Global Journal of Researches in Engineering: F
Electrical and Electronics Engineering
Volume 21 Issue 2 Version 1.0 Year 2021
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 2249-4596 & Print ISSN: 0975-5861

Implementation of an Associative Memory using a Restricted Hopfield Network

By Tet Yeap
University of Ottawa

Abstract- A trainable analog restricted Hopfield Network is presented in this paper. It consists of two layers of nodes, visible and hidden nodes, connected by weighted directional paths forming a bipartite graph with no intralayer connection. An energy or Lyapunov function was derived to show that the proposed network will converge to stable states. The proposed network can be trained using either the modified SPSA or BPTT algorithms to ensure that all the weights are symmetric. Simulation results show that the presence of hidden nodes increases the network's memory capacity. Using EXOR as an example, the network can be trained to be a dynamic classifier. Using A, U, T, S as training characters, the network was trained to be an associative memory. Simulation results show that the network can perform perfect re-creation of noisy images. Its recreation performance has higher noise tolerance than the standard Hopfield Network and the Restricted Boltzmann Machine. Simulation results also illustrate the importance of feedback iteration in implementing associative memory to re-create from noisy images.

Keywords: hopfield network, restricted boltzmann machine, energy function, lyapunov function, restricted hopfield network, trainable networks, hidden nodes, basin of attraction, attractors.

GJRE-F Classification: FOR Code: 290903

Strictly as per the compliance and regulations of:

Implementation of an Associative Memory using a Restricted Hopfield Network

Tet Yeap

Abstract- A trainable analog restricted Hopfield Network is presented in this paper. It consists of two layers of nodes, visible and hidden nodes, connected by weighted directional paths forming a bipartite graph with no intralayer connection. An energy or Lyapunov function was derived to show that the proposed network will converge to stable states. The proposed network can be trained using either the modified SPSA or BPTT algorithms to ensure that all the weights are symmetric. Simulation results show that the presence of hidden nodes increases the network's memory capacity. Using EXOR as an example, the network can be trained to be a dynamic classifier. Using A, U, T, S as training characters, the network was trained to be an associative memory. Simulation results show that the network can perform perfect re-creation of noisy images. Its recreation performance has higher noise tolerance than the standard Hopfield Network and the Restricted Boltzmann Machine. Simulation results also illustrate the importance of feedback iteration in implementing associative memory to re-create from noisy images.

Keywords: hopfield network, restricted boltzmann machine, energy function, lyapunov function, restricted hopfield network, trainable networks, hidden nodes, basin of attraction, attractors.

I. Introduction

In 1982, based on his studies of collective dynamical computation in neural networks, Hopfield [1, 2, 3] proposed an influential recurrent neural network with many potential applications such as content addressable memory and optimization engine for the traveling-salesman problem. He formulated an Energy function for the network using the Lyapunov Direct Method showing that the network converges to a stable state if it has symmetric weights. Each network node does not have self-feedback.

Hopfield network comes in two forms: analog or discrete. However, in either format, the network can only be programmed to memorize patterns using the Hebbian Rule and has a limited memory capacity to store 0.15N patterns where N is the network's number of nodes. Many have tried to improve the network's memory capacity problem and trainability issue [4, 5]. For example, instead of trying to memorize the patterns in one presentation cycle, Gardner [6, 7, 8] improved the network by presenting the training patterns repeatedly and using the perceptron convergence procedure to train each node to generate the correct state given the states of all the other nodes for a particular training vector.

A Boltzmann machine is a stochastic recurrent neural network with interconnected visible and hidden nodes introduced by Hinton [9, 10]. Like a Hopfield network, a Boltzmann has a similar energy function when the weights are symmetric and converges to a stable state when an input vector is presented to the visible nodes. A Boltzmann machine takes a long time to train. As a result, a restricted Boltzmann machine (RBM) was introduced [11, 12, 13]. It consists of two layers of nodes, L visible and M hidden nodes, connected by symmetric weights with no intralayer connection. Each node makes probabilistic decisions to be either on or off. The connection restriction allows for more efficient training algorithms, notably the gradient-based contrastive divergence algorithm, to be developed. The network is capable of learning the probabilistic pattern of a set of inputs.

An analog restricted Hopfield network (RHN) is proposed in this paper to solve the memory capacity and the trainability issue of the Hopfield network. Like an RBM, the architecture consists of two layers of nodes, visible and hidden nodes, connected by directional weighted connection paths. The network is a fully-connected bipartite graph and has no intralayer connection. The visible nodes are classified into either input or output nodes to manage the flow of information to/from the visible nodes. An energy or Lyapunov function was derived to prove that the proposed network always converges to stable states when an input vector is presented. The proposed network iterates, sending signals back and forth between the two layers until all its nodes reach an equilibrium state based on the corresponding basin of attraction, generating the desired output vector.

Two training algorithms were used to train the proposed network: Simultaneous Perturbation Stochastic Approximation (SPSA) [14, 15] and Back propagation Through Time (BPTT) [16, 17]. The SPSA algorithm was introduced by Spall and is simple to implement. It can estimate the gradient of the error function using only two final error values of the function. Therefore, the merit of this training rule was demonstrated to train the proposed network. The BPTT algorithm, on the other hand, is based on the fact that the temporal operation of an RHN may be unfolded into

Author: School of Electrical Engineering and Computer Science University of Ottawa. e-mail: tyeap@uottawa.ca

a multilayer perceptron so that a standard back propagation algorithm could be applied.

Simulation results show that the proposed network can be trained to implement a dynamic classifier implementing an EXOR function. Using A, U, T, S as training characters, the network was trained to be an associative memory. Simulation results show that the network performs perfect re-creation of these images even when the input image is noisy. The results also show that the proposed network performs better than the standard Hopfield Network and RBM.

The paper is organized as follows. Section 2 presents the background work on Hopfield Network and Restricted Boltzmann Machine. An RHN with hidden nodes function is presented in section 3. In section 4, two algorithms to train the network are introduced. Section 5 presents some simulation results on the performance of the RHN.

II. Background

a) Hopfield Network

An analog Hopfield network consists of fully interconnected nodes modeled as amplifiers, in conjunction with feedback circuits comprises of wires, resistors, and capacitors, as shown in Figure 1.

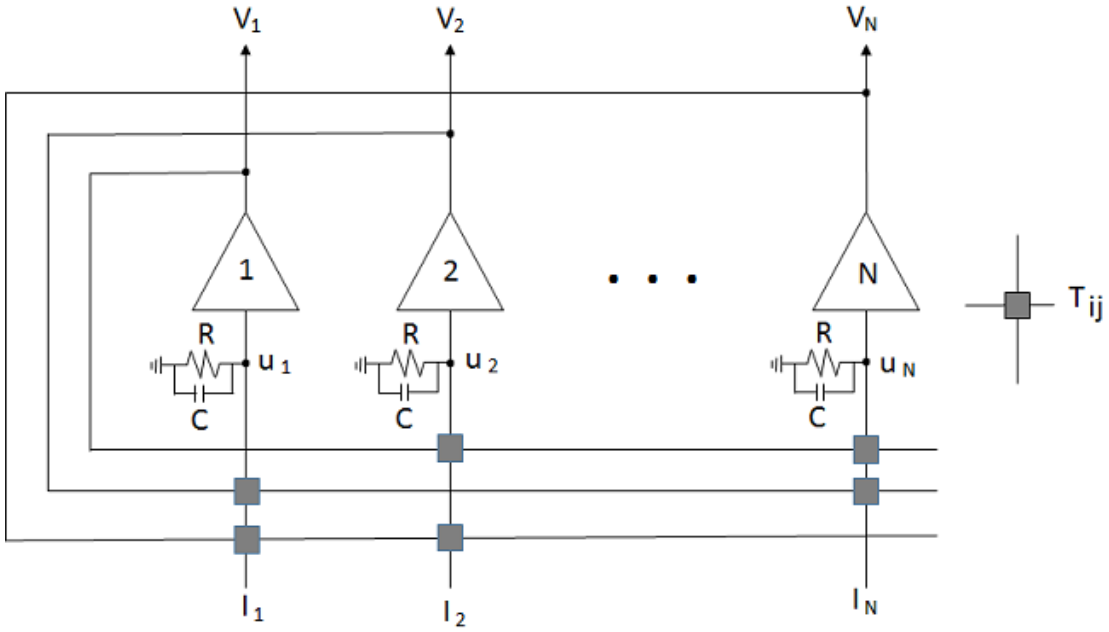

Figure 1: An analog Hopfield network

The dynamics of the network can be described by the following differential equations:

$$\frac{du_i}{dt} = \sum_{j=1}^{N} T_{ij} v_j - \frac{u_i}{\tau} + I_i \qquad (1)$$

$$\tau = RC$$

$$V_i = g(u_i)$$

Where N is the number of nodes in the network, u_i is the input voltage of the amplifier, T_{ij} is the weight or conductance connecting the output of node j to the input of node i, V_j is the output of node j, RC is the time constant of the network, I_i is the input to node i, and $g()$ is the output function of a node.

The following energy function for the network was derived by Hopfield using the Lyapunov Direct Method if the network has symmetric weights and each network node does not have self-feedback. For the initial-value problem, I_i input is applied to node i at $t = 0$ and then allow the network to evolve. The integration of the above differential equations provides the network states' evolution. With the energy function's existence, the network will always converge to a stable state.

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} V_i V_j - \sum_{i=1}^{N} V_i \frac{u_i}{\tau} - \sum_{i=1}^{N} V_i I_i \qquad (2)$$

Hopfield networks can only be programmed to memorize patterns using the Hebbian Rule. When the output function $g()$ is a sigmoid function, the network transforms the initial input vector it- eratively and continuously into the output vector in the range [0, 1]. To program the network to memorize specific binary input vectors ($S(p)$, $p = 1 \ldots P$), the weight or conductance T_{ij} is deter- mined by the following formula:

$$T_{ij} = \sum_{p=1}^{P} (2S_i(p) - 1)(2S_j(p) - 1) \qquad (3)$$

When the output function $g()$ is a hyperbolic tangent function, the network transforms the initial input vector

iteratively and continuously into the output vector in the range [1, 1]. To program the network to memorize specific binary input vectors ($S(p)$, $p = 1 \ldots P$), the weight or conductance T_{ij} is determined by the following formula:

$$T_{ij} = \sum_{p=1}^{P} S_i(p) S_j(p) \tag{4}$$

b) Restricted Boltzmann Machine (RBM)

A restricted Boltzmann machine (RBM) is a stochastic recurrent neural network [4, 9, 10] consisting of two layers of nodes, L visible and M hidden nodes, connected by symmetric weights with no intralayer connection. Each node makes probabilistic decisions to be either on or off. The network is capable of learning the probabilistic pattern of a set of inputs.

The following differential equations can describe the dynamics of the network:
In the forward path:

$$sum_i^H = \sum_{j=1}^{L} w_{ij}^H V_j^V + \theta_i^H \tag{5}$$

$$V_i^H = g(sum_i^H)$$

Where sum_i^H is the sum of all inputs to the hidden node i, w_{ij}^H is the weight connecting the output of visible node j to the input of hidden node i, V_j^v is the output of visible node j, θ_i^H is the threshold of hidden node i, V_i^H is the output of hidden node i, $g()$ is the Sigmoid logistic output function of hidden node i.
In the backward path:

$$sum_j^V = \sum_{i=1}^{M} w_{ji}^V V_i^H + \theta_j^V \tag{6}$$

$$V_j^V = g(sum_j^V)$$

Where sum_j^v is the sum of all inputs to the visible node i, $w_j\, i^v$ the weight connecting the output of hidden node i to the input of visible node j, V_i^H is the output of hidden node i, θ_j^V is the threshold of visible node j, V_j^V is the output of visible node j, $g()$ is the Sigmoid logistic output function of hidden node j.

For symmetric weights configuration, the energy or Lyapunov function of an RBM is given by:

$$E = -\sum_{i=1}^{M}\sum_{j=1}^{L} w_{ij}^H V_i^H V_j^V - \sum_{j=1}^{L} \theta_j^V V_j^V - \sum_{i=1}^{M} \theta_i^H V_i^H \tag{7}$$

With the existence of energy function, the network will always converge to a stable state when an input vector is presented to the visible nodes.

During the training of the network, an input vector p is presented. Let e_{ij}^+ (p) $= V_i^H V_j^v$ denotes the correlation of hidden node i and visible node j in the forward direction, and e_{ij}^- (p) $= V_i^H X_j^V$ denotes the correlation in the backward direction, where X_j^V is the value of visible unit j of pattern p that is estimated by the network. The weight is updated during a training process by:

$$w_{ij}(k+1) = w_{ij}(k) + \beta(e_{ij}^+(p) - e_{ij}^-(p)) \tag{8}$$

Where β is a small positive number.

III. Proposed Restricted Hopfield Network(RHN)

An analog restricted Hopfield network (RHN) is proposed to solve the memory capacity and train- ability issues of the Hopfield network. Like an RBM, the architecture consists of two layers of nodes, L visible and M hidden nodes, connected by directional weighted connection paths, as shown in Figure 2. The network is a connected bipartite graph and has no intralayer connection.

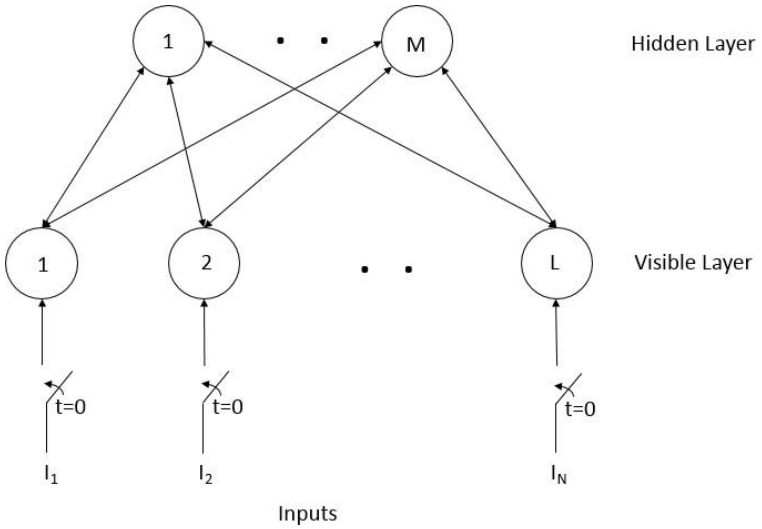

Figure 2: A Restricted Hopfield Network

The following differential equations describe the dynamics of the network: In the forward path:

$$\frac{du_i^H}{dt} = \sum_{j=1}^{L} w_{ij}^H V_j^V + \theta_i^H \qquad (9)$$

$$V_i^H = g(u_i)$$

Initial conditions:

$$V_j^V(0) = I_j$$

$$V_i^H(0) = 0$$

Where u_i^H is the sum of all inputs to the hidden nodes, W_{ij}^H is the weight connecting the output of visible node *j* to the input of hidden node *i*, V_j^V is the output of visible node *j*, θ_i^H is the threshold of hidden node i, V_i^H is the output of hidden node *i*, *g*() is the output function of hidden node *I*, and I_j is the initial input presented to the visible node *j*.

In the backward path:

$$\frac{du_j^V}{dt} = \sum_{i=1}^{M} w_{ji}^V V_i^H + \theta_j^V \qquad (10)$$

$$V_j^V = g(u_j^V)$$

Where u_j^V is the sum of all inputs to the visible nodes, w_{ji}^V is the weight connecting the output of hidden node *i* to the input of visible node *j*, V_i^H is the output of hidden node *i*, θ_j^V is the threshold of output node *j*, V_j^V is the output of visible node *j*, and *g*() is the output function of visible node *j*.

The network input can be either digital or analog, taking a value between 0 and 1. Using a Sigmoid function as output function *g*() for all the nodes, the output of all the nodes takes the value between 0 and 1. The proposed network will also work when the hyperbolic tangent function is used as the output function. In such a case, the output of all the nodes takes the value between 1 and 1. The proposed RHN always generates analog outputs between 0 and 1 using the Sigmoid output function or between−1 and 1 using the hyperbolic tangent output function.

Based on the Lyapunov Direct Method, it can be shown that Eqn. 11 is the energy or a Lyapunov function of the proposed network.

$$E = -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{L} w_{ij}^H V_i^H V_j^H - \frac{1}{2}\sum_{j=1}^{L}\sum_{i=1}^{M} w_{ji}^V V_j^H V_i^V - \sum_{i=1}^{M} V_i^H \theta_i^H - \sum_{i=1}^{L} V_i^V \theta_i^V \qquad (11)$$

Differentiating *E*, we get:

$$\frac{dE}{dt} = -\frac{d}{dt}\left(\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{L} w_{ij}^H V_i^H V_j^H\right) - \frac{d}{dt}\left(\frac{1}{2}\sum_{j=1}^{L}\sum_{i=1}^{M} w_{ji}^V V_j^H V_i^V\right) - \frac{d}{dt}\left(\sum_{i=1}^{M} V_i^H \theta_i^H\right) - \frac{d}{dt}\left(\sum_{i=1}^{L} V_i^V \theta_i^V\right) \qquad (12)$$

Expanding all the above terms in Eqn. 12, we get:

$$\frac{d}{dt}\left(\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{L} w_{ij}^H V_i^H V_j^H\right) = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{L} w_{ij}^H \frac{dV_i^H}{dt} V_j^V + \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{L} w_{ij}^H V_i^H \frac{dV_j^V}{dt}$$

$$\frac{d}{dt}(\frac{1}{2}\sum_{j=1}L\sum_{i=1}^{M}w_{ji}^{V}V_{j}^{H}V_{i}^{V})=\frac{1}{2}\sum_{j=1}^{L}\sum_{i=1}^{M}w_{ji}^{V}\frac{dV_{j}^{V}}{dt}V_{i}^{H}+\frac{1}{2}\sum_{j=1}^{L}\sum_{i=1}^{M}w_{ji}^{V}V_{j}^{V}\frac{dV_{i}^{H}}{dt} \tag{13}$$

$$\frac{d}{dt}(\sum_{i=1}^{M}V_{i}^{H}\theta_{i}^{H})=\sum_{i=1}^{M}\frac{dV_{i}^{H}}{dt}\theta_{i}^{H}$$

$$\frac{d}{dt}(\sum_{i=1}^{L}V_{i}^{V}\theta_{i}^{V})=\sum_{i=1}^{L}\frac{dV_{i}^{V}}{dt}\theta_{i}^{V}$$

Now consider the forward path:
Because the outputs of visible nodes are constant in the forward path, then:

$$\frac{dV_{i}^{V}}{dt}=0 \tag{14}$$

Assume that all the weights are symmetric, then:

$$w_{ij}^{H}=w_{ji}^{V} \tag{15}$$

Therefore, Equation 12 can be reduced to the following:

$$\frac{dE}{dt}=-\sum_{i=1}^{M}\sum_{j=1}^{L}w_{ij}^{H}\frac{dV_{i}^{H}}{dt}V_{j}^{V}-\sum_{i=1}^{M}\frac{dV_{i}^{H}}{dt}\theta_{i}^{H} \tag{16}$$

and can be simplified to:

$$\frac{dE}{dt}=-\sum_{i=1}^{M}\frac{dV_{i}^{H}}{dt}(\sum_{j=1}^{L}w_{ij}^{H}V_{j}^{V}+\theta_{i}^{H}) \tag{17}$$

Further simplified to:

$$\frac{dE}{dt}=-\sum_{i=1}^{M}\frac{dV_{i}^{H}}{dt}\frac{du_{i}^{H}}{dt} \tag{18}$$

Using Chain Rule:

$$\frac{dE}{dt}=-\sum_{i=1}^{M}\frac{dV_{i}^{H}}{du_{i}^{H}}\frac{du_{i}^{H}}{dt}\frac{du_{i}^{H}}{dt}$$

$$\frac{dE}{dt}=-\sum_{i=1}^{M}\frac{dV_{i}^{H}}{du_{i}^{H}}(\frac{du_{i}^{H}}{dt})^{2} \tag{19}$$

Since:

$$V_{i}^{H}=g(u_{i}^{H}) \tag{20}$$

If the output function is a Sigmoid or hyperbolic tangent function, then:

$$\frac{dV_{i}^{H}}{du_{i}^{H}}\,is\,always\,positive$$

Then:

$$\frac{dE}{dt}\,is\,always\,negative\,in\,the\,forward\,path$$

Now consider the backward path:
Because the outputs of hidden nodes are constant in the backward path, then:

$$\frac{dV_{i}^{H}}{dt}=0 \tag{21}$$

Assume that all the weights are symmetric, then:

$$w_{ij}^{H} = w_{ji}^{V} \tag{22}$$

Therefore, Equation 12 can be reduced to the following:

$$\frac{dE}{dt} = -\sum_{j=1}^{L}\sum_{i=1}^{M} w_{ji}^{V}\frac{dV_j^V}{dt}V_i^H - \sum_{j=1}^{L}\frac{dV_j^V}{dt}\theta_j^V \tag{23}$$

and can be simplified to:

$$\frac{dE}{dt} = -\sum_{j=1}^{L}\frac{dV_j^V}{dt}(\sum_{j=1}^{M} w_{ji}^{V}V_i^H + \theta_j^V) \tag{24}$$

Further simplified to:

$$\frac{dE}{dt} = -\sum_{i=1}^{L}\frac{dV_j^V}{dt}\frac{du_j^V}{dt} \tag{25}$$

Using Chain Rule:

$$\begin{aligned}\frac{dE}{dt} &= -\sum_{j=1}^{L}\frac{dV_j^V}{du_j^V}\frac{du_j^V}{dt}\frac{du_j^V}{dt}\\ \frac{dE}{dt} &= -\sum_{j=1}^{L}\frac{dV_j^V}{du_j^V}(\frac{du_j^V}{dt})^2\end{aligned} \tag{26}$$

Since:

$$V_j^V = g(u_j^V) \tag{27}$$

If the output function is a Sigmoid or hyperbolic tangent function, then:

$$\frac{dV_j^V}{du_j^V}\, is\, always\, positive$$

Then:

$$\frac{dE}{dt}\, is\, always\, negative\, in\, the\, backward\, path$$

Therefore, combining the forward and backward path, the energy function E of Eqn.11 is a Lyapunov function if and only if all the weights are symmetric.

The proposed RHN is a dynamic system. Therefore, it has attractors toward which a system tends to evolve for a wide variety of the system's initial conditions. The existence of a basin of attraction for each attractor guarantees that any initial condition in the nearby region will iterate to the attractor. When an input vector in a specific basin of attraction is presented, the proposed network sends signals back and forth between the hidden and visible layers until all the nodes reach an equilibrium state minimizing the energy function above, generating the desired output vector.

The visible nodes can be divided into either A input and B output nodes, as shown in Figure 3. In the forward computation path, each node in the hidden layer receives the weighted output of both the input and output nodes of the visible layer. In the backward computation direction, only the output nodes receive the weighted output of the hidden nodes. Therefore, when an input vector is presented to the input nodes, signals are sent back and forth between the hidden and output nodes until an equilibrium state is reached.

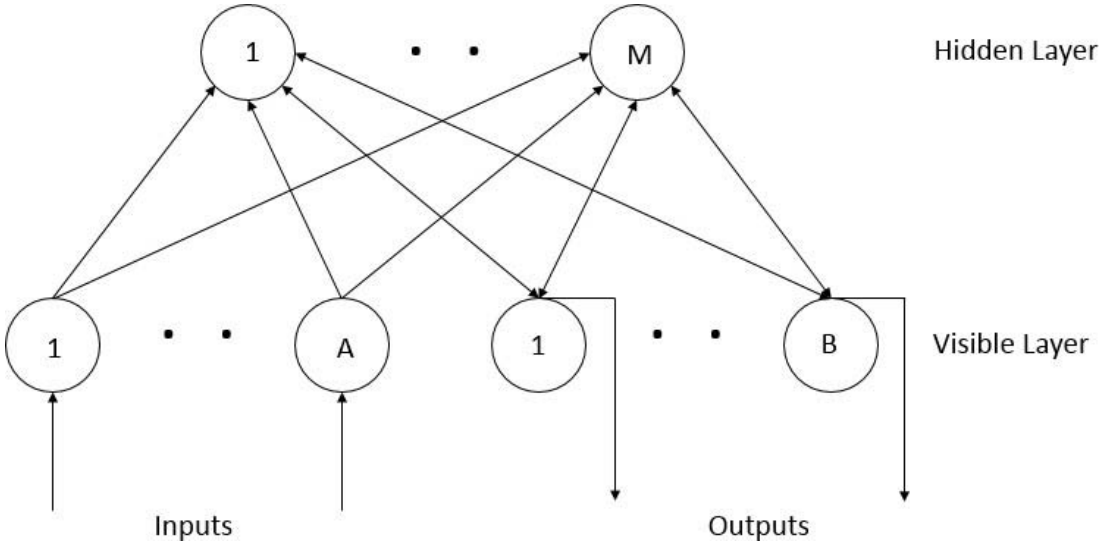

Figure 3: A Restricted Hopfield Network with Inputs and Outputs Nodes

The following differential equations describe the dynamics of the network:

In the forward path:

$$\frac{du_i^H}{dt} = -\sum_{j=1}^{B} w_{ij}^H V_j^O + \sum_{k=1}^{H} w_{ik}^H I_k + \theta_i^H \quad (28)$$

$$V_i^H = g(u_i^H)$$

Initial conditions:

$$V_j^O(0) = 0$$
$$V_i^H(0) = 0 \quad (29)$$

Where u_i^H is the sum of all inputs to the hidden nodes, W_{ij}^H is the weight connecting the output of output node *j* to the input of hidden node *i*, V_j^O is the output of output node *j*, W_{ik}^H is the weight connecting the output of input node k to the input of hidden node *i*, I_k is the output of input node *k*, θ_i^H is the threshold of hidden node *i*, V_i^H is the output of hidden node *i*, and *g*() is the output function of hidden node *i*.

In the backward path:

$$\frac{du_j^O}{dt} = \sum_{i=1}^{M} w_{ji}^O V_i^H + \theta_j^O \quad (30)$$

$$V_j^O = g(u_j^O)$$

Where u_j^O is the sum of all inputs to the output nodes, w_{ji}^O is the weight connecting the output of hidden node *i* to the input of output node *j*, V_i^H is the output of hidden node *i*, θ_j^O is the threshold of output node *j*, V_j^O is the output of output node *j*, and *g*() is the output function of output node *j*.

The proposed RHN always generates analog outputs between 0 and 1 when *g*() is a sigmoid function or between −1 and 1 when g() is a hyperbolic tangent function.

Using the Lyapunov Direct Method, it can be shown that Eqn. 31 is the energy or Lyapunov functions of the proposed network.

$$E = -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{B} w_{ij}^H V_i^H V_j^O - \frac{1}{2}\sum_{j=1}^{B}\sum_{i=1}^{M} w_{ji}^O V_j^O V_i^H$$
$$-\sum_{i=1}^{M}\sum_{k=1}^{A} w_{ik}^H V_i^H I_k - \sum_{i=1}^{M} V_i^H \theta_i^H - \sum_{i=1}^{B} V_i^O \theta_i^O \quad (31)$$

Differentiating *E*, we get:

$$\frac{dE}{dt} = -\frac{d}{dt}(\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{B} w_{ij}^H V_i^H V_j^O) - \frac{d}{dt}(\frac{1}{2}\sum_{j=1}^{B}\sum_{i=1}^{M} w_{ji}^O V_j^O V_i^H)$$
$$-\frac{d}{dt}(\sum_{i=1}^{M}\sum_{k=1}^{A} w_{ik}^H V_i^H I_k) - \frac{d}{dt}(\sum_{i=1}^{M} V_i^H \theta_i^H) - \frac{d}{dt}(\sum_{i=1}^{B} V_i^O \theta_i^O) \quad (32)$$

Expanding all the above terms in Eqn.32, we get:

$$\frac{d}{dt}(\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{B}w_{ij}^{H}V_i^{H}V_j^{O}) = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{B}w_{ij}^{H}\frac{dV_i^{H}}{dt}V_j^{O} + \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{B}w_{ij}^{H}V_i^{H}\frac{dV_j^{O}}{dt}$$

$$\frac{d}{dt}(\frac{1}{2}\sum_{j=1}^{B}\sum_{i=1}^{M}w_{ji}^{O}V_j^{O}V_i^{H}) = \frac{1}{2}\sum_{j=1}^{B}\sum_{i=1}^{M}w_{ji}^{O}\frac{dV_j^{O}}{dt}V_i^{H} + \frac{1}{2}\sum_{j=1}^{B}\sum_{i=1}^{M}w_{ji}^{O}V_j^{O}\frac{dV_i^{H}}{dt} \quad (33)$$

$$\frac{d}{dt}(\sum_{i=1}^{M}\sum_{k=1}^{A}w_{ik}^{H}V_i^{H}I_k) = \sum_{i=1}^{M}\sum_{k=1}^{A}w_{ik}^{H}\frac{dV_i^{H}}{dt}I_k + \sum_{i=1}^{M}\sum_{k=1}^{A}w_{ik}^{H}V_i^{H}\frac{dI_k}{dt}$$

For constant input Ik:

$$\frac{dI_k}{dt} = 0 \quad (34)$$

Then:

$$\frac{d}{dt}(\sum_{i=1}^{M}\sum_{k=1}^{A}w_{ik}^{H}V_i^{H}I_k) = \sum_{i=1}^{M}\sum_{k=1}^{A}w_{ik}^{H}\frac{dV_i^{H}}{dt}I_k \quad (35)$$

For the rest of the terms, we get:

$$\frac{d}{dt}\sum_{i=1}^{M}V_i^{H}\theta_i^{H} = \sum_{i=1}^{M}\frac{dV_i^{H}}{dt}\theta_i^{H}$$
$$\frac{d}{dt}\sum_{i=1}^{B}V_i^{O}\theta_i^{O} = \sum_{i=1}^{B}\frac{dV_i^{O}}{dt}\theta_i^{O} \quad (36)$$

Now consider the forward path:
Because the outputs of visible nodes are constant in the backward path, then:

$$\frac{dV_j^{O}}{dt} = 0 \quad (37)$$

Assume that all the weights are symmetric, then:

$$w_{ij}^{H} = w_{ji}^{O} \quad (38)$$

Therefore, Equation 32 can be reduced to:

$$\frac{dE}{dt} = -\sum_{i=1}^{M}\sum_{j=1}^{B}w_{ij}^{H}\frac{dV_i^{H}}{dt}V_j^{O} - \sum_{i=1}^{M}\sum_{k=1}^{A}w_{ik}^{H}\frac{dV_i^{H}}{dt}I_k - \sum_{i=1}^{M}\frac{dV_i^{H}}{dt}\theta_i^{H} \quad (39)$$

and can be simplified to:

$$\frac{dE}{dt} = -\sum_{i=1}^{M}\frac{dV_i^{H}}{dt}(\sum_{j=1}^{B}w_{ij}^{H}V_j^{O} + \sum_{k=1}^{A}w_{ik}^{H}I_k + \theta_i^{H}) \quad (40)$$

Further simplified to:

$$\frac{dE}{dt} = -\sum_{i=1}^{M}\frac{dV_i^{H}}{dt}\frac{du_i^{H}}{dt} \quad (41)$$

Using Chain Rule:

$$\frac{dE}{dt} = -\sum_{i=1}^{M}\frac{dV_i^{H}}{du_i^{H}}\frac{du_i^{H}}{dt}\frac{du_i^{H}}{dt}$$
$$\frac{dE}{dt} = -\sum_{i=1}^{M}\frac{dV_i^{H}}{du_i^{H}}(\frac{du_i^{H}}{dt})^2 \quad (42)$$

Since:

$$V_i^{H} = g(u_i^{H}) \quad (43)$$

If the output function is a sigmoid or hyperbolic tangent function, then:

$$\frac{dV_i^{H}}{du_i^{H}}\, is\, always\, positive \quad (44)$$

Then:

$$\frac{dE}{dt} \, is \, always \, negative \, in \, the \, forward \, path \tag{45}$$

Now consider the backward path:
Because the outputs of hidden nodes are constant in the backward path, then:

$$\frac{dV_i^H}{dt} = 0 \tag{46}$$

Assume all the weights are symmetric:

$$w_{ji}^O = w_{ij}^H \tag{47}$$

Therefore,

$$\frac{dE}{dt} = -\sum_{j=1}^{B}\sum_{i=1}^{M} w_{ji}^O \frac{dV_j^O}{dt} V_i^H - \sum_{j=1}^{B} \frac{dV_j^O}{dt}\theta_j^O \tag{48}$$

and can be simplified to:

$$\frac{dE}{dt} = -\sum_{j=1}^{B} \frac{dV_j^O}{dt}(\sum_{i=1}^{M} w_{ji}^O V_i^H + \theta_j^O) \tag{49}$$

Further simplified to:

$$\frac{dE}{dt} = -\sum_{j=1}^{B} \frac{dV_j^O}{dt}\frac{du_j^O}{dt} \tag{50}$$

Using Chain Rule:

$$\frac{dE}{dt} = -\sum_{j=1}^{B} \frac{dV_j^O}{du_j^O}\frac{du_j^O}{dt}\frac{du_j^O}{dt}$$

$$\frac{dE}{dt} = -\sum_{j=1}^{B} \frac{dV_j^O}{du_j^O}(\frac{du_j^O}{dt})^2 \tag{51}$$

Since:

$$V_j^O = g(u_j^O) \tag{52}$$

If the output function is a sigmoid or hyperbolic tangent function, then:

$$\frac{dV_j^O}{du_j^O} \, is \, always \, positive \tag{53}$$

Then

$$\frac{dE}{dt} \, is \, always \, negative \, in \, the \, backward \, path \tag{54}$$

Therefore, combining the forward and backward path, the energy function E of Eqn.31 is a Lyapunov function if and only if all weights are symmetric.

The proposed RHN is a dynamic system. When an input vector is presented to the input nodes, and the input vector is in a specific basin of attraction, the proposed network sends signals back and forth between the hidden and output nodes until the network reaches an equilibrium state or the corresponding attractor.

IV. Training of Proposed RHN

The proposed network can be trained using either the modified SPSA or BPTT algorithms to ensure that all the weights are symmetric, as described in the following.

a) SPSA Algorithm

As mentioned earlier (Equations 3 and 4), the conventional Hebbian rule is not suitable for training the proposed network because the outputs of the hidden nodes are unknown, and the equations cannot handle analog quantities.

The simultaneous perturbation stochastic approximation (SPSA) algorithm uses a gradient approximation that requires only 2N objective function measurements over all N iterations regardless of the optimization network's dimension [4, 9]. Therefore, the SPSA algorithm, as shown in the following formulae, is suited for a high-dimensional optimization problem of minimizing an objective function dependent on multiple adjustable symmetric weights.

$$\Delta W_{ij} = \frac{J(W(k) + c(k)\Delta(k)) - J(W(k) - c(k)\Delta(k))}{2c(k)\Delta_{ij}(k)} \tag{55}$$

$$W_{ij}(k+1) = W_{ji}(k+1) = W_{ij}(k) - a(k)\Delta W_{ij}(k)$$

At each iteration, a simultaneous perturbation delta vector with mutually independent zero-mean random variables is generated; each element $\Delta_{ij}(k)$ in $\Delta(k)$ matrix is generated with a probability of 0.5 of being either $+1$ or 1. Two weight matrices $W+$ and W are calculated by adding and subtracting the $\Delta(k)$ matrix scaled by gain sequence $c(k)$ to/from the current weight matrix $W(k)$ to compute their respective contributions $J(W+)$ and $J(W)$ to the objective function. Dependent on the outcome of the evaluation and scaled by gain sequences $a(k)$ and $c(k)$, the current weight matrix W is updated accordingly. The gain sequences $a(k)$ and $c(k)$ decrease as the number of iterations k increases, will converge to 0 as k approaches ∞.

The objective function J used for the optimization of the proposed RHN is:

$$J = \sum_{i=1}^{P}\sum_{j=1}^{B}(\hat{D}_{ij} - V_{ij}^{O}(k))^2 \tag{56}$$

Where $\hat{D}_{ij}$ is the j^{th} element of the desired output vector i, $V_{ij}^{O}(k)$ is the output value of the j^{th} output node when training pattern i is presented, B is the number of output nodes, and P is the number of training patterns.

b) Backward Propagation Through Time (BPTT) Algorithm

For simplicity and the ability to apply BPTT training algorithm, the proposed network in Figure 3 can be transformed into a discrete-time system and unfolded in time, as shown in Figure 4.

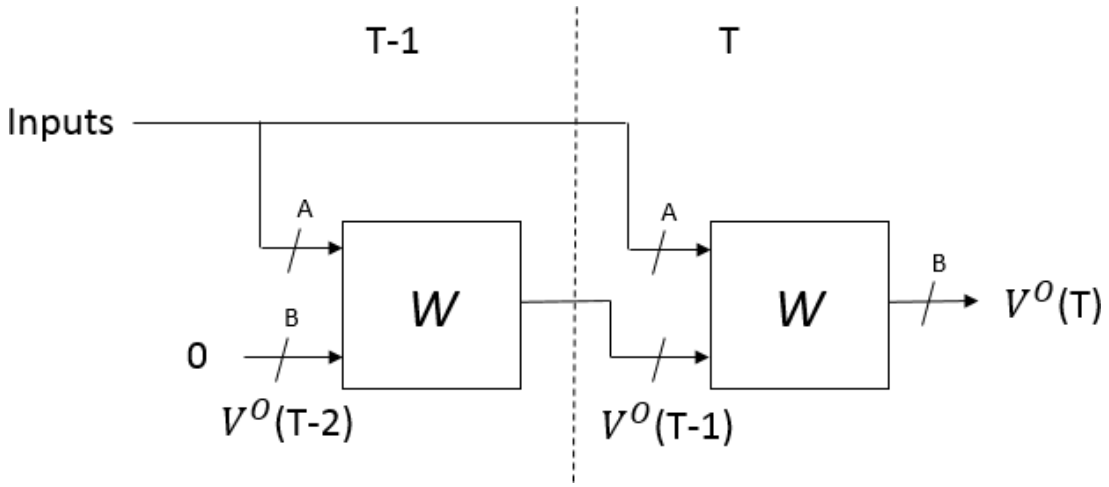

Figure 4: Unfolded RHN in 2-time steps

W represents all the weights in the proposed RHN. $V^O(T)$ and $V^O(T-1)$ are the outputs of the output nodes at T and $T-1$, respectively. The structure of the block is presented in Figure 5.

The error function J used for the optimization of the proposed RHN is:

$$J = \frac{1}{2}\sum_{p=1}^{P}\sum_{j=1}^{B}(\hat{D}_{pj} - V_{pj}^{O}(T))^2 \tag{57}$$

Where $\hat{D}_{pj}$ is the j^{th} element of the desired output vector k, $V^O(T)$ is the output value of the j^{th} output node when training pattern p is presented, B is the number of output nodes, and P is the number of training patterns.

Applying the BPTT algorithm,

$$\Delta W_{ji}^{O} = \varepsilon(\delta_{pj}^{O} V_{pj}^{H}(T) + \delta_{pj}^{H2} V_{i}^{H}(T-1)) \tag{58}$$

For $j = 1 \ldots A$:

$$\Delta W_{ij}^{H} = \varepsilon(\delta_{pi}^{H1} + \delta_{pi}^{H3}) I_{pj}^{O} \tag{59}$$

For $j = A+1 \ldots A+B, k = 1 \ldots B$:

$$\Delta W_{ij}^{H} = \varepsilon(\delta_{pi}^{H1} V_{pk}^{O}(T-1) + \delta_{pi}^{H3} V_{pk}^{O}(T-2)) \tag{60}$$

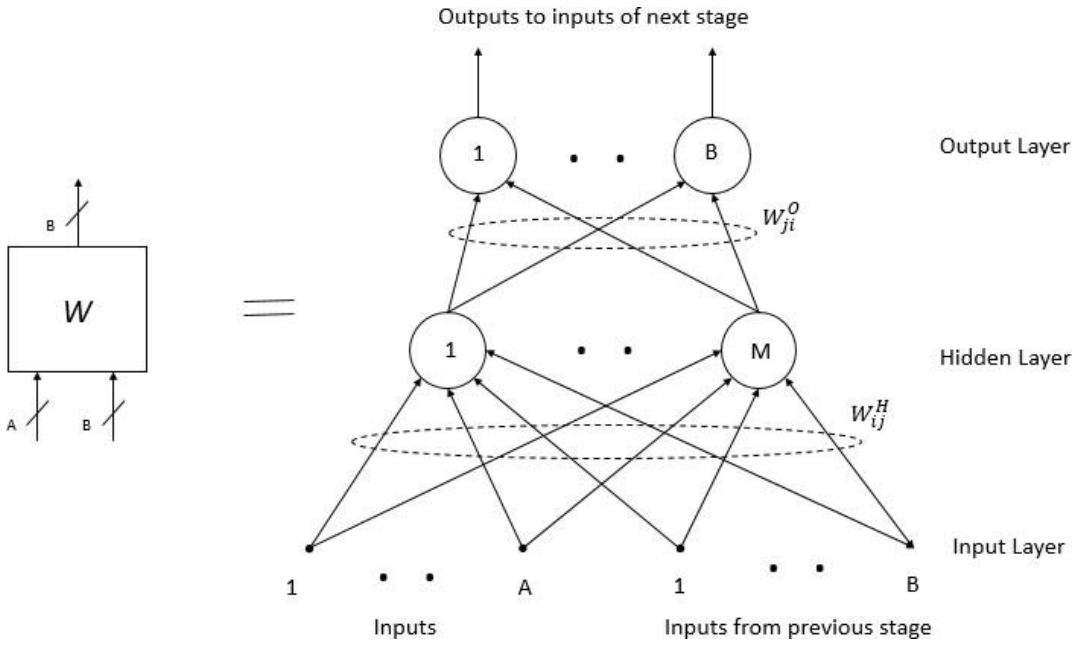

Figure 5: Structure of block W in Figure 4

Where the deltas are:

$$\begin{aligned}
\delta_{pj}^{O} &= g'(u_{pj}^{O}(T))(\hat{D}_{pj} - V_{pj}^{O}(T)) \\
\delta_{pi}^{H1} &= g'(u_{pi}^{H}(T)) \sum_{j=1}^{B} \delta_{pj}^{O} W_{ji}^{O} \\
\delta_{pj}^{H2} &= g'(u_{pj}^{O}(T-1)) \sum_{i=1}^{M} \delta_{pj}^{H1} W_{ij}^{H} \\
\delta_{pi}^{H3} &= g'(u_{pj}^{H}(T-1)) \sum_{i=1}^{B} \delta_{pj}^{H2} W_{ji}^{O}
\end{aligned} \tag{61}$$

Since the weights are symmetric, then:

$$W_{ji}^{O}(T+1) = w_{ij}^{H}(T+1) = w_{ji}^{O}(T) + \Delta W_{ji}^{O} + \Delta W_{ij}^{H} \tag{62}$$

V. Simulation Results

a) Comparison of Hopfield Network and RHN

Let us consider storing 3 vectors ([1100], [0110], [0101]) in a Hopfield network. Hopfield network can be programmed to memorize these three vector patterns using Hebbian Rule, and the weight matrix is:

$$\begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

The weight matrix will generate the correct output vector when the corresponding input vector with noise is provided.

Let us consider storing the fourth vector [1111] in the network. The weight matrix to store vector [1111] is:

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Adding this new weight matrix to the previous weight matrix of the Hopfield network programmed to store 3 vectors ([1100], [0110], [0101]) yields a weight matrix with zero elements, erasing all the previous programming, as shown in the following. Therefore, the Hopfield network cannot be programmed to remember all four vectors ([1100], [0110], [0101], [1111]) using Hebbian rule.

$$\begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

However, by introducing five hidden nodes in the proposed RHN, all the vectors can be easily stored and recalled correctly. The simulation shows that it is possible to increase the Hopfield network's memory capacity by using more hidden nodes. By increasing the number of hidden nodes to 50, the proposed network can remember all 16 binary vectors.

b) EXOR Problem

The EXOR or exclusive-or problem is a classical problem in neural network research. It is a problem of using a neural network to predict EXOR logic gates' outputs given two binary inputs. The network should return a "1" if the two inputs are not equal and a "0" if they are similar. The EXOR problem appears to be simple. However, Minksy and Papert in 1969 showed that this was a big problem for neural network architectures of the 1960s, providing a good test for the proposed network [18].

Using an RHN with input and output nodes, as shown in Figure 3, is used. An RHN with 2 input nodes, one output node, and four hidden nodes was created. It was trained to learn the EXOR problem. Figure 6 shows the training curve of the network using the SPSA algorithm described earlier. The network is trained after 1000 training iterations.

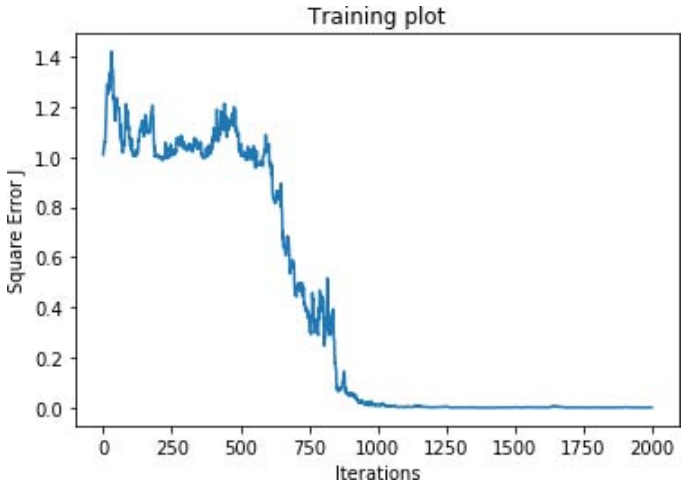

Figure 6: The training plot of square error J

Figures 7 and 8 show the basin of attraction and the network's energy profile, indicating four attractors, one attractor for each input pair.

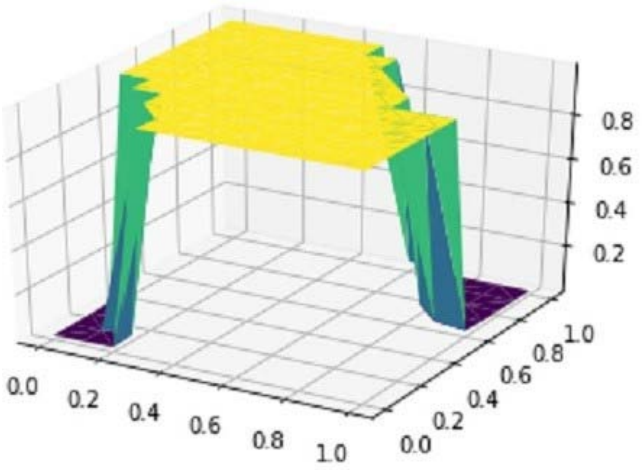

Figure 7: Basin of attraction of an RHN implementing EXOR function

Table 1 illustrates the effect of feedback as the output of an RHN network evolves when an initial input is presented. The EXOR output takes 3 to 4 feedback iterations to settle on the correct result.

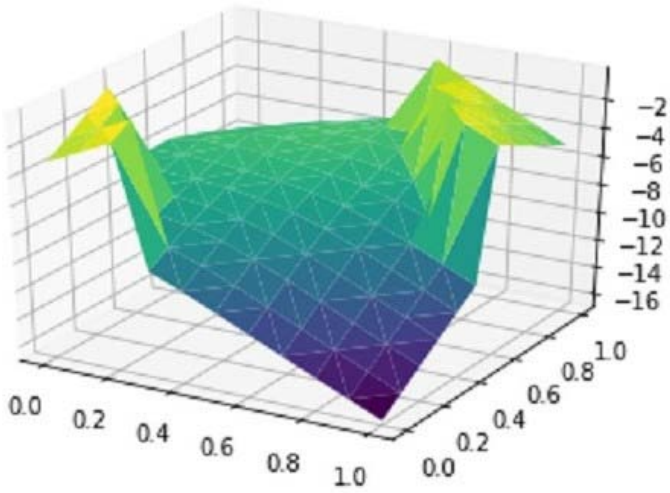

Figure 8: The energy function of an RHN implementing EXOR function

Table 1: The evolution of the RHN's output

Input		Output at each iteration					
		T = 0	T = 1	T = 2	T = 3	T = 5	T = 6
0	0	0	0.004	0.004	0.004	0.004	0.004
0	1	0	0.258	0.510	0.638	0.847	0.997
1	0	0	0.316	0.521	0.851	0.990	0.995
1	1	0	0.003	0.003	0.003	0.003	0.003

c) *Associative Memory Problem*

An RHN with 35 input nodes, 35 output nodes, and ten hidden nodes is created. The network is then trained to perform auto-encoding of *A, U, T, S* characters, each with 5 7 pixels. The input characters are distorted by changing some of the pixels to test the network's ability to re-create the characters in the presence of noise.

In Figure 9, the distorted images of *A, U, T, S* are on the left and re-created images are on the right. The figure shows that the network can perform perfect re-creation of these images even when distorted images with the Hamming distance of 13 are presented.

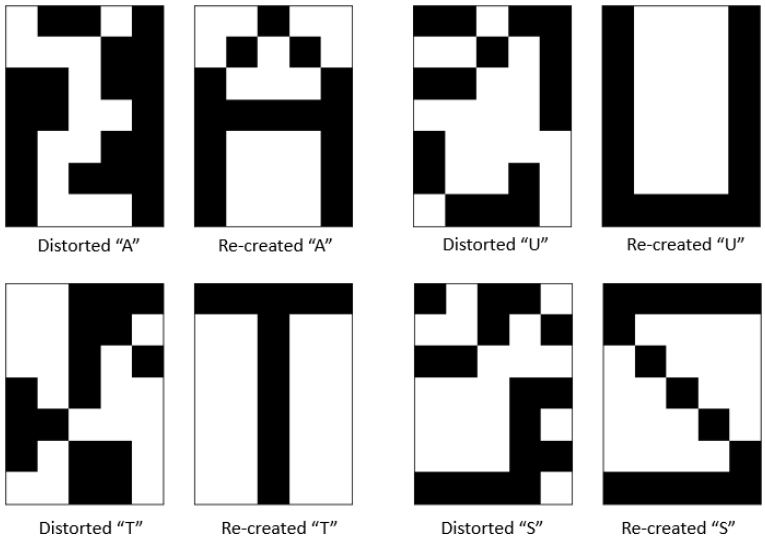

Figure 9: Distorted and re-created images of A, U, T, and S

As shown in Figure 10, an RHN trained with the BPTT algorithm can re-create the images with an average error rate of 2.4% when the Hamming distance is 5. The result is significantly improved compared with the RHN trained with the SPSA algorithm resulting in an error pattern of more than 4.4% for the same input. The classical Hopfield network and RBM can only achieve an error rate of 22.6% and 13.4%, respectively, for input vectors with the same Hamming distance. The RHN performs better compared to the Hopfield network and RBM.

d) Creating an Associative Memory Model of Hand Written Digits

The MNIST handwritten digit classification problem is a standard dataset used in computer vision and deep learning. The dataset contains 60,000 images, typically split into 50,000 training images and 10,000 validation images.

In this example, instead of performing handwritten classification, some of the 50,000 MNIST images are used as training inputs to associate with 3 5 pixel models representing handwritten digits, as shown in Figure 11. All 10,000 validation MNIST images will be used as test images to verify the network's associative function

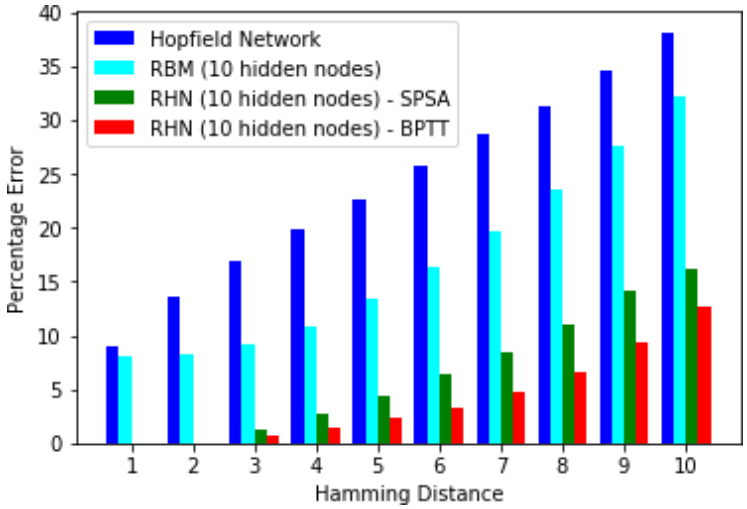

Figure 10: Performance comparison of the Hopfield Network and RBM with RHN

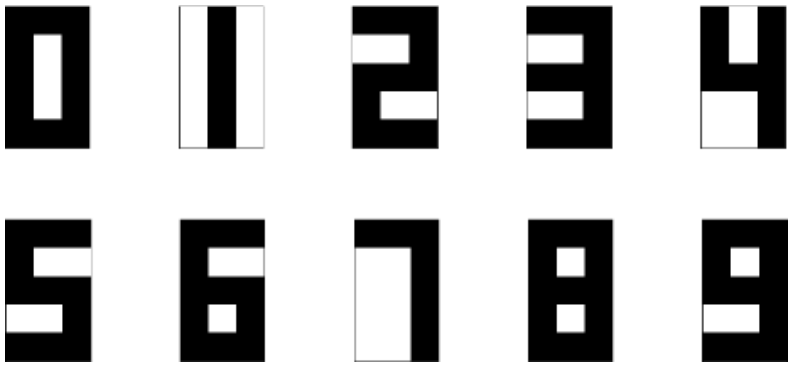

Figure 11: 3 × 5 pixel model of digits

The architecture to process MNIST images is inspired by a Convolution Neural Network architecture [19, 20], as shown in Figure 12. It consists of several layers: Input, 2D-Convolution, ReLU (Rectified Linear Unit), Max-pooling, Matrix to Vector, and RHN. The function of each layer in more detail is presented as follows:

- The Input layer will hold the raw pixel values of the image, in this case, an image of width 28, height 28;
- The 2D-Convolution layer will compute the output of nodes that are connected to local re- gions in the input image, each computing a dot product between their weights and a small area they are connected to in the input. Since four filters (horizontal line, vertical line, 45-degree line, and 135-degree line detection) are used in the architecture, this results in a volume equal to [26 × 26 × 4];
- The ReLU layer will apply an elementwise activation function, max(0,x), thresholding at zero. The layer leaves the size of the volume unchanged ([26 × 26 × 4]).
- The Max-pooling layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in a smaller volume of [7 × 7 × 4].
- The Matrix to Vector layer will convert a volume of data (in this case [7 × 7 × 4] elements) into a linear vector of 196 elements.
- An RHN with 196 input nodes, 100 hidden nodes, and 15 output nodes will be trained using the BPTT algorithm to associate an input image with a digit model of 3 × 5 pixels.

The network architecture was inspired by the visual cortex's organization, having a similar connectivity pattern of the retina, ganglion cells, and neurons in a human brain. Individual neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. A collection of such fields overlap to cover the entire image area.

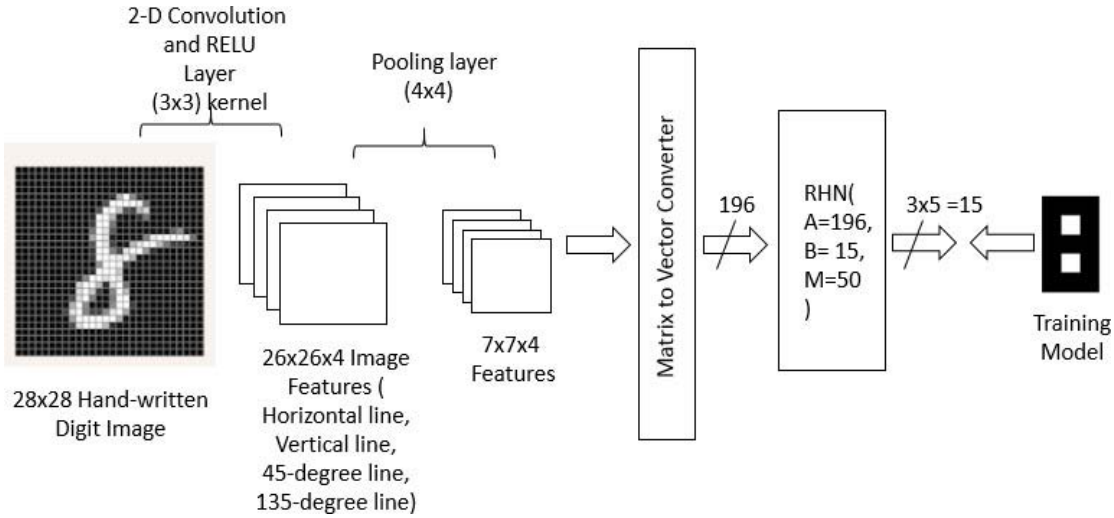

Figure 12: Image Processing and RHN

An RHN was trained with 1 percent of the 50,000 training images. Figure 13 shows the responses of an RHN when different input images are presented at zero, one, and two feedback iterations.

An RHN with zero feedback iteration is a feed-forward multilayer perceptron (MLP) acting as a mapping function. When input images "0" and "4" are presented, models of "0" and "4" with 1-bit error were

created as outputs at iteration 0. The network generated correct models of "0" and "4" after the first iteration. When input image "1" is presented, it takes two feedback iterations to generate the correct model of "1". When input image "6" is presented, the network first generated a wrong model of "5" and then corrected its output to generate the correct model of "6" after the first feedback iteration.

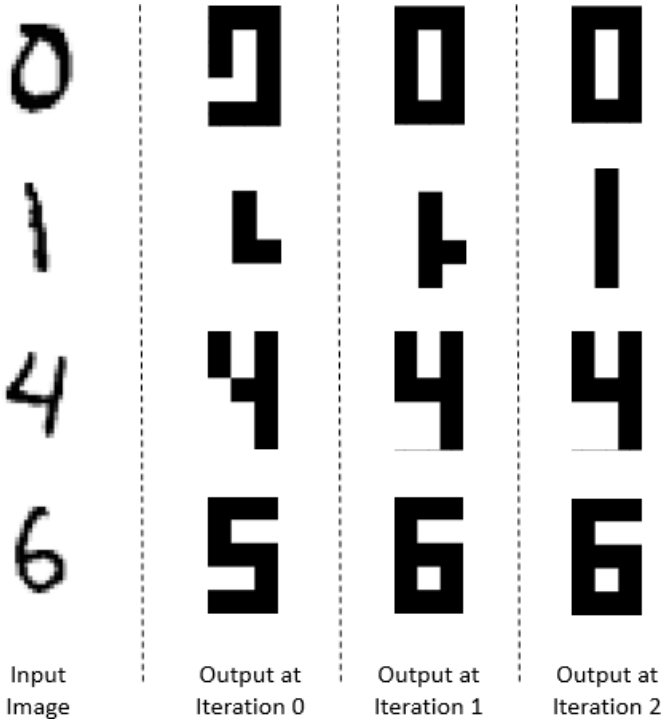

Figure 13: RHN's sample output at different iterations

The effect of feedback iterations on the performance of an RHN is illustrated in Figure 14. The figure shows that the network can re-create half of the models correctly without any feedback iteration. However, the network's performance can be improved by applying one or more feedback iterations. This result illustrates the importance of feedback iteration in implementing an associative memory using the proposed RHN.

Figure 15 shows a network's performance after being trained using only 500 and 5000 training images. The graph shows that the network could re-create 6674 of the images perfectly when 500 training images were used, while the network can re-create 7849 of the images perfectly when 5000 training images were used.

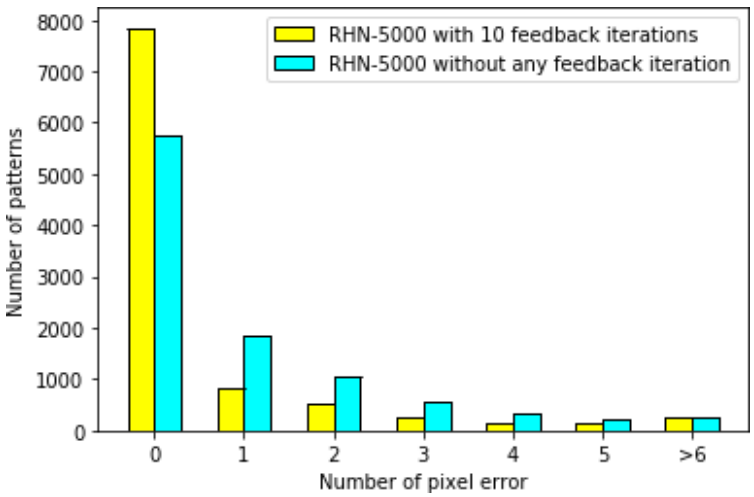

Figure 14: Effect of feedback iterations on the performance of an RHN

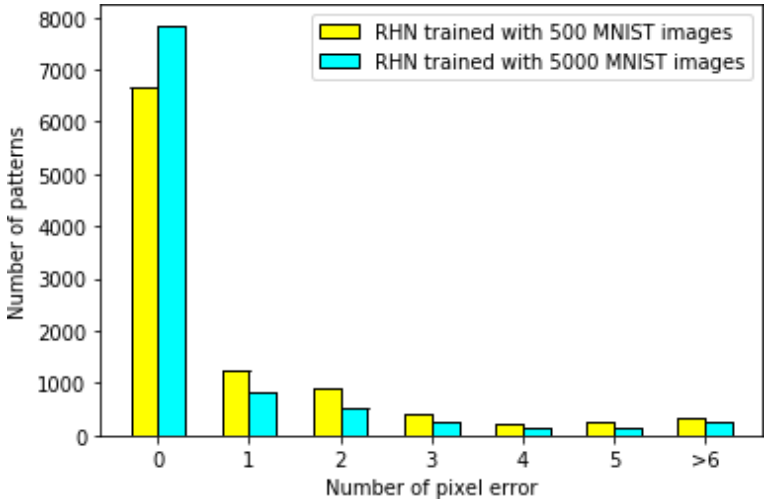

Figure 15: Performance of RHNs trained with 500 and 5000 MNIST training images

VI. Conclusion

A trainable analog Restricted Hopfield Network is presented in this paper. An energy or Lyapunov function was derived to show that the proposed network will converge to stable states when an input vector is introduced. The proposed network can be trained using either the modified SPSA or BPTT algorithms to ensure that all the weights are symmetric. Simulation results show that the presence of hidden nodes increases the network's memory capacity. Using A, U, T, S as training characters, the network can be trained to be an associative memory. Simulation results show that the network can perform perfect re-creation of noisy images and perform better than the standard Hopfield Network and RBM. Simulation results also illustrate the importance of feedback iteration in implementing associative memory to re-create from noisy images.

References Références Referencias

1. John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
2. John J Hopfield and David W Tank. "neural" computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.
3. John J Hopfield and David W Tank. Computing with neural circuits: A model. *Science*, 233(4764):625–633, 1986.
4. ROBERTJ McEliece, Edwardc Posner, EUGENER Rodemich, and SANTOSHS Venkatesh. The capacity of the hopfield associative memory. *IEEE transactions on Information Theory*, 33(4):461–482, 1987.
5. Amos J Storkey and Romain Valabregue. The basins of attraction of a new hopfield learning rule. *Neural Networks*, 12(6):869–876, 1999.
6. Elizabeth Gardner. Maximum storage capacity in neural networks. *EPL (Europhysics Letters)*, 4(4):481, 1987.
7. Elizabeth Gardner. The space of interactions in neural network models. *Journal of physics A: Mathematical and general*, 21(1):257, 1988.
8. Elizabeth Gardner and Bernard Derrida. Optimal storage properties of neural network models. *Journal of Physics A: Mathematical and general*, 21(1):271, 1988.
9. David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.
10. Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR, 2009.
11. Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.
12. Ilya Sutskever, Geoffrey E Hinton, and Graham W Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in neural information processing systems*, pages 1601–1608, 2009.
13. Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798, 2007.
14. James C Spall et al. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.
15. James C Spall. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins apl technical digest*, 19(4):482–492, 1998.
16. Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10): 1550–1560, 1990.

17. Simon Haykin. N*eural networks and learning machines, 3/E*. Pearson Education India, 2010.
18. Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.
19. Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Back propagation applied to handwritten zip code recognition. *Neural computation*, 1(4): 541–551, 1989.
20. Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.