matrix W from the X- to Y-layer having maximal attraction radius b_x. To compare the error correcting capability of a BAM design, several well known methods including ABAM [15], GBAM [16], feedforward BAM [12], and optimized ABAM [17] are used for comparison. The average error correction probability $\mathrm{prob}(r)$ defined below will be utilized as the performance index for this purpose. Let $P(X^m, r)$ denote the probability that a given distorted pattern of X^m which lies on the Hamming ball of radius r, $r \in \{0, 1, 2, \ldots\}$ can be successfully recalled to the corresponding pattern vector Y^m. Pattern vector X^m is said to be recalled successfully if within 100 times of two-way recalling a r-bit distorted X^m can reach and then remain fixed in its association pattern pair (X^m, Y^m) correctly. Otherwise, it is considered not to be recalled successfully. Let $P_x(r)$ be the average of $P(X^m, r)$ over all the 26 possible X^m. The average error correction probability $\mathrm{prob}(r)$ is the average of $P_x(r)$ and $P_y(r)$ computed. In this study, we generated respectively for each CGA characters 100 r-bit distorted X^m and Y^m randomly to compute $P_x(r)$ and $P_y(r)$. Table IV lists the $\mathrm{prob}(r)$ versus erroneous bit number r of different BAM design schemes, which is also shown in Fig. 1. From the table, our correction probability and the second best, feedforward BAM, are excellent for small numbers, up to six, of bit error, but our method outperforms the feedforward approach greatly afterwards, for example by 12% when $r = 8$ and by 31% when $r = 10$. Our new methods can remain as high as 98.6% recognition rate when the patterns are within 5-bit distortion, and can retain 90% recall rate tolerant up to a 9-bit error of prototype patterns.

In the following, the advantage gained from the first-order BAM model to a second-order one will be summarized. To the best of the author's knowledge, GBAM model is the best design scheme among numerous first-order BAM design algorithms developed. The first-order ABAM model designed by our proposed learning rule also produces a very good recall rate, which

TABLE VI
AVERAGE ERROR CORRECTION PROBABILITY $\mathrm{prob}(r)$ VERSUS HAMMING DISTANCE r OF ADAPTIVE LOCAL AND ADAPTIVE LEARNING RULES OF SOABAM DESIGN

	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$	$r=9$	$r=10$
Adaptive Local	1.000	1.000	0.999	0.998	0.993	0.985	0.974	0.952	0.929	0.900	0.849
Adaptive	1.000	1.000	0.999	0.995	0.990	0.978	0.957	0.936	0.912	0.879	0.825

is competitive to the best GBAM model. In terms of recall rate, our FOABAM design outperforms GBAM design when $r = 1$ and 2; GBAM is better than ours afterwards. As one can expect, our SOABAM design model outperforms the FOABAM design model in the recall rate by a great margin except for single bit error case, $r = 1$. This superiority implicitly implies that the proposed SOABAM model has greatly increased the memory capacity of a BAM model [7] and [16]. The new SOABAM design has suggested a way to build a memorized memory model for CGA letter set from almost incapable memorized models designed by several first-order BAM algorithms.

2) Basin of Attraction and Convergence Time Improvements of Adaptive Local Training Rule: The computer CPU time required to design SOABAM by the proposed adaptive local training rule with $\lambda = 1.6$ and by adaptive training rule with $\lambda = 1.6$ also are shown in Table V. From the table, it is seen that the CPU time, a Pentium III-800 PC also, required by the adaptive local training rule is only one-fiftieth of that by the adaptive training rule. SOABAM design by the adaptive learning is very time consuming, which require CPU time 12 037.2 mins to reach $b_x = 4596$ and $by = 6665$, in which these reachable maximal b_x and b_y are almost one-third of those obtained by the new local training. Comparison of recall rates of SOABAM design by the adaptive local rule and the adaptive perceptron learning are summarized in Table VI. Both learning schemes produce excellent recall capability, and the better one still goes to the proposed adaptive local learning rule.

VI. CONCLUSION

In this paper, we have derived the adaptive local training rule for SOABAM design. A necessary and sufficient condition that can guarantee the recall of all prototype pattern pairs is first derived. Then, a new theorem to design a SOABAM to enlarge the basin of attraction of training pattern pairs is proposed. Accordingly, a new adaptive local training rule to design a SOABAM is then formulated. The adaptive nature of the new learning rule is ascribed to a relaxation mechanism inclusion and the learning rate being proportional to the difference between the current value and the target value to be learned. That, every training iteration only updates the minimal weight entries which are most needful, signifies the local nature of our learning rule. In implementation aspect, our new training rule is further incorporated with pocket algorithm concept to save the best design connection weight matrix during the course of iteration learning procedure. The new adaptive local learning rule can not only design a SOABAM 50 times faster but also converge to an attraction radius three times larger than an adaptive only one. The simulation studies and comparisons on the familiar 7×7 pixel character set have proven the effectiveness of the new training scheme over existing BAM design approaches. Extending the proposed SOABAM design to HOABAM is possible and straightforward.

APPENDIX

Theorem 2: Suppose that $H_y = H(A, Y^m)$ is the Hamming distance between an input pattern A and a stored pattern Y^m, that $F^m_{x_k} \geq b_x > 0$ for $k = 1, 2, \ldots, n$, and that $|u_{ijk}| \leq D$. If

$$H_y = H(A, Y^m) \leq -\frac{p}{2} + \frac{1}{2}\sqrt{p^2 + \frac{b_x}{D}} \tag{19}$$

then the input A converges to X^m in one recall iteration.

Proof: Without loss of generality, suppose the different bits between A and Y^m are located at the first H_y neurons. That is, $A_k = -y^m_k$ if $k \leq H_y$ and $A_k = y^m_k$ if $k > H_y$. The next state A'_k is

$$\begin{aligned}
A'_k &= \mathrm{sgn}\left(\sum_{j=1}^{p}\sum_{i=1}^{p} A_j A_i u_{ijk}\right) \\
&= \mathrm{sgn}\left[\sum_{j=1}^{p} A_j \left(\sum_{i=1}^{H_y}(-y^m_i)u_{ijk} + \sum_{i=H_y+1}^{p} y^m_i u_{ijk}\right)\right] \\
&= \mathrm{sgn}\left[\sum_{j=1}^{p} A_j \left(\sum_{i=1}^{p} y^m_i u_{ijk} - 2\sum_{i=1}^{H_y} y^m_i u_{ijk}\right)\right] \\
&= \mathrm{sgn}\left[\sum_{j=1}^{H_y}(-y^m_j)\left(\sum_{i=1}^{p} y^m_i u_{ijk} - 2\sum_{i=1}^{H_y} y^m_i u_{ijk}\right)\right. \\
&\qquad \left. + \sum_{j=H_y+1}^{p} y^m_j \left(\sum_{i=1}^{p} y^m_i u_{ijk} - 2\sum_{i=1}^{H_y} y^m_i u_{ijk}\right)\right] \\
&= \mathrm{sgn}\left[\sum_{j=1}^{p} y^m_j \left(\sum_{i=1}^{p} y^m_i u_{ijk} - 2\sum_{i=1}^{H_y} y^m_i u_{ijk}\right)\right. \\
&\qquad \left. - 2\sum_{j=1}^{H_y} y^m_j \left(\sum_{i=1}^{p} y^m_i u_{ijk} - 2\sum_{i=1}^{H_y} y^m_i u_{ijk}\right)\right] \\
&= \mathrm{sgn}\left[\sum_{j=1}^{p}\sum_{i=1}^{p} y^m_j y^m_i u_{ijk} - 2\sum_{j=1}^{p}\sum_{i=1}^{H_y} y^m_j y^m_i u_{ijk}\right. \\
&\qquad \left. + 4\sum_{j=1}^{H_y}\sum_{i=1}^{H_y} y^m_j y^m_i u_{ijk} - 2\sum_{j=1}^{H_y}\sum_{i=1}^{p} y^m_j y^m_i u_{ijk}\right]. \\
&= x^m_k
\end{aligned}$$

The above equation can be satisfied if the following inequality holds

$$\begin{aligned}
&\left(\sum_{j=1}^{p}\sum_{i=1}^{p} y^m_j y^m_i u_{ijk}\right)\cdot x^m_k \\
&\quad \geq \left(2\sum_{j=1}^{p}\sum_{i=1}^{H_y} y^m_j y^m_i u_{ijk} - 4\sum_{j=1}^{H_y}\sum_{i=1}^{H_y} y^m_j y^m_i u_{ijk}\right. \\
&\qquad \left. + 2\sum_{j=1}^{H_y}\sum_{i=1}^{p} y^m_j y^m_i u_{ijk}\right)\cdot x^m_k.
\end{aligned} \tag{20}$$

From the assumptions, we can obtain the following two inequalities

$$F_{x_k}^m = \left(\sum_{i=1}^{p} \sum_{j=1}^{p} y_i^m y_j^m u_{ijk} \right) \cdot x_k^m \geq b_x$$

and

$$\begin{aligned} &\left(2 \sum_{j=1}^{p} \sum_{i=1}^{H_y} y_j^m y_i^m u_{ijk} - 4 \sum_{j=1}^{H_y} \sum_{i=1}^{H_y} y_j^m y_i^m u_{ijk} \right. \\ &\quad \left. + 2 \sum_{j=1}^{H_y} \sum_{i=1}^{p} y_j^m y_i^m u_{ijk} \right) \cdot x_k^m \\ &\quad \leq 2 \sum_{j=1}^{p} \sum_{i=1}^{H_y} |u_{ijk}| + 4 \sum_{i=1}^{H_y} \sum_{j=1}^{H_y} |u_{ijk}| + 2 \sum_{j=1}^{H_y} \sum_{i=1}^{p} |u_{ijk}| \\ &\quad \leq 4pH_yD + 4H_y^2D. \end{aligned}$$

Obviously, inequality (20) can be met if $b_x \geq 4pH_yD + 4H_y^2D$, which is given by assumption (19). This is because

$$\begin{aligned} F_{x_k}^m &\geq b_x \\ &\geq 4pH_yD + 4H_y^2D \\ &\geq \left(2 \sum_{j=1}^{p} \sum_{i=1}^{H_y} y_j^m y_i^m u_{ijk} - 4 \sum_{j=1}^{H_y} \sum_{i=1}^{H_y} y_j^m y_i^m u_{ijk} \right. \\ &\quad \left. + 2 \sum_{j=1}^{H_y} \sum_{i=1}^{p} y_j^m y_i^m u_{ijk} \right) \cdot x_k^m. \end{aligned}$$

In this case, the input A converges to X^m in one iteration. Q.E.D.

REFERENCES

[1] P. K. Simpson, *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations*. Elmsford, NY: Pergamon, 1990.

[2] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[3] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst. Man. Cybern.*, vol. 18, pp. 49–60, Jan./Feb. 1988.

[4] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Acad. Sci. U.S.A.*, vol. 79, 1982, pp. 2554–2558.

[5] Y. F. Wang, J. B. Cruz, Jr., and J. H. Mulligan, Jr., "On multiple training pairs for bidirectional associative memory," *IEEE Trans. Neural Networks*, vol. 1, pp. 275–276, Sept. 1991.

[6] M. H. Hassoun and J. Song, "Adaptive Ho-Kashyap rules for perceptron training," *IEEE Trans. Neural Networks*, vol. 3, pp. 51–61, Jan. 1992.

[7] X. Zhuang, Y. Huang, and S.-S. Chen, "Better learning for bidirectional associative memory," *Neural Netw.*, vol. 6, pp. 1131–1146, 1993.

[8] T. Wang, X. Zhuang, and X. Xing, "Designing bidirectional associative memories with optimal stability," *IEEE Trans. Syst. Man, Cybern. B*, vol. 24, pp. 778–790, May 1994.

[9] H. Oh and S. C. Kothari, "Adaptation of the relaxation method for learning in bidirection associative memory," *IEEE Trans. Neural Networks*, vol. 5, pp. 576–583, July 1994.

[10] K. Haines and R. Hecht-Niclson, "A BAM with increased information storage capacity," in *Proc. IJCNN 88*, vol. 1, 1988, pp. 181–190.

[11] H. Kang, "Multilayer associative neural network (MANN): Storage capacity versus perfect recall," *IEEE Trans. Neural Networks*, vol. 5, pp. 812–822, Sept. 1994.

[12] Y. Wu and D. A. Pados, "A feedforward bidirectional associative memory," *IEEE Trans. Neural Networks*, vol. 11, pp. 859–866, July 2000.

[13] C.-S. Leung, L.-W. Chan, and E. Lai, "Stability, capacity, and statistical dynamics of second-order bidirectional associative memories," *IEEE Trans. Syst. Man, Cybern.*, vol. 25, pp. 1414–1424, Oct. 1995.

[14] ——, "Stability and statistical properties of second-order bidirectional associative memory," *IEEE Trans. Neural Networks*, vol. 8, no. 10, pp. 267–277, Mar. 1997.

[15] Z. B. Xu, Y. Leung, and X.-W. He, "Asymmetric bidirectional associative memories," *IEEE Trans. Syst. Man, Cybern. B*, vol. 24, pp. 1558–1564, Oct. 1994.

[16] H. Shi, Y. Zhao, and X. Zhuang, "A general model for bidirectional associative memories," *IEEE Trans. on Syst. Man, Cybern. B*, vol. 28, pp. 511–519, Aug. 1998.

[17] J. Park, C.-H. Kwon, and D. Park, "An optimization-based design procedure for asymmetric bidirectional associative memories," *IEEE Trans. Neural Networks*, vol. 12, pp. 169–170, Jan. 2001.

[18] D. Psaltis, C. H. Park, and J. Hong, "Higher order associative memories and their optical implementation," *Neural Networks*, vol. 1, pp. 149–163, 1988.

[19] K. Y. Hsu, H. Y. Li, and D. Psaltis, "Holographic implementation of a fully connected neural network," *Proc. IEEE*, vol. 78, no. 10, pp. 1637–1645, Oct. 1990.

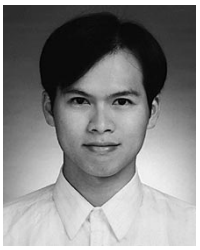
[20] S. J. Perantonis and P. J. G. Lisboa, "Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 241–251, Mar. 1992.

[21] R. Rosenblatt, *Principles of Neurodynamics*. New York: Spartan, 1959.

[22] S. I. Gallant, "Perceptron-based learning algorithms," *IEEE Trans. Neural Networks*, vol. 1, pp. 179–191, 1990.

Jyh-Yeong Chang received the B.S. degree in control engineering in 1976 and the M.S. degree in electronic engineering in 1980, both from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, R.O.C. He received the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, in 1987.

From 1976 to 1978, and 1980 to 1982, he was a Research Fellow at Chung Shan Institute of Science and Technology (CSIST), Taiwan. Since 1987, he has been an Associate Professor in the Department of Electrical and Control Engineering at NCTU. His research interests include fuzzy sets and systems, image processing, pattern recognition, and neural network applications.



Chien-Wen Cho received the B.S. degree in control engineering in 1996 from National Chiao Tung University (NCTU), Hsinchu, Taiwan, R.O.C. He is currently working toward the Ph.D. degree in the Department of Electrical and Control Engineering at National Chiao Tung University.

He was a Research and Design Engineer and Manager in the CNC field for five years with Victor Machinery Co. in Taiwan. His research interests include pattern recognition, image processing, fuzzy neural theory, and data mining applications.