

Research Article

Encoding Static and Temporal Patterns with a Bidirectional Heteroassociative Memory

Sylvain Chartier¹ and Mounir Boukadoum²

¹ School of Psychology, University of Ottawa, 136 Jean-Jacques Lussier, Ottawa, ON, Canada K1N 6N5

² Département d'informatique, Université du Québec à Montréal, Case postale 8888, Succursale Centre-ville, Montréal, QC, Canada H3C 3P8

Correspondence should be addressed to Sylvain Chartier, schartie@uottawa.ca

Received 15 September 2010; Accepted 8 December 2010

Academic Editor: Haydar Akca

Copyright © 2011 S. Chartier and M. Boukadoum. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Brain-inspired, artificial neural network approach offers the ability to develop attractors for each pattern if feedback connections are allowed. It also exhibits great stability and adaptability with regards to noise and pattern degradation and can perform generalization tasks. In particular, the Bidirectional Associative Memory (BAM) model has shown great promise for pattern recognition for its capacity to be trained using a supervised or unsupervised scheme. This paper describes such a BAM, one that can encode patterns of real and binary values, perform multistep pattern recognition of variable-size time series and accomplish many-to-one associations. Moreover, it will be shown that the BAM can be generalized to multiple associative memories, and that it can be used to store associations from multiple sources as well. The various behaviors are the result of only topological rearrangements, and the same learning and transmission functions are kept constant throughout the models. Therefore, a consistent architecture is used for different tasks, thereby increasing its practical appeal and modeling importance. Simulations show the BAM's various capacities, by using several types of encoding and recall situations.

1. Introduction

Being able to recognize and recall patterns of various natures and in different contexts is something that human beings accomplish routinely and with little effort. But these tasks are difficult to reproduce by artificial intelligent systems. A successful approach has consisted of distributing information over parallel networks of processing units, as done in biological neural networks. This brain-inspired, artificial neural network approach offers the ability to develop attractors for each pattern if feedback connections are allowed (e.g., [1, 2]). It also exhibits great stability and adaptability with regard to noise and pattern degradation, and can perform generalization tasks. In particular, the Bidirectional Associative

Memory (BAM) model has shown great promise for pattern recognition for its capacity to be trained using a supervised or unsupervised scheme [3]. Given n bipolar column-matrix pairs, $(\mathbf{X}_1, \mathbf{Y}_1), (\mathbf{X}_2, \mathbf{Y}_2), \dots, (\mathbf{X}_i, \mathbf{Y}_i), \dots, (\mathbf{X}_n, \mathbf{Y}_n)$, BAM learning is accomplished with a simple Hebbian rule, according to the equation:

$$\mathbf{W} = (\mathbf{Y}_1 \mathbf{X}_1^T) + (\mathbf{Y}_2 \mathbf{X}_2^T) + \dots + (\mathbf{Y}_n \mathbf{X}_n^T) = \mathbf{Y} \mathbf{X}^T. \quad (1.1)$$

In this expression, \mathbf{X} and \mathbf{Y} are matrices that represent the sets of bipolar pairs to be associated, and \mathbf{W} is the weight matrix. To assure perfect storage and retrieval, the input patterns needed to be orthogonal ($\mathbf{X}^T \mathbf{X} = \mathbf{K}$, with typically $\mathbf{K} = \mathbf{I}$ the identity matrix; [4]). In that case, all the positive eigenvalues of the weight matrix will be equal and, when an input is presented for recall, the correct output can be retrieved. For example, if the input \mathbf{x} represents the encoded pattern \mathbf{X}_i , then the output will be given by

$$\begin{aligned} \mathbf{y} &= \mathbf{W} \mathbf{x} = \mathbf{Y} \mathbf{X}^T \mathbf{x} \\ &= \mathbf{Y}_1 \mathbf{X}_1^T \mathbf{x} + \mathbf{Y}_2 \mathbf{X}_2^T \mathbf{x} + \dots + \mathbf{Y}_i \mathbf{X}_i^T \mathbf{x} + \dots + \mathbf{Y}_n \mathbf{X}_n^T \mathbf{x} \\ &= \mathbf{Y}_1 0 + \mathbf{Y}_2 0 + \dots + \mathbf{Y}_i 1 \dots + \mathbf{Y}_n 0 \\ &= \mathbf{Y}_i. \end{aligned} \quad (1.2)$$

The \mathbf{y} output will thus correspond to the \mathbf{Y}_i encoded stimulus. Equation (1.1) uses a one-shot learning process since Hebbian association is strictly additive. A more natural learning procedure would make the learning incremental but, then, the weight matrix would grow unbounded with the repetition of the input stimuli during learning. In addition, the eigenvalues will reflect the frequency of presentation of each stimulus. This property may be acceptable for orthogonal patterns, but it leads to disastrous results when the patterns are correlated. In that case, the weight matrix will be dominated by its first eigenvalue, and this will result in recalling the same pattern whatever the input. For correlated patterns, a compromise is to use a one-shot learning rule to limit the domination of the first eigenvalue, and to use a recurrent nonlinear transmission function to allow the network to filter out the different patterns during recall. Kosko's BAM effectively used a signum transmission function to recall noisy patterns, despite the fact that the weight matrix developed by using (1.1) is not optimal. The nonlinear transmission function usually used by the BAM network is expressed by the following equations:

$$\begin{aligned} \mathbf{y}(t+1) &= f(\mathbf{W} \mathbf{x}(t)), \\ \mathbf{x}(t+1) &= f(\mathbf{W}^T \mathbf{y}(t)), \end{aligned} \quad (1.3)$$

where $f()$ is the signum function and t is a given discrete time step. Initially, BAMs had poor storage capacity, could only store binary patterns, and were limited to static inputs. Nowadays, storage and recall performance have much improved; BAMs can learn and recall binary patterns with good performance (e.g., [5–20]), and that capability has been extended to real-valued patterns (e.g., [7, 8, 21–24]). Attention has also been given to learning and retrieving temporal sequences (see [25] for a review). These multistep associations have been

studied essentially with gradient descent techniques (e.g., [26, 27]), competitive techniques (e.g., [28]) or complex architectures (e.g., [29, 30]). Few authors have studied this type of association in the case of simple BAM networks (e.g., [7, 31–33]).

In this paper, focus will be drawn on BAM properties for different kinds of associations. At first analysis of the transmission function will be studied both analytically and numerically. It will be followed by simulations of the classic case of one-to-one association. The next section will present many-to-one association that allows mean extraction from the presentation of different exemplars. Then, we focus on temporal association. In this case, both limit cycle and steady-state behaviors will be presented. Finally, more complex architectures will be explored by using the Multidirectional Associative Memory (MAM) architecture while keeping the same learning and transmission functions.

2. Analysis of the Transmission Function

This section describes how the transmission function is derived from both analytical and numerical results of the one-dimensional cubic function.

2.1. One-Dimensional Symmetric Setting

The transmission function used in the model is based on the classic Verhulst equation [34]. Since, this quadratic function has only one stable fixed-point, it is extended to a cubic form described by the dynamic equation

$$\frac{dx}{dt} = f(x) = rx - x^3, \quad (2.1)$$

where x represents the input value, and r is a free parameter that affects the equilibrium states of (2.1). Figure 1 illustrates its shape for $r = 1$. The fixed-points are the roots of $f(x) = 0$. For example, if $r = 1$ the corresponding fixed-points are $\bar{x} = -1, 0$, and 1 . The stability properties of the fixed-points are determined by computing the derivative of (2.1):

$$f'(x) = r - 3x^2. \quad (2.2)$$

If the derivative at a given fixed-point is greater than zero, then a small perturbation results in growth; else, if the derivative is negative, then a small perturbation results in decay. The first situation represents an unstable fixed-point whereas the second represents a stable one. In the previous example, both $\bar{x} = 1$ and $\bar{x} = -1$ are stable fixed-points while $\bar{x} = 0$ is an unstable fixed-point. This is illustrated in Figure 1 by filled and empty circles, respectively.

Another way to visualize the dynamics of a recurrent neural network is based on the physical idea of energy [1]. The energy function, noted $E(x)$, can be defined by

$$-\frac{dE}{dx} = \frac{dx}{dt} \quad (2.3)$$

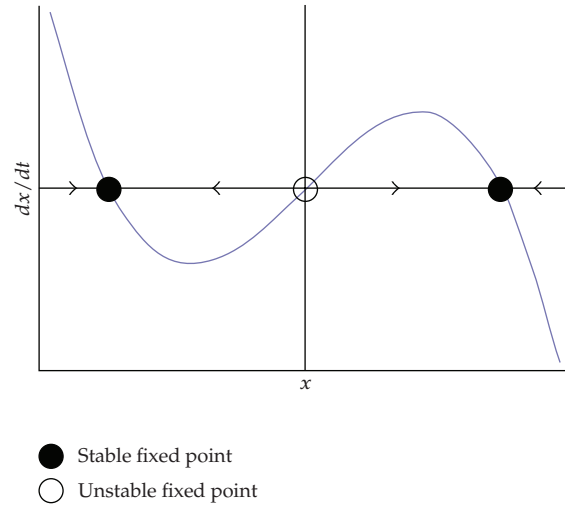


Figure 1: Transmission function when $r = 1$ showing two stable and one unstable fixed-points.

where the negative sign indicates that the state vector moves downhill in the energy landscape. Using the chain rule, it follows from (2.3) that

$$\frac{dE}{dt} = \frac{dE}{dx} \left(-\frac{dE}{dx} \right) = -\left(\frac{dE}{dx} \right)^2 \leq 0. \quad (2.4)$$

Thus, $E(t)$ decreases along trajectories or, in other words, the state vector globally converges towards lower energy states. Equilibrium occurs at locations of the vector field where local minima correspond to stable fixed-points, and local maxima correspond to unstable ones. To find them, we need to find $E(x)$ such that

$$-\frac{dE}{dx} = rx - x^3. \quad (2.5)$$

The general solution is

$$E(x) = -\frac{rx^2}{2} + \frac{x^4}{4} + C, \quad (2.6)$$

where C is an arbitrary constant (we use $C = 0$ for convenience). Figure 2 illustrates the energy function when $r = 1$. The system exhibits a double-well potential with two stable equilibrium points ($\bar{x} = -1$ and $\bar{x} = 1$).

The r parameter plays an important role in determining the number and positions of the fixed-points. Figure 3 illustrates the situation. For r less than zero, the system has only one (stable) fixed-point, $\bar{x} = 0$; for r greater than zero, there are three fixed-points: $\bar{x} = 0$ and $\bar{x} = \pm\sqrt{r}$, of which the first one is unstable and the other two are stable. Finally, for r equal to zero, we have a pitchfork bifurcation point. We deduce from the preceding that we must have $r > 0$ for the system to store binary stimuli.

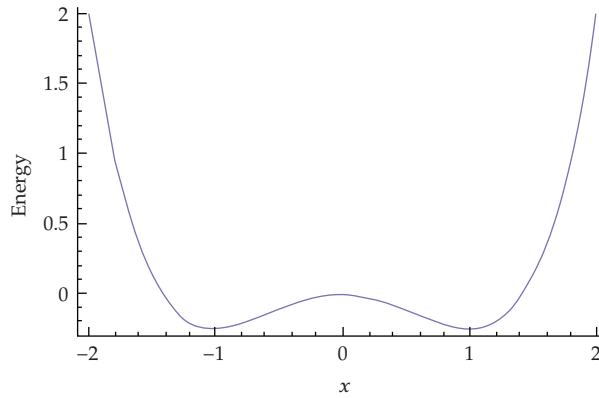


Figure 2: Energy landscape of a 1-dimensional system with $r = 1$.

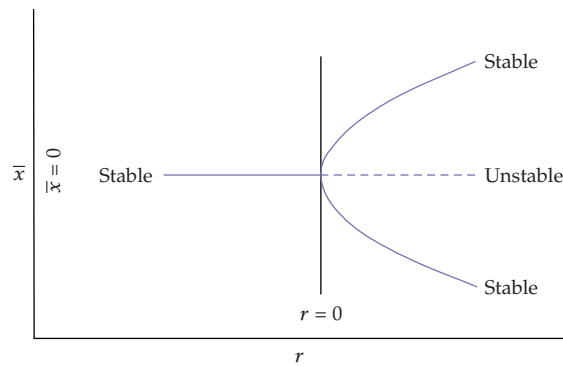


Figure 3: Phase diagram indicating the presence of a bifurcation for $r = 0$.

2.2. M-Dimensional Symmetric Setting

In a m -dimensional space, (2.1) takes a vector form and becomes

$$\frac{dx}{dt} = f(x) = r * x - x^3, \tag{2.7}$$

where, $x = (x_1, x_2, \dots, x_m)^T$, and $r = (r_1, r_2, \dots, r_m)^T$ and m is the network's dimension. When the weight matrix is diagonal, the system is uncoupled and the analysis is straightforward. As in the one-dimensional system, the fixed-points are defined by the roots of (2.7). Their stability properties of each one are determined by finding the eigenvalues of its Jacobian matrix. Depending on the eigenvalues, different types of fixed-point behaviors are obtained. For example, if $r = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ there will be nine fixed-points as illustrated in Figure 4. Four of them are stable: $(-1, -1)$, $(-1, 1)$, $(1, -1)$, and $(1, 1)$; they are indicated by filled circles. The other five are unstable and indicated by empty circles. Here also, the stability of the fixed-point can

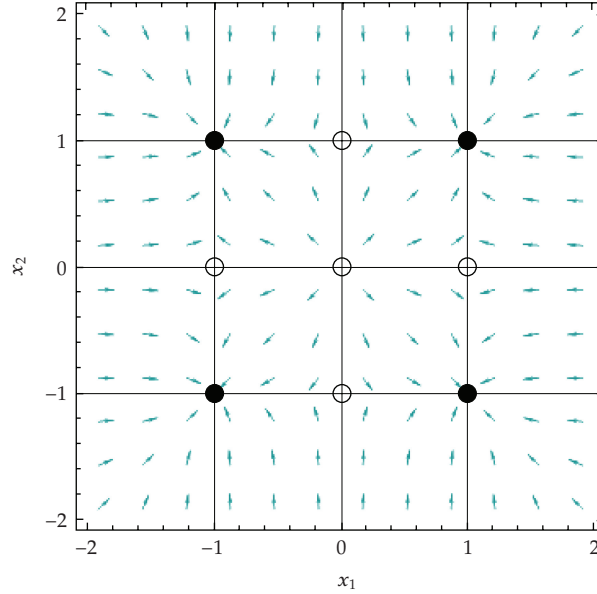


Figure 4: Phase portrait of a two-dimensional system with $\mathbf{r} = [1, 1]^T$. The stable fixed-points are represented by filled circles and the unstable ones by empty circles.

be determined from the energy of the system. In matrix notation, the energy function is

$$E(x) = -r * \frac{\mathbf{x}^T \mathbf{x}}{2} + \frac{\mathbf{x}^T \mathbf{x}^3}{4}. \quad (2.8)$$

In the case of bipolar pattern and if \mathbf{r} is set to $[1, 1]^T$, then the energy of the system will be equal to $-m/4$.

The function is plotted in Figure 5 for a two-dimensional system, using the previous values of \mathbf{r} . The stable fixed-points correspond to the local minima in the plot and they partition the recall space into four equal wells. For example, if the desired correct output (fixed-point) is $\bar{\mathbf{x}} = [1, 1]^T$, the probability that this fixed-point attracts a uniformly distributed pattern \mathbf{x} whose elements $x_i \in [-1, 1]$ is 25%.

2.3. Numerical Approximation Using Euler's Method

From the previous analysis, two stable fixed-points exist when r_i is positive. For simplicity, if all the element of \mathbf{r} are set to one, then (2.7) becomes

$$\frac{d\mathbf{x}}{dt} = \mathbf{x} - \mathbf{x}^3. \quad (2.9)$$

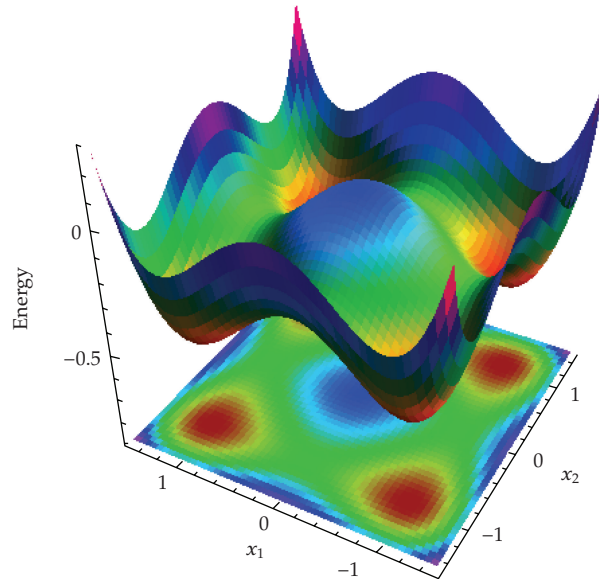


Figure 5: Energy landscape of the same two-dimensional system as in Figure 4.

This last differential equation can then be approximated in discrete time by using the Euler method:

$$\mathbf{x}(t+1) = \delta(\mathbf{x}(t) - \mathbf{x}^3(t)) + \mathbf{x}(t), \quad (2.10)$$

where δ is small positive constant and t is a given discrete time step. Rearranging the terms yields

$$\mathbf{x}(t+1) = (\delta + 1)\mathbf{x}(t) - \delta\mathbf{x}^3(t). \quad (2.11)$$

However, as it is, (2.11) does not reflect the impact of weight connections. To take into account that the connection strength between units is modified by a learning rule, we pose $\mathbf{x}(t) = \mathbf{W}\mathbf{x}(t) = \mathbf{a}(t)$, where \mathbf{W} represents the weight connections. Equation (2.11) then becomes

$$\mathbf{x}(t+1) = (\delta + 1)\mathbf{a}(t) - \delta\mathbf{a}^3(t). \quad (2.12)$$

This last function is illustrated in Figure 6. As the figure shows, $\mathbf{a}_i(t) = 1$, then $\mathbf{x}_i(t+1)$ will have the same value of 1.

The slope of the derivative of the transmission function will determine its stability. In our case, it is desired that the fixed-points have a stable monotonic approach. Therefore, the

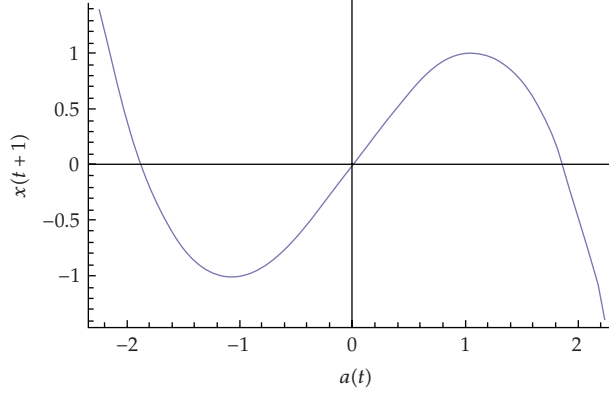


Figure 6: Transmission function for a value of $\delta = 0.4$.

slope must be positive and less than one [35]:

$$0 < \frac{dx(t+1)}{da(t)} < 1 = 0 < (\delta + 1) - 3\delta a^2(t) < 1. \quad (2.13)$$

This condition is satisfied when $0 < \delta < 0.5$ for bipolar stimuli. In that case, $\mathbf{a} = \mathbf{W}\bar{\mathbf{x}}(t) = \pm 1$. Another way to analyze the behavior of the network for the whole range of δ is by performing a Lyapunov Analysis. This analysis will allow us to discriminate between aperiodic (chaos) attractor from periodic one. The Lyapunov exponent for the case of a one-dimensional network is approximated by

$$\lambda \approx \frac{1}{T} \sum_{t=1}^T \log \left| \frac{dx(t+1)}{dx(t)} \right|, \quad (2.14)$$

where T is the number of network iterations, set to 10,000 to establish the approximation. Again, for simplicity, we perform the analysis in the case of independent units. In other words, the derivative term is obtained from (2.13) when $\mathbf{W} = \mathbf{I}$ (for simplicity), so that λ is given by

$$\lambda \approx \frac{1}{T} \sum_{t=1}^T \log \left| 1 + \delta - 3\delta x^2(t) \right|. \quad (2.15)$$

In order to estimate the range of values for a given period, a bifurcation diagram can be performed. Figure 7 illustrates both Lyapunov and bifurcation analysis and shows that for values of δ less than 1.0, the network converges to a fixed-point attractor. However, at higher values (e.g., 1.6), the network may converge to an aperiodic attractor.

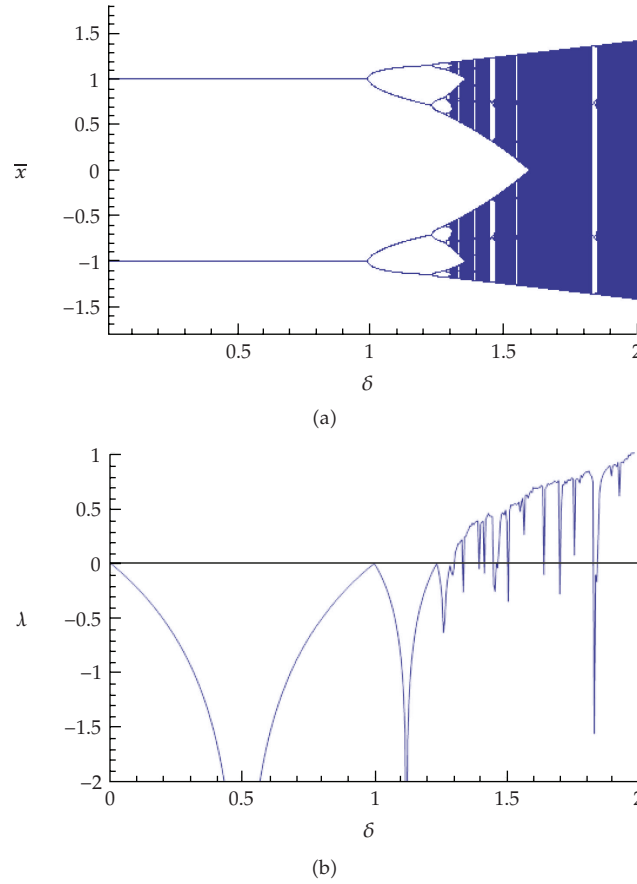


Figure 7: Bifurcation and Lyapunov exponent diagrams as a function of δ .

2.4. Numerical Approximation Using Forth-Order Runge Kutta's Method

Another numerical approximation of the network's dynamics is by using the Forth-Order Runge Kutta (FORK) method. Contrarily to Euler's method, FORK uses an average estimation to approximate the next time step

$$\begin{aligned}
 x(t+1) &= x(t) + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}, \\
 k_1 &= (rx(t) - x(t)^3)\Delta, \\
 k_2 &= (r(x(t) + 0.5k_1) - (x(t) + 0.5k_1)^3)\Delta, \\
 k_3 &= (r(x(t) + 0.5k_2) - (x(t) + 0.5k_2)^3)\Delta, \\
 k_4 &= (r(x(t) + k_3) - (x(t) + k_3)^3)\Delta,
 \end{aligned} \tag{2.16}$$

where Δ is a small approximation parameter. Again, to take into account weight connections, we pose that $\mathbf{x}(t) = \mathbf{W}\mathbf{x}(t) = \mathbf{a}(t)$. Equation (2.16) thus becomes

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{a}(t) + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}, \\ k_1 &= \left(r(\mathbf{a}(t)) - \mathbf{a}(t)^3 \right) \Delta, \\ k_2 &= \left(r(\mathbf{a}(t) + 0.5k_1) - (\mathbf{a}(t) + 0.5k_1)^3 \right) \Delta, \\ k_3 &= \left(r(\mathbf{a}(t) + 0.5k_2) - (\mathbf{a}(t) + 0.5k_2)^3 \right) \Delta, \\ k_4 &= \left(r(\mathbf{a}(t) + k_3) - (\mathbf{a}(t) + k_3)^3 \right) \Delta. \end{aligned} \quad (2.17)$$

This last function is illustrated in Figure 8. As the figure shows, if $\mathbf{a}_i(t) = 1$, then $\mathbf{x}_i(t+1)$ will have the same value of 1. Again, like any nonlinear dynamic system, to guarantee that a given output converges to a fixed-point $\bar{\mathbf{x}}(t)$, the slope of the derivative of the transmission function must be positive and less than one [35]:

$$0 < \frac{d\mathbf{x}(t+1)}{d\mathbf{a}(t)} < 1. \quad (2.18)$$

This condition is satisfied when $0 < \Delta < 0.135$ for bipolar stimuli. In that case, $\mathbf{a} = \mathbf{W}\bar{\mathbf{x}}(t) = \pm 1$. Here also, bifurcation and Lyapunov exponent diagrams were performed. Figure 8 shows that if the value of Δ is lower than 1.7, the network will converge to a fixed-point. However, if the value is too high then a given input will converge to an aperiodic attractor.

2.5. Performance Comparison between Euler and FORK Approximations

The purpose of this simulation was to compare the performance between Euler and FORK approximations. Although FORK gives a more precise approximation of ordinary differential equation (2.7), we need to evaluate if a better approximation translates into a better radius of attraction.

2.5.1. Methodology

A low and a high memory load was used as a basis of comparison. For the low memory load situation, the first 10 correlated patterns were associated together ($a - j$). The patterns are 7×7 pixels images of alphabetic characters where a white pixel is given the value -1 and a black pixel the value 1. This led to a memory load equal to 20% (10/49) of the 49-dimensional space capacity. Normally, such a high load value cannot be handled by Kosko's BAM which is about 14%. For the high memory load situation, the associations were extended to all the 26 correlated patterns ($a - z$). This last situation led to a memory load equal to

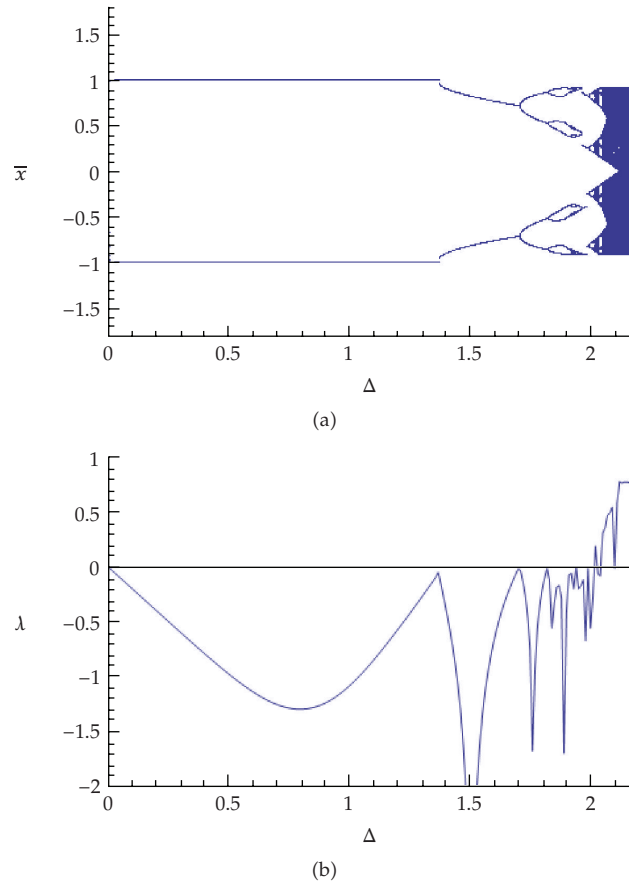


Figure 8: Bifurcation and Lyapunov exponent diagrams as a function of Δ .

53% (26/49). Usually, around 50% is about the maximum load an optimized Hebbian-type associative memory can store without major performance decline [36]. Figure 9 illustrates the stimuli used for the simulation. The images were converted to vectors of 49 dimensions before being input to the network.

Both methods were tested with their output parameter set to 0.05 and 0.025. These values are much lower than the theoretical limit that was found analytically. The simulation results (not reported) show that a higher value makes the model unable to associate the patterns. The associations between patterns were accomplished using the learning rule (3.4) and (3.5) that will be described in the next section.

After associating the desired pattern pairs, the radiuses of attraction obtained by Euler and FORK were evaluated for noisy recall tasks. The first task consisted of recalling noisy patterns obtained by generating random normally distributed vectors that were added to the patterns. Each noisy pattern vector was distributed with a mean of 0 and a standard deviation of π (where π represents the desired proportion of noise). For the simulation, π varied from 0.1 to 1.0. The second task was to recall the correct associated stimulus from a noisy input obtained by randomly flipping pixels in the input pattern. The number of pixel flips varied from 0 to 10, thus corresponding to a noise proportion of 0 to 20%.

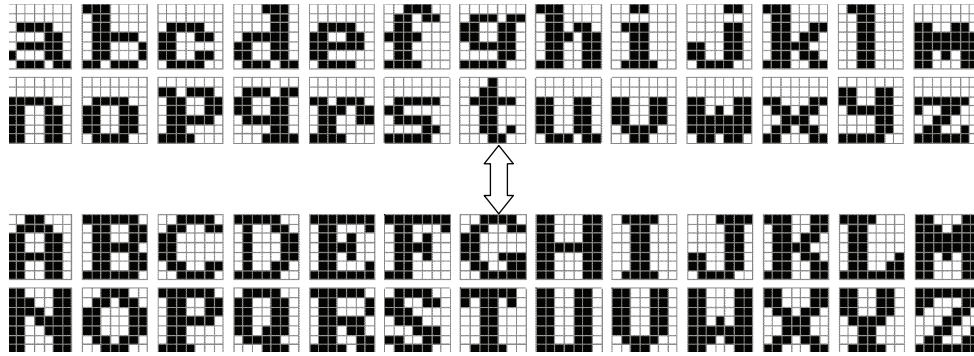


Figure 9: Pattern set used for numerical evaluations.

2.5.2. Results

Figure 10 illustrates the various performances. If the memory load is low (10 patterns over 49 pixels) as shown in the left column, then Euler's approximation has a slight advantage over FORK for random pixel flipping. This advantage is increased if the memory load is high (26 patterns over 49 pixels) as shown in the right column. Moreover, the higher the value of the output parameter, the higher the performance will be. However, if the output parameter is set to high, then the network might not converged to a fixed-point. Therefore, the choice for an output transmission will be based on Euler methods.

3. Model Description

3.1. Architecture

The proposed BAM architecture is similar to the one proposed by Hassoun [37] as illustrated in Figure 11, where $x(0)$ and $y(0)$ represent the initial input-states (stimuli); t is the number of iterations over the network; and W and V are weight matrices. The network is composed of two Hopfield-like neural networks interconnected in head-to-toe fashion. These interconnected layers allow a recurrent flow of information that is processed in a bidirectional way. The y -layer returns information to the x -layer and vice versa. If similar patterns are presented at both layers, then the network act like an autoassociative memory and if the patterns at the y -layer are associated with different ones in the x -layer, then the network acts like a heteroassociative memory [3]. As a result, it encompasses both unsupervised (self-supervised) and supervised learning. In this particular model, the two layers can be of different dimensions and contrary to usual BAM designs, the weight matrix from one side is not necessarily the transpose of that from the other side.

3.2. Transmission Function

Based, on the numerical results of the Section 2.5.2. , the transmission function is expressed by the following equations. The input activations to each of the y - and x -layers are computed

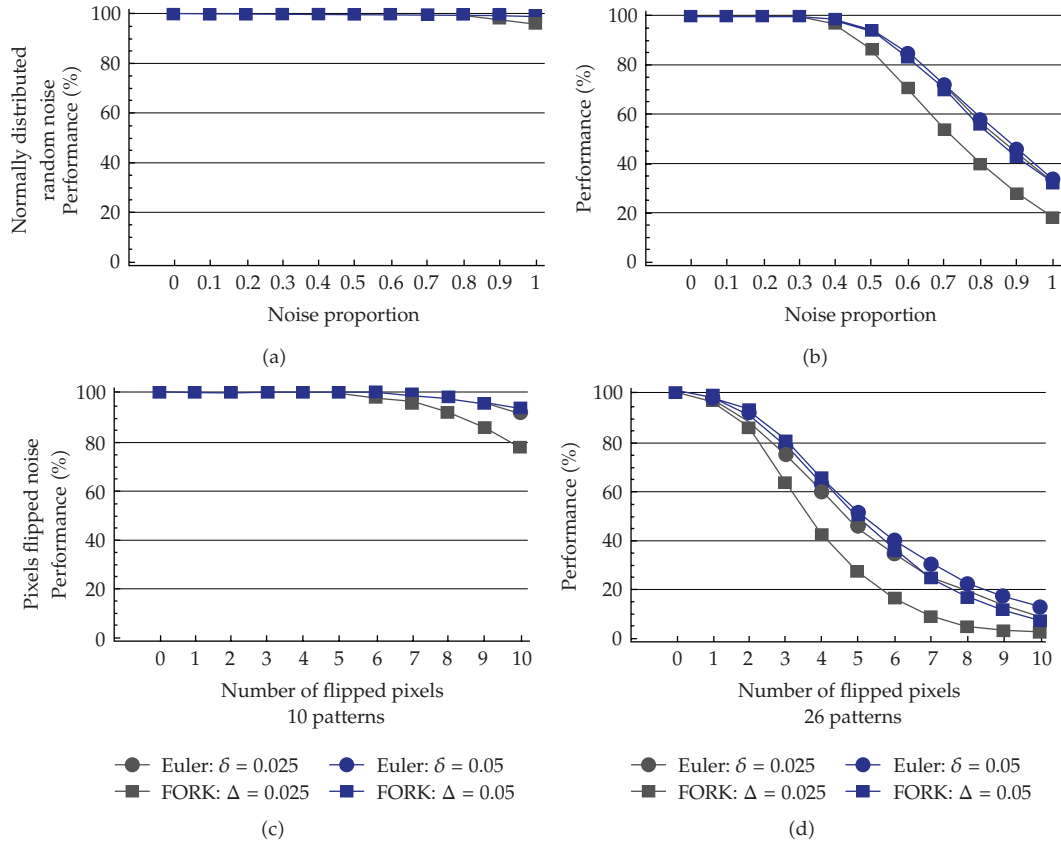


Figure 10: Performance comparison between Euler and FORK methods for different output parameters (0.05 and 0.025). Upper left panel low memory load under normally distributed noise Upper right high memory load under normally distributed noise. Lower left low memory load under pixels flipped noise. Lower right high memory load under pixels flipped noise.

as follows:

$$\begin{aligned} \mathbf{a}(t) &= \mathbf{W}\mathbf{x}(t), \\ \mathbf{b}(t) &= \mathbf{V}\mathbf{y}(t). \end{aligned} \tag{3.1}$$

The outputs are then obtained using Euler’s approximation (2.12) defined by the following equations:

$$\begin{aligned} \forall i, \dots, N, \quad \mathbf{y}_i(t+1) &= (1 + \delta)\mathbf{a}_i(t) - \delta\mathbf{a}_i^3(t), \\ \forall i, \dots, M, \quad \mathbf{x}_i(t+1) &= (1 + \delta)\mathbf{b}_i(t) - \delta\mathbf{b}_i^3(t), \end{aligned} \tag{3.2}$$

where N and M are the number of units in each layer, i is the index of the respective vector element, $\mathbf{y}(t+1)$ and $\mathbf{x}(t+1)$ represent the layers’ contents at time $t+1$, and δ is a general output parameter. The shape of this function is illustrated in Figure 6. This function has the

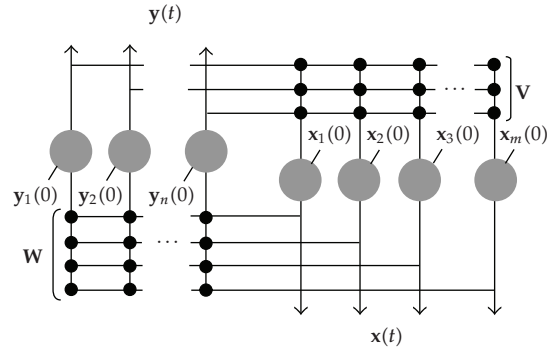


Figure 11: Architecture illustration of the bidirectional associative memory.

advantage of exhibiting continuous-valued (or gray-level) attractor behavior (for a detailed example, see [7]) when used in a recurrent network. Such properties contrast with standard nonlinear transmission functions, which only exhibit bipolar attractor behavior (e.g., [3]).

3.3. Learning

The network tries to solve the following nonlinear constraints:

$$\begin{aligned} \mathbf{X} &= f(\mathbf{V}\mathbf{Y}), \\ \mathbf{Y} &= f(\mathbf{W}\mathbf{X}), \end{aligned} \quad (3.3)$$

where f is the transmission function defined previously (3.2). The form of the constraints and the recurrent nature of the underlying network call for a learning process that is executed online: the weights are modified in function of the input and the obtained output. In addition, since incremental learning is favored, the network must then be able to self-converge. As a result, the learning is based on the time difference Hebbian association [7, 38–40]. It is formally expressed by the following equations:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \eta(\mathbf{y}(0) - \mathbf{y}(t))(\mathbf{x}(0) + \mathbf{x}(t))^T, \quad (3.4)$$

$$\mathbf{V}(k+1) = \mathbf{V}(k) + \eta(\mathbf{x}(0) - \mathbf{x}(t))(\mathbf{y}(0) + \mathbf{y}(t))^T, \quad (3.5)$$

where η represents the learning parameter. In (3.4) and (3.5), the weight updates follow this general procedure: first, initial inputs $\mathbf{x}(0)$ and $\mathbf{y}(0)$ are fed to the network, then, those inputs are iterated t times through the network (Figure 1). This results in outputs $\mathbf{x}(t)$ and $\mathbf{y}(t)$ that are used for the weight updates. Therefore, the weights will self-stabilize when the feedback is the same as the initial inputs ($\mathbf{y}(t) = \mathbf{y}(0)$ and $\mathbf{x}(t) = \mathbf{x}(0)$); in other words, when the network has developed fixed-points. This contrasts with most BAMs, where the learning is performed solely on the activation (offline). Learning convergence is a function of the value of the learning parameter η . For simplicity, we assumed that both input and output are the same ($\mathbf{y}(0) = \mathbf{x}(0)$). Therefore, to find the maximum value that the learning parameter can be

set to we need to find the derivation of learning equation when the slope is positive and then solve it for η [35]

$$\frac{\partial \mathbf{W}(k+1)}{\partial \mathbf{W}(k)} = \mathbf{W}(k) + \eta(\mathbf{x}(0) - \mathbf{x}(t))(\mathbf{x}(0) + \mathbf{x}(t))^T > 0. \quad (3.6)$$

If $t = 1$ and in the case of a one-dimensional pattern in a one-dimensional network, the situation simplifies to

$$\frac{\partial w(k+1)}{\partial w(k)} = w(k) + \eta \left(1 - (\delta + 1)w(k) + \delta(w(k))^3 \right)^2 > 0. \quad (3.7)$$

In this case, the network will converges when $w(k+1) = w(k) = 1$ and the solution will be

$$\eta < \frac{1}{2(1-2\delta)}, \quad \delta \neq \frac{1}{2}. \quad (3.8)$$

Just as any network, as the network increases in dimensionality, η must be set at lower values. Therefore, in the case of BAM of M and N dimensions, the learning parameter must be set according to

$$\eta < \frac{1}{2(1-2\delta) \text{Max}[M, N]}, \quad \delta \neq \frac{1}{2}. \quad (3.9)$$

4. Simulations

The following simulations will first provide a numerical example to illustrate how the learning and recall are performed using a small network. The next simulation then reproduces a classic BAM one-to-one association's task. Finally, the third simulation extends the association property of the model to many-to-one association's task.

4.1. Numerical Example

The first simulation uses a toy example to show in details how the learning and recall processes are performed.

4.1.1. Learning

For this simulation the transmission function parameter (δ) was set to 0.1 and the learning parameter (η) was set to 0.01. Also, to limit the simulation time, the number of output iterations before each weight matrix update was set to $t = 1$ for all simulations. Learning

was carried out according to the following procedure:

Weight initialization at zero;

- (1) random selection of a pattern pair,
- (2) computation of $\mathbf{x}(t)$ and $\mathbf{y}(t)$ according to the transmission function (3.2),
- (3) computation of the weight matrix (\mathbf{W} and \mathbf{V}) update according to (3.4) and (3.5),
- (4) repetition of steps (1) to (3) until the weight matrices converge.

The stimuli pairs for the simulation are given by

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ -1 & 1 \end{bmatrix}. \quad (4.1)$$

First Learning Trial

Assume that the second pair is randomly selected. Then,

$$\mathbf{x}(0) = [1 \ 1 \ 1 \ -1]^T, \quad \mathbf{y}(0) = [1 \ -1 \ 1]^T \quad (4.2)$$

and (3.2) yield zero-vectors for $\mathbf{y}(1)$ and $\mathbf{x}(1)$ since the weight connections are initially set at zero:

$$\mathbf{x}(1) = [0 \ 0 \ 0 \ 0]^T, \quad \mathbf{y}(1) = [0 \ 0 \ 0]^T. \quad (4.3)$$

Using (3.4) and (3.5), the weight matrices are updated and their values are

$$\mathbf{W}(1) = \begin{bmatrix} 0.01 & 0.01 & 0.01 & -0.01 \\ -0.01 & -0.01 & -0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & -0.01 \end{bmatrix}, \quad \mathbf{V}(1) = \begin{bmatrix} 0.01 & -0.01 & 0.01 \\ 0.01 & -0.01 & 0.01 \\ 0.01 & -0.01 & 0.01 \\ -0.01 & 0.01 & -0.01 \end{bmatrix}. \quad (4.4)$$

Second Learning Trial

Assume that the first pair is randomly selected. Therefore,

$$\mathbf{x}(0) = [1 \ 1 \ -1 \ -1]^T, \quad \mathbf{y}(0) = [1 \ -1 \ -1]^T. \quad (4.5)$$

Using (3.2) we compute $\mathbf{y}(1)$ and $\mathbf{x}(1)$ to get

$$\mathbf{x}(1) = [0.011 \ 0.011 \ 0.011 \ -0.011]^T, \quad \mathbf{y}(1) = [0.022 \ -0.022 \ 0.022]^T. \quad (4.6)$$

Using (3.4) and (3.5), the weight matrices are updated and their values are now

$$\mathbf{W}(2) = \begin{bmatrix} 0.0199 & 0.0199 & 0.0003 & -0.0199 \\ -0.0199 & -0.0199 & -0.0003 & 0.0199 \\ -0.0003 & -0.0003 & 0.02 & 0.0003 \end{bmatrix},$$

$$\mathbf{V}(2) = \begin{bmatrix} 0.02 & -0.02 & 0.0003 \\ 0.02 & -0.02 & 0.0003 \\ -0.0003 & 0.0003 & 0.0199 \\ -0.02 & 0.02 & -0.0003 \end{bmatrix}. \quad (4.7)$$

Notice that the weight matrices are not the transposed of each other like Kosko's [3].

k Learning Trial

If the process is iterated over and over, (e.g., 100 learning trials) the weight matrices converge to

$$\mathbf{W}(100) = \begin{bmatrix} 0.33 & 0.33 & 0 & -0.33 \\ -0.33 & -0.33 & 0 & 0.33 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{V}(100) = \begin{bmatrix} 0.5 & -0.5 & 0 \\ 0.5 & -0.5 & 0 \\ 0 & 0 & 1 \\ -0.5 & 0.5 & 0 \end{bmatrix}. \quad (4.8)$$

4.1.2. Recall

To test if the network can effectively recall a given pattern, a pattern is selected and it is iterated through the network until stabilization. For example, if the following pattern is presented to the network (noisy version of pattern 2) the different state-vectors will be:

$$\mathbf{x}(0) = [1 \ 1 \ 1 \ 1]^T. \quad (4.9)$$

The output of the second layer will give

$$\mathbf{y}(1) = [0.363 \ -0.363 \ 1]^T. \quad (4.10)$$

Then, this output is sent back to the first layer:

$$\mathbf{x}(1) = [0.344 \ 0.344 \ 1 \ -0.344]^T. \quad (4.11)$$

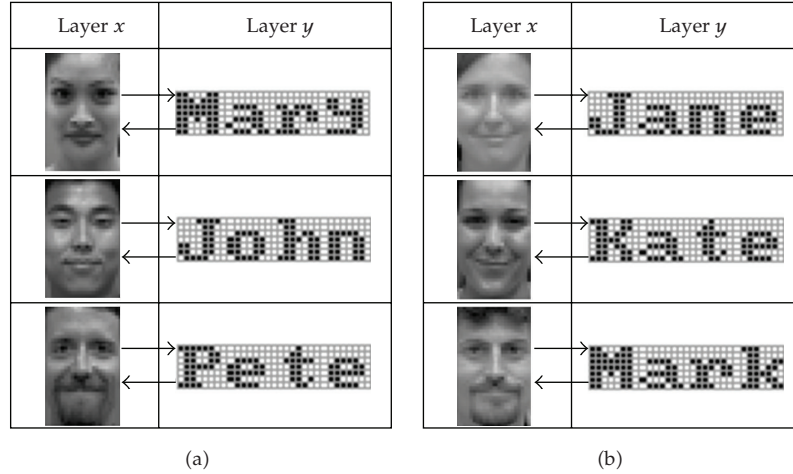


Figure 12: Patterns pairs used to trained the network.

The cycle will be repeated until convergence

$$\begin{aligned}
 \mathbf{y}(2) &= [0.344 \quad -0.344 \quad 1]^T, \\
 \mathbf{x}(2) &= [0.395 \quad 0.395 \quad 1 \quad -0.395]^T, \\
 &\vdots \\
 \mathbf{y}(50) &= [1 \quad -1 \quad 1]^T, \\
 \mathbf{x}(50) &= [1 \quad 1 \quad 1 \quad -1]^T.
 \end{aligned} \tag{4.12}$$

Therefore, the new input is classified as part of the second pattern pair. The next section presents a classic example of one-to-one association.









4.2. One-to-One Association

The task of the network consists of associating a given picture with a name. Therefore, the network should output from a picture the corresponding name, and from the name the corresponding picture.

4.2.1. Methodology

The first stimulus set represents 8-bit grey-level pictures. Each image has a dimension of 38×24 and therefore forms a 912-dimensional real-valued vector. They are reduced size versions of the California Facial Expressions set (CAFE, [41]). Each pixel was normalized to values between -1 and 1 . The second set consists of 4-letter words on a 7×31 grid that identify each picture. For each name a white pixel is assigned a value of -1 and a black pixel

Table 1: Network recall for a noisy input.

Original	Noisy version	Output ($k = 1$)		Output ($k = 2$)		Output ($k = 100$)	
		y	x	y	x	y	x
							

a value of +1. Each name forms a 217-dimensional bipolar vector. Therefore, the \mathbf{W} weight matrix has 217×912 connections and the \mathbf{V} weight matrix 912×217 connections. The network task was to associate each image with its corresponding name as depicted in Figure 12. The learning parameter (η) was set to 0.0025 and the output parameter (δ) to 0.1. Both values met the requirement for weight convergence and fixed-point development. Since the model's learning is online and iterative, the stimuli were not presented all at once. In order to save computational time, the number of iterations before each weight update was set to $t = 1$. The learning followed the same general procedure described in the previous simulation:

- (1) initialization of weights to zero,
- (2) random selection of a pair following a uniform distribution,
- (3) stimuli iteration through the network according to Equations (3.1) and (3.2) (one cycle),
- (4) weight update according to Equations (3.4) and (3.5),
- (5) repetition of 2 to 4 until the desired number of learning trials is reached ($k = 15\,000$).

4.2.2. Results

It took about 12 000 learning trial before the learning converged. The network was able to perfectly associate a name with the corresponding picture, and vice versa. Table 1 to Table 4 illustrates some examples of the noise tolerance and pattern completion properties of the network. More precisely, Table 1 shows how the network was able to recall the proper name under a noisy input. The input consisted of the first picture contaminated with a noise level of 22%. In other words, 200 pixels (out of 912) were randomly selected and their values were multiplied by -1 .

In addition, by the bidirectional nature of the network, it is also possible to not only recall the appropriate name but also to clean the noisy input. Table 2 shows an example of pattern completion. In this case the eye band consisted of pixels of zero value. The network was able to correctly output the name and restore the missing eyes. The network can also output a picture given an appropriate noisy name. For example, Table 3 shows that the network was able to recall the appropriate name even though the input name was incorrect (Kate instead of Katy). Finally, Table 4 shows that since "Pete" is the only name that begins with a "P", then only the first letter is necessary for the network to output the correct face. The general performance of the network has been evaluated by Chartier and Boukadoum [7]. The next section extends the simulations by introducing many-to-one associations.

Table 2: Network recall for an incomplete pattern.















Original	Noisy version	Output ($k = 1$)		Output ($k = 100$)	
		y	x	y	x
					

Table 3: Network recall for an incorrect name.

Original	Noisy version	Output ($k = 1$)		Output ($k = 2$)		Output ($k = 100$)	
		x	y	x	y	x	y
							

4.2.3. Transmission Function with Hard Limits

For some applications (e.g., [7]), It is desired that the transmission function lies within a fixed range of values. Saturating limits at -1 and 1 can be added to (3.2) and the transmission function is then expressed by the following equations:

$$\forall j, \dots, N, \quad \mathbf{y}_j(t+1) = f(\mathbf{a}_j(t)) = \begin{cases} 1 & \text{if } \mathbf{a}_j(t) > 1 \\ -1 & \text{if } \mathbf{a}_j(t) < -1, \\ (\delta + 1)\mathbf{a}_j(t) - \delta\mathbf{a}_j^3(t), & \text{else,} \end{cases} \quad (4.13)$$

$$\forall j, \dots, M, \quad \mathbf{x}_j(t+1) = f(\mathbf{b}_j(t)) = \begin{cases} 1 & \text{if } \mathbf{b}_j(t) > 1 \\ -1 & \text{if } \mathbf{b}_j(t) < -1, \\ (\delta + 1)\mathbf{b}_j(t) - \delta\mathbf{b}_j^3(t), & \text{else.} \end{cases}$$

In contrast to a sigmoid transmission function, this function is not asymptotic at the ± 1 values, and it still has the advantage of exhibiting continuous-valued (or gray-level) attractor behavior when used in a recurrent network. Figure 13 illustrates the shape of the transmission function when $\delta = 0.5$.

We compared the same network with and without hard limits on the same task (Figure 9) previously described in Section 2.5.1. The performance of the network was evaluated with transmission function (δ) values between 0.05 to 0.3; values outside this range gave worst results. In addition, the network was compared with the best results ($\delta = 0.05$) obtained from the simulation illustrated in Figure 10 when no hard limits was used. After the association of the desired pattern pairs, radius of attraction of the network was evaluated

Table 4: Network recall for missing name parts.

Original	Noisy version	Output ($k = 1$)		Output ($k = 10$)		Output ($k = 100$)	
		x	y	x	y	x	y

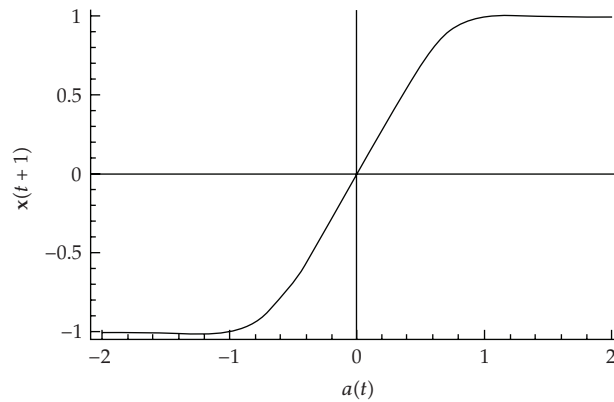


Figure 13: Transmission function when $\delta = 0.5$ with hard limits at -1 and $+1$.

under noisy input obtained by randomly flipping pixels in the input pattern. The number of pixel flips varied from 0 to 10.

Figure 14 illustrated the various performances. The results show that under low to medium noise, the network will have the best performance (about 10% increases) if no hard limits are used. However, under medium-to-high level of noise the situation is reverse; the network will have the best performance (about 5% increases) if hard limits are used ($\delta = 0.1$).

4.3. Many-to-One Association

This simulation illustrates many-to-one association. The idea is to associate different emotions depicted by different peoples to the proper name tag. Therefore, upon the presentation of a given images, the network should output the corresponding emotion. The simulation is based on Tabari et al. [42].

4.3.1. Methodology

As in the previous section, the first stimulus set represents 8-bits grey-level pictures. Each image has a dimension of 38×24 and therefore forms a 912-dimensional real value vector. They are reduced size version of the California Facial Expressions sample (CAFE, [41]). Each image pixels was rescaled to values between -1 and 1 . For each of the 9 individuals 7 images reflect a given emotion (anger, disgust, fear, happy, maudlin, neutral, and surprised). The

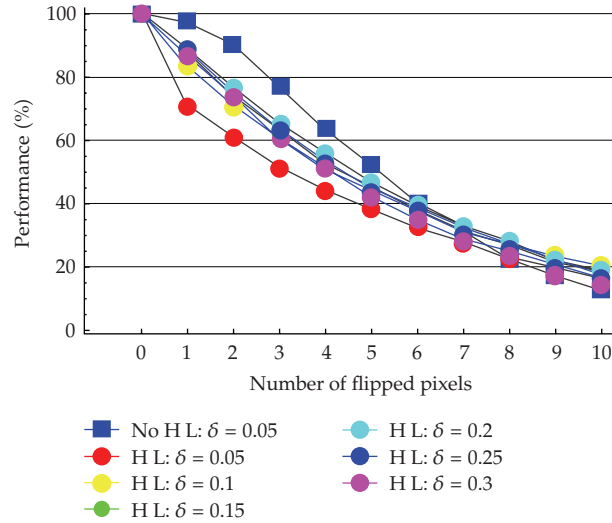


Figure 14: Comparison between the transmission function with (disks) or without hard limits (square) in function of various values of delta.

second set consists of letters placed on a 7×7 grid that identify each emotion (A, D, F, H, M, N, and S). For each letter a white pixel is assigned a value of -1 and a black pixel a value of $+1$. Each name forms a 49-dimensional bipolar vector. Therefore, the \mathbf{W} weight matrix has 49×912 connections and the \mathbf{V} weight matrix 912×49 connections. The network task was to associate each emotion, expressed by different people, with the corresponding letter (Figure 15). η was set to 0.0025 and δ to 0.1. Both values met the requirement for weight convergence and fixed-point behavior. The learning procedure followed the general procedure expressed previously.

4.3.2. Results

Once the learning trials were finished (about 1000 epochs), the network was able to correctly recall each emotion from the different people expressing it. As in one-to-one association, the network was able to remove the noise and correctly recall the appropriate emotion tag. Table 5 shows two examples of noisy images that were contaminated by 200-pixel flips (22%). Table 6 shows that the network recalled the appropriate emotion tag even in the absence of picture parts that are usually deemed important for identifying emotions, that is, the eyes and the mouth.

However, the most important property in this context is the fact that the two weight matrices are dynamically linked together. As was shown before, the network will output the corresponding letter given a face, but which face should the network output for a given letter? In this case, going from the letter layer to the picture layer implies one-to-many association. Without an architecture modification that can allow contextual encoding [7], the network will not be able to output the correct pictures. Rather, the network will average all the people's emotional expressions. Therefore, as shown in Figure 16, the network is able to extract what features in the images make each emotion different.

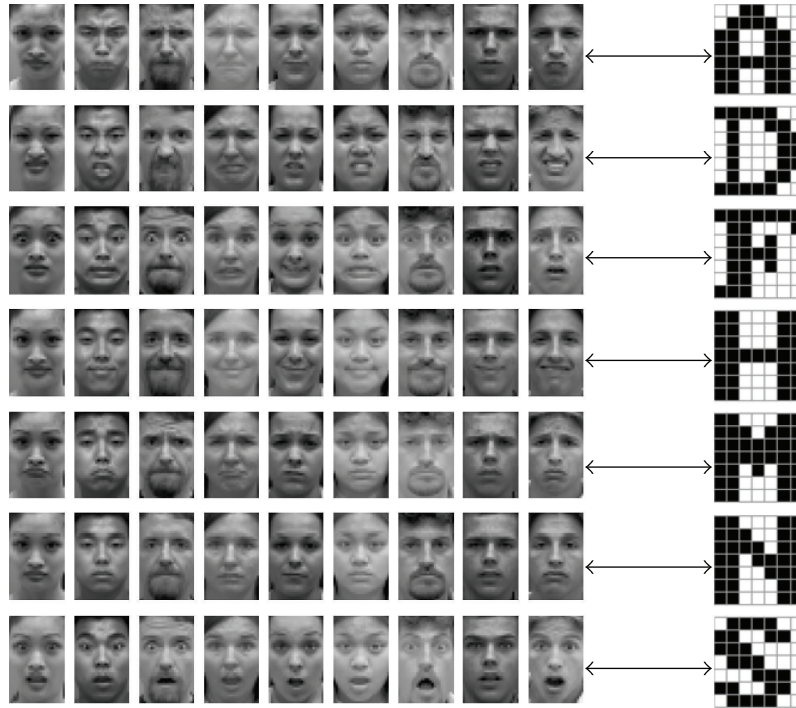








Figure 15: Patterns pairs used to trained the network.







Table 5: Network recall using a noisy input: 200 pixels flipped (22%).

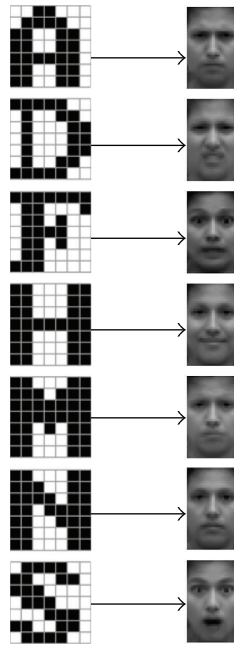
Step 0 (Input)	Step 0 (Noisy input)	Step 1 (Output)
		
		

5. Model Extension to Temporal Associative Memory

Until now, the association between the different stimuli is static. In some situations, the network can also perform multistep pattern recognition [43]. Such is the case when the output is a time-series. To allow such encoding, the network’s architecture is be modified.

Table 6: Network recall when important feature is removed (eyes and mouth).

Step 0 (Input)	Step 0 (Noisy input)	Step 1 (Output)
		
		

**Figure 16:** Pictures reconstructed using the emotion tags.

5.1. Architecture Modification

Figure 17 shows that, instead of having two heteroassociative networks connected together as in the Okajima et al. model [44], the network is composed of a heteroassociative and an autoassociative layer. The heteroassociative layer is used to map a given pattern to the next layer, whereas the autoassociative layer acts like a time delay circuit that feeds back the initial input to the heteroassociative layer. With this delay, it becomes possible for the overall network to learn temporal pattern sequences. Figure 17 illustrates the difference between a temporal associative memory and a bidirectional associative memory.

The initial value of the input pattern to the heteroassociative part is $\mathbf{y}(0)$, and its output, $\mathbf{y}(t)$, feeds the autoassociative part as $\mathbf{x}(0)$. The latter yields an output, $\mathbf{x}(t)$, that is

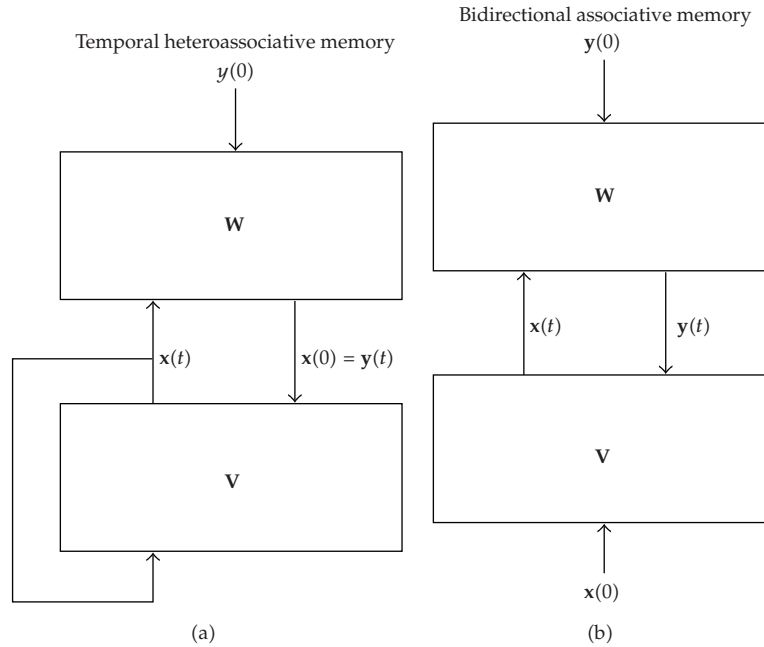


Figure 17: Comparison between the temporal associative memory and the standard BAM.

fed back into the heteroassociative part as well as into the autoassociative part. Thus, the autoassociative part serves as a context unit to the heteroassociative part. These two networks work in synergy to allow the necessary flow of information for, as our simulations will show, multistep pattern recall while still being able to filter noise out of the inputs.

5.2. Simplification of the Learning Function

In the case of autoassociative learning, the function expressed by (3.5) can be simplified by using the fact that the weight matrix is then square and symmetric. The symmetry property has the effect of canceling the cross terms in (3.5), and the autoassociative learning function becomes

$$\begin{aligned} \mathbf{V}(k+1) &= \mathbf{V}(k) + \eta \left(\mathbf{x}(0)\mathbf{x}(0)^T + \mathbf{x}(0)\mathbf{x}(t)^T - \mathbf{x}(t)\mathbf{x}(0)^T - \mathbf{x}(t)\mathbf{x}(t)^T \right), \\ \mathbf{V}(k+1) &= \mathbf{V}(k) + \eta \left(\mathbf{x}(0)\mathbf{x}(0)^T - \mathbf{x}(t)\mathbf{x}(t)^T \right), \end{aligned} \quad (5.1)$$

where \mathbf{V} represents the connections weight matrix. The learning rule is then a sum of a positive and a negative Hebbian term [7, 38].

5.3. Simulation 1: Limit Cycle Behavior

This simulation illustrates how the network can perform temporal association. The task consists of learning different planar rotation of the same object.

Table 7: Five binary sequences composed of 8 time steps.

Sequence	45°	90°	135°	180°	225°	270°	315°	0°
1								
2								
3								
4								
5								

Table 8: Network recall using a noisy input: 700 pixels flipped (28%).

Step 0 (Input)	Step 1 (Output)	Step 2 (Output)	Step 3 (Output)	Step 4 (Output)	Step 5 (Output)	Step 6 (Output)	Step 7 (Output)
	Step 8 (Output)	Step 9 (Output)	Step 10 (Output)	Step 11 (Output)	Step 12 (Output)	Step 13 (Output)	Step 14 (Output)

5.3.1. Methodology

Five different pattern sequences must be learned. Each sequence is composed of the same image with a 45 degrees planar rotation. Table 7 illustrates the five patterns sequences. The network has to associative each pattern of a sequence with the following one. In other words the network will associate the 45° with 90°, the 90° with 135°, the 135° with 180°, the 180° with the 225°, the 225° with 270°, the 270° with the 315°, the 315° with the 0°, and the 0° with the 45° image. Therefore, the steady-state of the network should be a limit cycle of period 8. Each binary stimulus was placed on a 50 × 50 grid, where white and black pixels were assigned -1 and 1 values, respectively. The free parameters were set to $\eta = 0.0002$ and $\delta = 0.1$, in accordance to the specification given in Chartier et al. [8].

5.3.2. Results

The network took about 2000 learning trials to converge and was able to correctly learn the 5 pattern sequences. It was also able to operate with noisy inputs and perform some generalization. For instance, Table 8 illustrates how the network was able to remove noise

Table 9: Network recall using a different picture: another example of butterfly.































Step 0 (Input)	Step 1 (Output)	Step 2 (Output)	Step 3 (Output)	Step 4 (Output)	Step 5 (Output)	Step 6 (Output)	Step 9 (Output)
							
	Step 8 (Output)	Step 9 (Output)	Step 10 (Output)	Step 11 (Output)	Step 12 (Output)	Step 13 (Output)	Step 14 (Output)
							

Table 10: Network recall using a different planar rotation: 275°.

Step 0 (Input)	Step 1 (Output)	Step 2 (Output)	Step 3 (Output)	Step 4 (Output)	Step 5 (Output)	Step 6 (Output)	Step 7 (Output)
							
	Step 8 (Output)	Step 9 (Output)	Step 10 (Output)	Step 11 (Output)	Step 12 (Output)	Step 13 (Output)	Step 14 (Output)
							

from the “fish” image as the temporal sequence was recalled. Table 9 shows that the network can generalize its learning to use similar objects. In this case a new “butterfly” picture was used as the initial input. As the stimulus iterates through the network, the network recalled the original “butterfly” depicted in Table 7 (the butterfly output at step 8 is different from the original one at step 0). Finally, Table 10 shows that the network can correctly output the image sequence under small planar rotation variations of the initial input image. In this particular example the new “dinosaur” picture represents a 275° rotation (270° + 5°). Although that particular rotation was not part of initial sequence set, the network was able to correctly recall the appropriate picture sequence.

5.4. Simulation 2: Fixed-Point Behavior

Again, this simulation illustrates how the network can perform temporal association. The task is to learn different planar rotations of the same object. Contrarily to the previous simulation, once the network has been through every planar rotation it converges to a fixed-point.

Table 11: Five binary sequences composed of 8 time steps.

Sequence	45°	→ 90°	→ 135°	→ 180°	→ 225°	→ 270°	→ 315°	→ 0° ↻
1								
2								
3								
4								
5								

Table 12: Network recall using a noisy input: 700 pixels flipped (28%).

Step 0 (Input)	Step 1 (Output)	Step 2 (Output)	Step 3 (Output)	Step 4 (Output)	Step 5 (Output)	Step 6 (Output)	Step 7 (Output)

5.4.1. Methodology

Five different pattern sequences must be learned. Each sequence is composed of the same image with a 45° planar rotation. Table 11 illustrates the five patterns sequences. The network must associative each pattern of a sequence with the following one. In other words the network will associate the 45° with 90°, the 90° with 135°, the 135° with 180°, the 180° with the 225°, the 225° with 270°, the 270° with the 315°, the 315° with the 0° and *the 0° with the 0°* Image. This last association will creates a fixed-point attractor that can then be used for other types of association in more complex architectures as will be shown in the next section. Each binary stimulus was placed on a 50 × 50 grid, where white and black pixels were assigned a -1 and 1 value, respectively. The free parameters were set to $\eta = 0.0002$ and $\delta = 0.1$, in accordance to the specification given in [8].

5.4.2. Results

The network took about 2000 learning trials before convergence occurred, and it was able to correctly learn the 5 pattern sequences. The same examples for noise tolerance and generalization were used to compare the network performance between the limit cycle and the fixed-point condition. Table 12 shows that the network can eliminate noise as the sequence is recalled. Table 13 shows learning generalization to other similar objects. After 9 time steps, the network recalled the “butterfly” that is depicted in Table 11. Finally, Table 14 shows that the network can also output the correct image sequence under small initial variations of

Table 13: Network recall using a different picture: another example of butterfly.

















Step 0 (Input)	Step 1 (Output)	Step 2 (Output)	Step 3 (Output)	Step 4 (Output)	Step 5 (Output)	Step 6 (Output)	Step 9 (Output)
							

Table 14: Network recall using a different planar rotation: 275°.

Step 0 (Input)	Step 1 (Output)	Step 2 (Output)	Step 3 (Output)	Step 4 (Output)	Step 5 (Output)	Step 6 (Output)	Step 7 (Output)
							

planar rotation. In this particular example the “dinosaur” picture represent a 275° rotation as in the previous section.

6. Multidirectional Associative Memories

Sometimes association must be generalized to more than two directions. To deal with multiple associations, Hagiwara [45] proposed a Multidirectional Associative Memory [MAM]. The architecture was later extended to deal with temporal sequences [46]. This section shows some properties of a MAM that is composed of one temporal heteroassociative memory and one bidirectional associative memory. As in the previous simulations, only the network topology is modified as Figure 18 shows; the learning and transmission function remain the same. In this simulation the information received by the y -layer at time t consist of $x(t)$ and $z(t)$. The feedback sent by $z(t)$ is useful only once the network reaches steady-state, the last pattern of the stimuli set. Therefore, to minimize its effect during the temporal recall, the feedback value of $z(t)$ is lower than $x(t)$. This is formally expressed by





































$$y(t) = f(W1x(t) + \alpha W2z(t)), \quad (6.1)$$

where $0 < \alpha \ll 1$, $W1$ and $W2$ represent the weight connections linking $x(t)$ and $z(t)$, respectively. Taken together, $W1$ and $W2$ form the W weight matrix.

6.1. Methodology

The simulation performs the multistep pattern recognition described in the previous section in combination with a standard one-to-one association. The one-to-one association is performed only when the network is at a given fixed-point. In other words the association will occur only when the output is at a 0° planar rotation for the recalled sequence (Table 11). The association is made between the given picture and the first letter of its name as illustrated in Figure 19. The learning and transmission function parameters were set to $\eta = 0.0001$ and $\delta = 0.1$. The z -layer feedback parameter was set to $\alpha = 0.4$.

Table 15: Multistep recall of the "runner" binary picture (45° planar rotation).

Step	Input	Out 1	Out 2	Out 1 + α Out 2
1				
2				
3				
4				
5				
6				
7				
8				
9				

6.2. Results

Of course, if the temporal or bidirectional associative sections are independently activated, the resulting network performance will be the same as the previous sections. What matter in this case is the output state of the network $y(t)$. The best case is when a pattern is presented to the network that is closed to its fixed-point behavior (315°). The effect of the feedback coming from z-layer will be limited. On the opposite, the worst case is when a pattern is presented with a 45° planar rotation. In this case, it will need 8 time steps before the output vector is at its fixed-point behavior. From this point on, the picture identification process can occur. If the feedback coming from the z-layer is too strong, it could deviate the output vector's

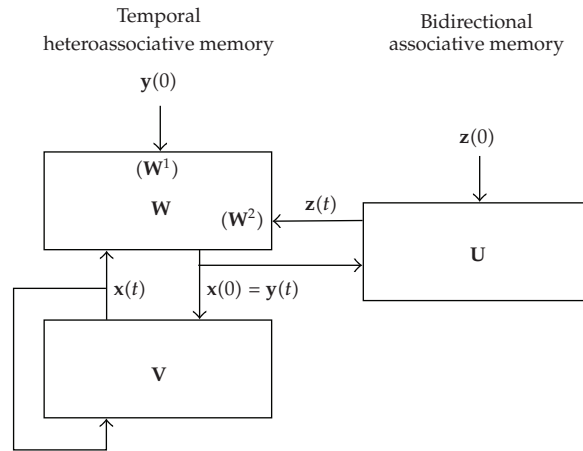


Figure 18: Architecture of the multidirectional associative memory. The network is composed of a temporal heteroassociative memory and a bidirectional associative memory.

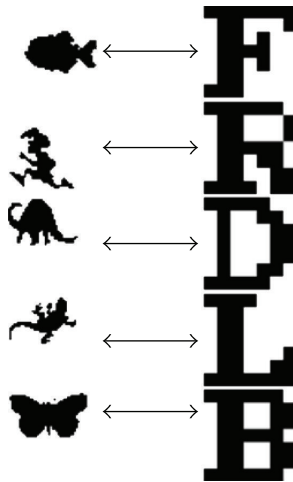


Figure 19: Patterns pairs used to trained the network.

trajectory. This could make the network converge to the wrong fixed-point and lead to an incorrect name tag. Because of this, the feedback value of the z-layer must be low.

Table 15 shows an example of the “runner” given an initial 45° planar rotation. After one time step the network correctly output the 90° picture (Out 1). Of course, the name tag is incorrect (Out 2). Now the new input will correspond to the output of the x layer (Out 10) and the feedback given by the z-layer (Out 2 not shown). The overall effect will be $Out\ 1 + \alpha\ Out\ 2$ and will be used as the new input. The process is repeated until convergence. After 9 time steps, Table 15 shows that the correct 0° “runner” picture is output as well as the correct name tag letter “R”.

7. Conclusion

Several interesting properties of a BAM architecture have been shown. First, it was shown that a simple time-delay Hebbian learning can perform one-to-one association and many-to-one association with both binary and real-valued pattern. In addition, the BAM can be modified in such a way that both heteroassociation and autoassociation can be accomplished within the same architecture. In this case, the autoassociative part act like a time delay and the overall network can be used for temporal association. A simple case of one-step delay was shown. However, by adding l extra layers of autoassociation, the network can be easily modified to handle l -step delays. Finally, if a combination of temporal and bidirectional associations is grouped into a multidirectional associative memory more complex behaviors can be obtained.

In all cases, the same learning and transmission function are used. The only modification concerned the network topology. This property gives the network a high internal consistency. More complex architectures are possible, but a really powerful implementation improvement would be to develop an algorithm that guides the architecture growth based on the problem to be solved. Therefore, the architecture could be modified in function of the task to be performed and the desired behaviors, using BAMs as building blocks.

Acknowledgment

This research was supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

- [1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, 1982.
- [2] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, "Distinctive features, categorical perception, and probability learning: some applications of a neural model," *Psychological Review*, vol. 84, no. 5, pp. 413–451, 1977.
- [3] B. Kosko, "Bidirectional associative memories," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 49–60, 1988.
- [4] T. Kohonen, "Correlation matrix memories," *IEEE Transactions on Computers*, vol. 21, no. 4, pp. 353–359, 1972.
- [5] M. E. Acevedo-Mosqueda, C. Yáñez-Márquez, and I. López-Yáñez, "Alpha-Beta bidirectional associative memories: theory and applications," *Neural Processing Letters*, vol. 26, no. 1, pp. 1–40, 2007.
- [6] S. Arik, "Global asymptotic stability analysis of bidirectional associative memory neural networks with time delays," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 580–586, 2005.
- [7] S. Chartier and M. Boukadoum, "A bidirectional heteroassociative memory for binary and grey-level patterns," *IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 385–396, 2006.
- [8] S. Chartier, P. Renaud, and M. Boukadoum, "A nonlinear dynamic artificial neural network model of memory," *New Ideas in Psychology*, vol. 26, no. 2, pp. 252–277, 2008.
- [9] S. Du, Z. Chen, Z. Yuan, and X. Zhang, "Sensitivity to noise in bidirectional associative memory (BAM)," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 887–898, 2005.
- [10] T. D. Eom, C. Choi, and J. J. Lee, "Generalized asymmetrical bidirectional associative memory for multiple association," *Applied Mathematics and Computation*, vol. 127, no. 2-3, pp. 221–233, 2002.
- [11] H. Oh and S. C. Kothari, "Adaptation of the relaxation method for learning in bidirectional associative memory," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 576–583, 1994.

- [12] C.-S. Leung, "Optimum learning for bidirectional associative memory in the sense of capacity," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 5, pp. 791–796, 1994.
- [13] D. Shen and J. B. Cruz Jr., "Encoding strategy for maximum noise tolerance bidirectional associative memory," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 293–300, 2005.
- [14] H. Shi, Y. Zhao, and X. Zhuang, "A general model for bidirectional associative memories," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 28, no. 4, pp. 511–519, 1998.
- [15] Y. F. Wang, J. B. Cruz Jr., and J. H. Mulligan Jr., "Two coding strategies for bidirectional associative memory," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 81–92, 1990.
- [16] T. Wang, X. Zhuang, and X. Xing, "Weighted learning of bidirectional associative memories by global minimization," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 1010–1018, 1992.
- [17] L. Wang and X. Zou, "Capacity of stable periodic solutions in discrete-time bidirectional associative memory neural networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 51, no. 6, pp. 315–319, 2004.
- [18] Y. Wu and D. A. Pados, "A feedforward bidirectional associative memory," *IEEE Transactions on Neural Networks*, vol. 11, no. 4, pp. 859–866, 2000.
- [19] Z. B. Xu, Y. Leung, and X. W. He, "Asymmetric bidirectional associative memories," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 10, pp. 1558–1564, 1994.
- [20] X. Zhuang, Y. Huang, and S. Chen, "Better learning for bidirectional associative memory," *Neural Networks*, vol. 6, no. 8, pp. 1131–1146, 1993.
- [21] G. Costantini, D. Casali, and R. Perfetti, "Neural associative memory storing gray-coded gray-scale images," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 703–707, 2003.
- [22] C.-C. Wang, S.-M. Hwang, and J.-P. Lee, "Capacity analysis of the asymptotically stable multi-valued exponential bidirectional associative memory," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 26, no. 5, pp. 733–743, 1996.
- [23] D. Zhang and S. Chen, "A novel multi-valued BAM model with improved error-correcting capability," *Journal of Electronics*, vol. 20, no. 3, pp. 220–223, 2003.
- [24] J. M. Zurada, I. Cloete, and E. van der Poel, "Generalized Hopfield networks for associative memories with multi-valued stable states," *Neurocomputing*, vol. 13, no. 2–4, pp. 135–149, 1996.
- [25] L. Wang, "Multi-associative neural networks and their applications to learning and retrieving complex spatio-temporal sequences," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 29, no. 1, pp. 73–82, 1999.
- [26] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.
- [27] H. Wakuya and J. M. Zurada, "Bi-directional computing architecture for time series prediction," *Neural Networks*, vol. 14, no. 9, pp. 1307–1321, 2001.
- [28] G. Bradski, G. A. Carpenter, and S. Grossberg, "STORE working memory networks for storage and recall of arbitrary temporal sequences," *Biological Cybernetics*, vol. 71, no. 6, pp. 469–480, 1994.
- [29] B. B. Nasution and A. I. Khan, "A hierarchical graph neuron scheme for real-time pattern recognition," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 212–229, 2008.
- [30] J. A. Starzyk and H. He, "Anticipation-based temporal sequences learning in hierarchical structure," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 344–358, 2007.
- [31] D.-L. Lee, "A discrete sequential bidirectional associative memory for multi step pattern recognition," *Pattern Recognition*, vol. 19, pp. 1087–1102, 1988.
- [32] L. Wang, "Heteroassociations of spatio-temporal sequences with the bidirectional associative memory," *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1503–1505, 2000.
- [33] R. W. Zhou and C. Quek, "DCBAM: a discrete chainable bidirectional associative memory," *Pattern Recognition Letters*, vol. 17, no. 9, pp. 985–999, 1996.
- [34] A. A. Koronovskii, D. I. Trubetskoy, and A. E. Khramov, "Population dynamics as a process obeying the nonlinear diffusion equation," *Doklady Earth Sciences*, vol. 372, pp. 755–758, 2000.
- [35] D. Kaplan and L. Glass, *Understanding Nonlinear Dynamics*, Springer, New York, NY, USA, 1995.
- [36] L. Personnaz, I. Guyon, and G. Dreyfus, "Information storage and retrieval in spin-glass like neural networks," *Journal of Physics Letters*, vol. 46, pp. L359–L365, 1985.
- [37] M. H. Hassoun, "Dynamic heteroassociative neural memories," *Neural Networks*, vol. 2, no. 4, pp. 275–287, 1989.
- [38] S. Chartier and R. Proulx, "NDRAM: nonlinear dynamic recurrent associative memory for learning bipolar and nonbipolar correlated patterns," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1393–1400, 2005.
- [39] B. Kosko, "Unsupervised learning in noise," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 44–57, 1990.

- [40] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.
- [41] M. N. Dailey, G. W. Cottrel, and J. Reilly, HYPERLINK, 2001, <http://www.cs.ucsd.edu/users/gary/CAFE/>.
- [42] K. Tabari, M. Boukadoum, S. Chartier, and H. Lounis, "Reconnaissance d'expressions faciales à l'aide d'une mémoire associative bidirectionnelle à fonction de sortie chaotique," in *Proceedings of Maghrebian Conference on Software Engineering and Artificial Intelligence*, pp. 422–426, Agadir, Morocco, December 2006.
- [43] S. Chartier and M. Boukadoum, "A sequential dynamic heteroassociative memory for multistep pattern recognition and one-to-many association," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 59–68, 2006.
- [44] K. Okajima, S. Tanaka, and S. Fujiwara, "A heteroassociative memory network with feedback connection," in *Proceedings of the 1st International Joint Conference on Neural Networks*, pp. H.711–H.718, San Diego, Calif, USA, 1987.
- [45] M. Hagiwara, "Multidirectional associative memory," in *Proceeding of the International Joint Conference on Neural Networks*, pp. 3–6, Washington, DC, USA, 1990.
- [46] A. F. R. Araujo and M. Vieira, "Temporal multidirectional associative memory: adaptable, continuous, and self-connected MAM," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 2, pp. 1194–1198, Houston, Tex, USA, June 1997.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

