International Journal of Computer Applications (0975 – 8887)
Volume 170 – No.10, July 2017

Comparative Analysis of Recurrent Networks for Pattern Storage and Recalling of Static Images

Jay Kant Pratap Singh Yadav
Assistant Professor
Department of CSE
NIET, Greater Noida

Laxman Singh
Professor
Department of ECE
NIET, Greater Noida

Zainul Abdin Jaffery
Professor
Department of EE
Jamia Millia Islamia University

ABSTRACT

Auto associative memory is widely used network for pattern storage and recalling of patterns. Hopfield network, Hamming network are popularly known auto associative memory networks. In this paper we present comparative analysis in term of storage and recalling efficiency of Hopfield network and Hamming network and we choose images of letters. The results of the simulation for Hopfield and hamming network for character recognition under high noise are delineated and mentioned.

General Terms

Pattern Recognition, Neural Network, Hopfield Network, Hamming Network.

Keywords

Auto associative, Pattern, Recurrent network, Hebbian rule

1. INTRODUCTION

Here we take Hopfield network and Hamming network for performance analysis. This paper is organized in five sections. Section II describes structure and properties of Hopfield network .The method of weight matrix calculation are presented and its properties is considered and analyzed. In section III Hamming network is presented, its structure and properties are considered; algorithm of weight adjustment is described. In section IV simulation results of Hopfield and Hamming network with different level of noise is presented. Section V is used for conclusion.

2. HOPFIELD NETWORK

Hopfield Neural Network is an example of a network that can be defined as a dynamic feedback system, where the output of a fully direct operation serves as an input for the next operation of a network as shown in Fig 1. Networks that work with feedback is recurrent networks or feedback networks. Each direct operation of a network is called iteration. Like all other nonlinear dynamic systems, recurrent networks are capable of showing the whole variety of different behaviors. In particular, one potential pattern of behavior is that the system can be stable, i.e., it can converge at the single fixed (motionless) point. If the motionless point is an input to such a dynamic system, we will have the same point at the output. This keeps the system in the same state. Periodic cycles or chaotic behavior are also possible.

It has been shown that the Hopfield networks are stable. In general, it can be more than a fixed point. This depends on the starting point chosen for the initial iteration, to which fixed point a network converges. Motionless points are called attractors.

The set of points (vectors) attracted to a certain attractor during the iterations of a network is referred to as the "attraction area" or "attraction basin" of this attractor. The set of motionless points or fixed points of Hopfield's network works as a memory. In this case, the network can operate as associative memory. These input vectors, which enter the sphere of attraction of a separate attractor, are connected (connected) with it. For example, the attractor may be a desirable image. The area of an attraction can consist of noisy or incomplete versions of this image. There is a hope that images that vaguely recall a desirable image will be remembered by a network associated with this image.

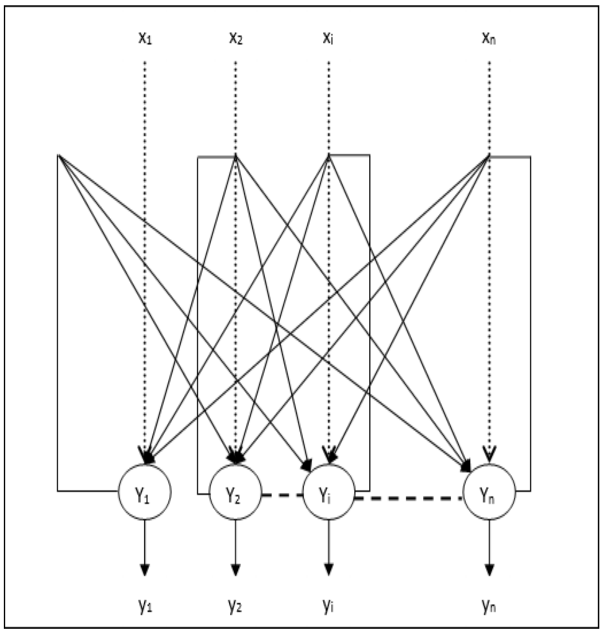


Fig 1: Architecture of Binary Hopfield neural network

In Fig. 1 the binary Hopfield network is represented. Input and output vectors are bipolar i.e., “+1”and “-1”. There is a symmetric weight matrix $W = \|w_{ij}\|$ of integers with zeros are set on a diagonal. The input vector X is multiplied by a weight matrix using conventional matrix and vector multiplication. The input vector X is fed to the corresponding neurons, and the output vector is calculated. However, for each iteration, only one component of an output vector $Y = [y_j]$ is used. This method is known as “asynchronous update”. This component which can be chosen randomly or by turn enters to a threshold element, whose output is bipolar (–1, or +1). Corresponding component of an input vector is replaced by this value and thus forms an input vector for the next iteration. The process continues until the input and output vectors become equal, i.e., the motionless point is reached. This algorithm is discussed below.

2.1 Algorithm of Asynchronous Updates

At the first moment an input vector X is fed with weight$||w_{ij}||$ to input neurons and the total signal at the input of jth neuron $S_j(x)$ is defined. Outputs of neurons are fed to their inputs. The following operations are to be made:

Calculate components of an output vector y_j, j = 1, 2, ..., n, using the formula

$$y_j = F(\sum_{i=1}^{n} w_{ij} x_i \qquad (1)$$

Where

$$F(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{yprevious} & \text{if } x = 0 \end{cases}$$

- To execute asynchronous correction, i.e. [2–4]:

Step 1: start with an input vector (x_1; x_2; ...; x_n).

Step 2: find y_j according to the formula (1).

Step 3: replace (x_1; x_2; ...; x_n) with (y_1; x_2; ...; x_n)= Y and a feed Y back to input X.

Step 4: repeat process to find y_2; y_3, etc. and replace the corresponding inputs.

Repeat steps 2–3 until the vector: Y = (y_1; y_2; ...; y_n) ceases to change.

It was proved each such step reduces the value of communications energy E if at least one of outputs has changed:

$$E = 1/2 \sum_{i=1}^{n} . \sum_{j=1}^{n} w_{ij} x_i x_j \qquad (2)$$

so convergence to a motionless point (attractor) is provided.

Asynchronous correction and zeros on a diagonal of a matrix W guarantee that power function (2) will decrease with each iteration [2][5]. Asynchronous correction is especially essential to ensuring convergence to a motionless point. If we allow whole vector to be corrected on each iteration, it is possible to receive a network with periodic cycles as terminal states of an attractor, but not with motionless points.

2.2 Patterns of Behavior of Hopfield's Network

Weight matrix distinguishes behavior of one Hopfield's network from another so there is a question: "How to define this weight matrix?" The answer is it should be given a set of certain weight vectors which are called etalon arrays. There is a hope that these etalon arrays will be the fixed points of a resultant Hopfield's network, though it is not always so. In order to ensure these etalons to be attractors, the weight matrix W =|| w_{ij}|| should be calculated so [5]:

$$w_{ij} = \sum_{k=1}^{N} (x_{ki} - 1)(x_{kj} - 1) \text{ if } i \neq j \qquad (3)$$

=0 otherwise if i=j

where N is a number of the etalon arrays, X_k is the kth etalon array.

If etalon arrays form a set of orthogonal vectors, it is possible to guarantee that if the weight matrix is determined as shown above in formula (3), each etalon vector will be a motionless point. However, generally in order that etalons become motionless points, orthogonality isn't obligatory. It should be noted that Hopfield network weights aren't trained like Back Propagation or RBF networks but are calculated in accordance with formula (3).

One of the major drawback of Hopfield network is "cross associations" in which similar type of letters cannot be recalled perfectly for example, in our experiment letter p and q have similarity so when one letter is recalled with certain noise either imperfect image of that letter is retrieved or other letter is recalled .

3. HAMMING NEURAL NETWORK

When there is no need that the network would display an etalon pattern in an explicit form, that is it is enough to define, say, a class of a pattern, associative memory is realized successfully by Hamming's network. This network is characterized, in comparison with Hopfield's network, by smaller costs of memory and volume of calculations that becomes obvious of its structure and work (Fig. 2) [2, 6].

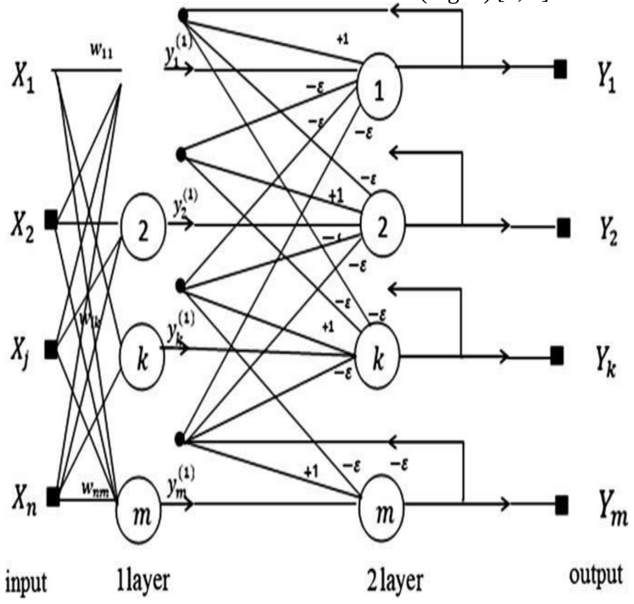


Fig 2: Architecture of Hamming network

The network consists of two layers. The first and second layers both have m neurons, where m is a number of patterns. Neurons of the first layer have n synapses connected to the network inputs (forming a fictitious zero layer). Neurons of the second layer are connected among themselves by synaptic links. The only synapse with positive feedback for each neuron is connected to his axon.

The idea of network working consists in finding of Hamming distance from the tested pattern to all patterns represented by their weights. Hamming distance is the measure of number of different bits in two binary vectors. The network has to choose a pattern which has the minimum of Hamming distance to an unknown input pattern therefore the only one of a network outputs corresponding to this pattern will be made active.

At an initialization stage the following values are assigned to weight coefficients of the first layer and a threshold of activation function (T).

$$w = \frac{x_i^k}{2} \text{ where } i = 1 \text{ to } n \text{ and } k = 1 \text{ to } m \qquad (4)$$

$$T_k = \frac{n}{2} \text{ where } k = 1 \text{ to } m \qquad (5)$$

Here x^k_i is the i^th an element of the k^th pattern.

Weight coefficients of the braking synapses in the second layer are assigned to some value $-\varepsilon$, where $0< \varepsilon<1/m$., where m is a number of patterns.

The neuron synapse connected with its own axon has a weight (+1).

3.1 Algorithm of work of a Hamming Network

1. Enter the unknown vector $X=\{x_i: i = 1 \text{ to } n\}$; to a network input and determine outputs of the first layer neurons (the superscript in brackets in formula (6) specifies number of a layer):

$$y_j^{(1)} = f(s_j^{(1)}) = f\left(\sum w_{ij}\, x_i + T_j\right), j = 1 \text{ to } m \qquad (6)$$

After that initialize states of axons of the second layer with received values:

$$y_j^{(2)} = y_j^{(1)}, j = 1 \text{ to } m \qquad (7)$$

2. Calculate new states of the second layer neurons

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum y_k^{(2)}(p), j = 1 \text{ to } m \qquad (8)$$

and values of their axons:

$$y_j^{(2)}(p+1) = f\left[s_j^{(2)}(p+1)\right], j = 1 \text{ to } m \qquad (9)$$

Here f is the activation function has a threshold, thus the size of a threshold should be rather big so that any possible values arguments won't lead to saturation.

Check, whether the output of the second layer neurons has changed since the last iteration. If no then stop otherwise pass to a step 2 of next iteration.

From the description of algorithm it is evident that the role of the first layer is very conditional and limited, having used once on a step 1 value of its weight coefficients, the network doesn't come back to it anymore, so that the first layer may in general be removed from the network by using a matrix of weight coefficients.

There are following advantages of Hamming neural network:

- small costs of memory;
- the network responds;
- extremely simple algorithm of work;

capacity of a network doesn't depend on dimension of an input signal (as in Hopfield's network) and exactly equals to a number of neurons.

4. SIMULATIONS OF HOPFIELD AND HAMMING NETWORKS

Simulation of Hopfield and Hamming networks are performed by implementing algorithms of both networks using MatLab R2010b on window environment. Comparative experiment of Hopfield's and Hamming's neural networks in a problem of symbols recognition were carried out. For learning of a network input sample of letters (1, 7, e, q, p) was used. Then generated noisy patterns from this sample were entered and their recognition was performed. Level of noise changed from 0 to 50 %. Results of recognition of the specified symbols are presented in Fig. 3. On the screen 4 images (patterns) are presented (from top to down): the initial image—a etalon, the noisy image, result of Hamming network, result of Hopfield's network (Figs. 3, 4, 5, 6, 7).

4.1 Analysis of Results

Results of the experiments with Hopfield's and Hamming's networks are presented in the Table 1. A corresponding table element is a result of recognition by a network.

Table 1.Comparative results of experiments with Hopfield and Hamming Networks

Recognition of letters (Hamming , Hopfield)	0%	10%	20%	30%	40%	50%
1	(1,1)	(1,0.5)	(1,1)	(1,.5)	(1,.5)	(0,0)
7	(1,1)	(1,0.5)	(1,0.5)	(1,1)	(1,0.5)	(0,0)
e	(1,1)	(1,0.5)	(1,1)	(1,1)	(0,0)	(0,0)
q	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(0,0)
p	(1,1)	(1,1)	(1,0)	(1,0)	(1,0)	(0,0)

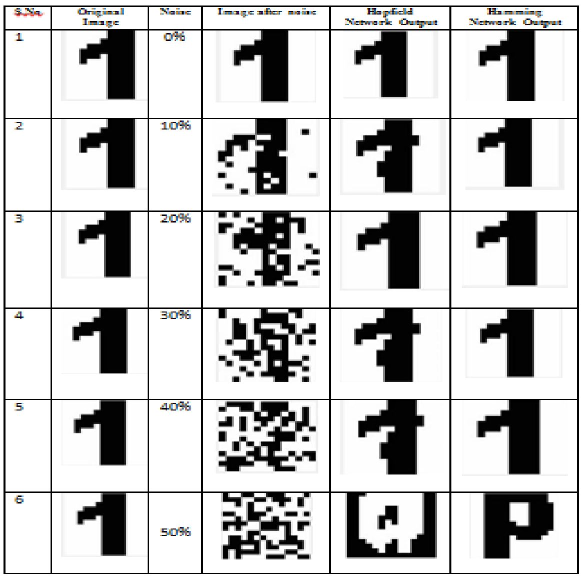
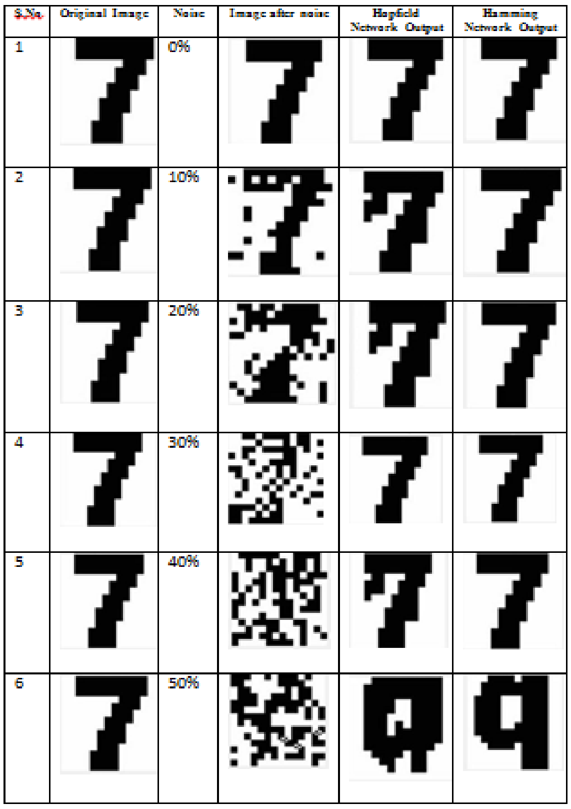
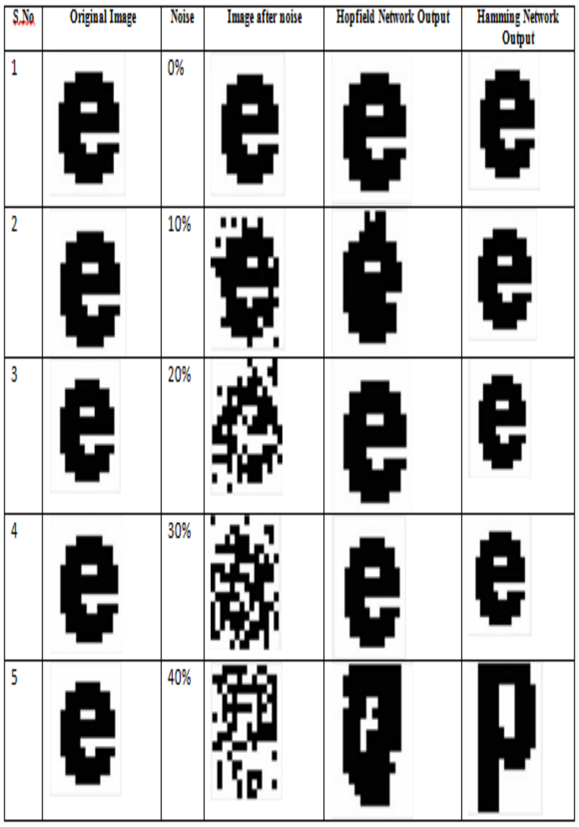

S.No.
Original Image
Noise
Image after noise
Hopfield Network Output
Hamming Network Output
1
0%
2
10%
3
20%
4
30%
5
40%
6
50%

Fig. 3. Recognition of symbol 1



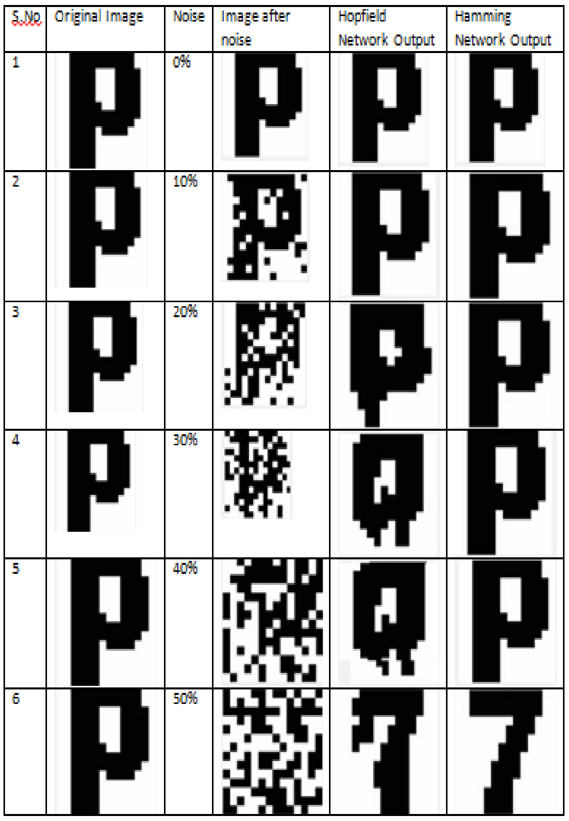
S.No.	Original Image	Noise	Image after noise	Hopfield Network Output	Hamming Network Output
1		0%			
2		10%			
3		20%			
4		30%			
5		40%			
6		50%			

Fig. 4. Recognition of symbol 7



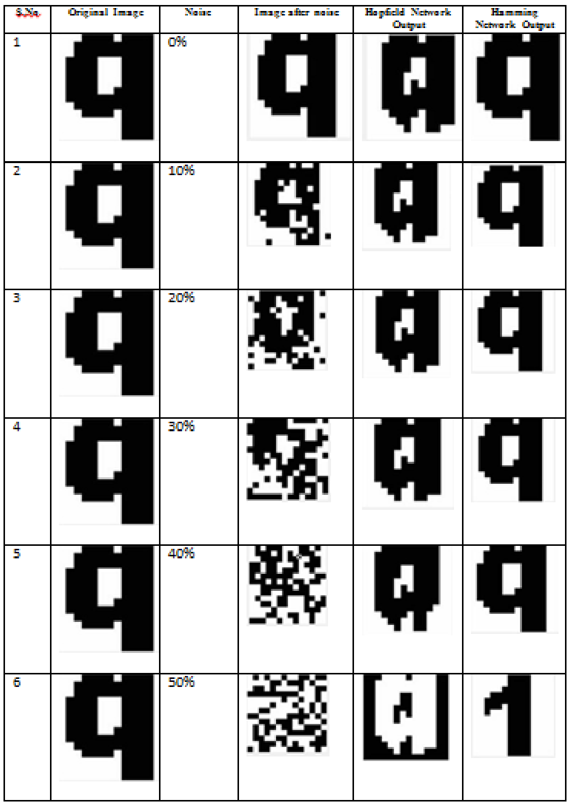
S.No	Original Image	Noise	Image after noise	Hopfield Network Output	Hamming Network Output
1		0%			
2		10%			
3		20%			
4		30%			
5		40%			

Fig. 5. Recognition of symbol e



S.No	Original Image	Noise	Image after noise	Hopfield Network Output	Hamming Network Output
1		0%			
2		10%			
3		20%			
4		30%			
5		40%			
6		50%			

Fig. 6. Recognition of symbol p



S.No.	Original Image	Noise	Image after noise	Hopfield Network Output	Hamming Network Output
1		0%			
2		10%			
3		20%			
4		30%			
5		40%			
6		50%			

Fig.7. Recognition of symbol q

5. CONCLUSION

Table 1 shows the comparative results of Hopfield and Hamming networks. Where (Hamming, Hopfield) of a symbol at the specified noise level shows recognition of a letter, namely, 0 indicates not recognized, 1 is completely recognized and 0.5 indicates recognized with defects.

Hamming network is in general perform better than Hopfield network and recognized correctly up to 40% of noise level . Hopfield network results of recognition are much worse, at recognition of symbol with a similar type letter (e, p, q) there were difficulties i.e. recognition level 30%. It is the effect of cross associations.

6. REFERENCES

[1] J.J. Hopfield, "Neural Networks and physical systems with emergent collective computational abilities," Proc. Natl. Acad. Sci. USA, 1982, 79, 2554 — 2558.

[2] I. Kanter and H. Sompolinsky, "Associative Recall of memory without errors," Phys. Rev. A, 1987, vol. 35, pp. 380-392.

[3] Y.P. Zaychenko, "Fundamentals of intellectual systems design," Kiev. Publishing House, Slovo, 2004, pp. 352. (rus).

[4] B. Yegnanarayana, Artificial Neural Networks," Prentice Hall of India, 2006.

[5] Neil Davey, S.P. Hunt and Rod Adams, "High Capacity Recurrent Associative Memories," Neurocomputing – IJON, 2004, vol. 62, pp. 459-491, DOI: 10.1016/j.neucom.2004.01.007.

[6] F.L. Chung and T. Lee, "Fuzzy competitive learning," Neural Netw. 7, 1994, 539 – 552.

[7] W. Tarkowski, M. Lewenstein and A. Nowak, "Optimal Architectures for Storage of Spatially Correlated Data in Neural Network Memories," ACTA Physica Polonica B, 1997, vol. 28, No. 7, pp. 1695 - 1705.

[8] K. Deb and D. Anand Joshi, "A. Real-coded evolutionary algorithms with parent-centric recombination," In: Proceedings of the IEEE Congress on Evolutionary Computation, 2002, pp. 61 – 66.

[9] J.J. Hopfield, "Neurons, dynamics and computation," Phys. Today, 47, 40 – 46, 1994, 78 2 Neural Networks with Feedback and Self-organization.

[10] S. Heykin, "Neural networks," Full course (2nd edn), Transl. engl., Moscow.-Publishing House Williams, 2006, pp.1104. (rus).

[11] Christophe L. Labiouse, Albert A. Salah and Irina Starikova, "The Impact of Connectivity on the Memory Capacity and the Retrieval Dynamics of Hopfield – type Networks," Proc. Of the Santa Fc Complex Systems Summer School, pp. 77-84.

[12] Mikhail Z Zgurobsky and Yuriy P Zaychenko, "The Fundamental of Computational Intelligence," Springer International Publishing Switzerland, 2016, pp.- 39-59.

[13] T. Kohonen, "Self-Organization and Associative Memory," 3rd ed., Berlin Springer-Verlag, 1989.

[14] T. Kohonen and M. Ruohonen, "Representation of Associated Data by Matrix Operators," IEEE Trans. Computers C – 22(7), pp. 701-702.

[15] N. Davey, S. Hunt and R. Adams, "High Capacity Recurrent Associative Memories," 2004, Neurocomputing – IJON 62, 459 - 491.