Studies on an Electronic Analog of a Recurrent Neural Network with Retrieval Phase Weight Adaptations

Y.V.Joshi, Anand.I., Vishwanathan Mohan[1]
Dept of Electronics and Telecommunication Engineering
SGGS Institute of Engineering and Technology, Nanded-431606, India.
(Email: vmohan@sggs.ac.in)

***Abstract*— It is argued that weight adaptations even during retrieval phase can greatly enhance the performance of a neurodynamic associative memory. Our simulations with an electronic implementation of an associative memory showed that extending the Hopfield dynamics with an appropriate adaptive law in retrieval phase could give rise to significant improvements in storage capacity and computational reliability. Weights, which are supposed to encode the information stored in the Hopfield neural network, are usually held constant once training /Storage is complete. In our case, weights also change during retrieval, hence losing information in the process, but resulting in much better retrieval of stored patterns. We describe and characterize the functional elements comprising the network, the learning system, and include the experimental results obtained from applying the network for character recognition in various noisy conditions. Stability issues emerging as a consequence of retrieval phase weight adaptation and implications of weights being used as transitory, intermediary variables are briefly discussed.**

I Introduction

Recurrent neural networks exhibiting emergent computation have given a new impetus to neural network research. In a seminal paper, Hopfield [3] demonstrated that a system of fully connected two state McCullough Pitt's neurons can exhibit emergent computational characteristics. As a generalization over the two state model, a network of neurons whose outputs can take a continuous range of values has been proposed later by the same author [5]. Such recurrent networks usually burrow a trajectory in the state space which begins with the computational problem and ends with the computational solution (where the system reaches equilibrium), user specifying the initial conditions which define where the trajectory begins. Several interesting generalizations of the Hopfield model like allowing higher order interactions among neurons [9], probabilistic updates to neuron states [10] , neurons having complex valued outputs [1] have been made by various authors. Globally stable dynamic networks that learn and adapt at the same time have been proposed by Cohen and Grossberg [4] and Kosko [7] among others. To our knowledge, an electronic neural associative memory running weight adaptations even during retrieval phase does not exist in the current literature.

A great number of past studies have shown that adaptive analog VLSI serves as an excellent medium to implement systems taking poorly conditioned inputs and performing specific perceptive tasks with an approximate level of accuracy [11-14]. Study and technical reports of ANN products indicate that development of analog hardware for neural networks is both practicable and promising. VLSI implementations of low-level biological sensory-processing systems characterized by homogeneous as well as heterogeneous connection strengths have already been developed [11,14]. In the present work the learning paradigm is an intrinsic part of the network; the weight dynamics and the forward dynamics are integral and are inseparable. Weight updates are calculated in a similar manner as the feed forward mapping, updates being parallel in nature. In such networks, the ability to process information with relatively slow and inaccurate elements in a massively parallel fashion is broadened to encompass the learning paradigm as well. Weights are normally adaptive as a part of training; on completion of training the weights are frozen and the network is used for regular pattern retrieval. In this paper we demonstrate that adapting weights in this fashion dramatically enhances the network capacity as an associative memory. Unlike a connectionist network where knowledge is encoded in the form of weights, in our network the weights also change during retrieval, hence losing information in the process but resulting in much better recognition and computational reliability.

The paper is outlined as follows. A brief description of the network dynamics is given in section II. Section III describes the architecture, weight dynamics and circuit implementation of the functional elements. Experimental results obtained by applying the network for character recognition tasks and simulation results comparing relative retrieval performance of the adaptive network with the Hopfield neural network is explained in section IV.Results are discussed in the final section.

II NETWORK DYNAMICS

(For reasons of space we refer the reader to [3, 5] for details of the Hopfield Neural Network (HNN))

A. Mathematical model

Extending the dynamics of HNN as follows can accommodate weight adaptations in retrieval phase:

$$\tau \frac{du_i}{dt} = -G_i u_i + \sum_{j=1}^{N} w_{ij} V_j \qquad (2.1)$$

$$V_i = g(\lambda u_i) \qquad (2.2)$$

$$\rho \frac{dw_{ij}}{dt} = -w_{ij} + \alpha V_i V_j \qquad (2.3)$$

Equations 2.1 and 2.2 describe the dynamics of HNN, while equation 2.3 describes the dynamics at the interconnections between the neurons. V_i is the output state of the i'th neuron; w_{ij} represents the strength of the connection between the i'th and the j'th neuron; u_i is an internal variable of i'th neuron; g(.) is known as a sigmoid function, usually chosen to be the tanh(.) function; λ is the steepness of the sigmoid function; τ and ρ are time constants of relaxation for the forward dynamics and weight dynamics respectively. Weight matrix is hermitian and Wii =0. Equation 2.1 defines the time-evolution of the state u_i at neuron unit i; the value of u_i is fed through the sigmoid $g(u_i)$ to generate the node's output signal. Weight dynamics (eqn. 2.3) take place at a slower time scale than activation dynamics ($\rho << \tau$) [6] and is consistent with the locality requirement i.e. the connection between i^{th} and j^{th} neuron depend only outputs of i^{th} and j^{th} neurons only.

$$\frac{dw_{ij}}{dt} = F\left(W_{ij}, V_i, V_j\right) \qquad (2.4)$$

B. Stability Concern

Dynamics of the system described by (Eqns 2.1, 2.2 and 2.3) is dissipative since there is a Lyapunov energy function, E, associated with it .The energy function can be obtained by adding an extra term to the Lyapunov function of HNN dynamics and is as follows:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} V_i V_j + \sum_{i=1}^{N}\left(Gi \int_0^{V_i} g^{-1}(V)dV\right) + \tfrac{1}{2}\sum_i \sum_j W_{ij} W_{ji} \qquad (2.5)$$

Proof: See Appendix

Since the network dynamics has a Lyapunov function, it has fixed points, which are interpreted as stored memories of the network. This implies that the final output V settles in a fixed point.

C. The complete scenario

The retrieval process in case of adaptive weights progresses as follows: *Initially, the weight matrix has all the stored patterns encoded in itself. That is, the initial weight matrix is calculated using Hebbian rule [2, 5].*

$$w_{ij} = \sum_{p=1}^{P} S_i^p S_j^p / N \qquad (2.6)$$

Where S^p (p = 1 to P) are the set of patterns to be stored. During retrieval, the *network is triggered by some initial condition within the neighborhood of a stored pattern, say, S^p.* In tandem with the network state dynamics (2.1,2.2), weights also change according to equation 2.3. *Finally, once the network state settles on the nearest stored pattern, S^p, the weight matrix sheds all the components other than those that correspond to S^p. When the weights converge we have,*

$$\rho \frac{dw_{ij}}{dt} = 0;\ or\ ,\ w_{ij} = \alpha V_i V_j \qquad (2.7)$$

which is nothing but the Hebb's rule. *In such a state, the network has only one stored pattern; information regarding all other stored patterns is destroyed in the process of retrieving one particular pattern. Therefore, to retrieve a different pattern in a later session, the weight matrix must be reloaded again so that all the stored patterns are encoded.*

To sum up, activation dynamics (Eqn 2.1, 2.2) compute a nonlinear mapping from input space to output space. The second unit modeled by (Eqn 2.3) coexists with the forward layer and estimates the updates for each weight in a collective and parallel fashion. Each processing element that computes the weight w_{ij} receives as input, v_i, v_j which are the output of i^{th} and j^{th} neurons respectively. This layer is referred to as the adaptive layer. Initial storage of patterns is one-shot and adaptations are carried out online, in retrieval phase only. Forward dynamics and Weight dynamics process simultaneously and form the entire network i.e. the nonlinear mapping and adaptivity are imbedded in the same network.

III CIRCUIT IMPLEMENTAION

A simple circuit model of (Eqns. 2.1, 2.2) is shown in figure 1. However, preliminary simulations showed that the model shown in Fig. 1 is not suited for simulating large networks due to the inadequate current pumping capacity of the opamp [2]. To solve this problem we used class AB current amplifiers after the main and the inverter opamps (Fig. 2). Without the current amplifiers, for large networks (with N > 10 neurons), the opamp outputs did not settle at $\pm V_{cc}$, but dropped to 0 with time. This modified circuit was used in all the simulations for comparison of performance with the adaptive model. The current equation at the positive terminal of the noninverting opamp is,

$$C\frac{du_i}{dt} + \frac{u_i}{R_t} + \sum_{j=1}^{N}\frac{(u_i - \gamma_{ij} V_j)}{R_{ij}} = 0 \qquad (3.1)$$

Where $\gamma_{ij}(\pm 1)$ represents the sign of connection between i'th and j'th neurons. Note that in Fig. 1, the output V_i of the first opamp is passed through an inverter. In the Hopfield model, if the connection from i'th neuron to j'th neuron is negative, the output of the inverter ($-V_i$) is connected to the j'th neuron through a resistor (R_{ij}); if it is positive V_i is directly given as input to the j'th neuron. The last equation can also be written as:

$$C\frac{du_i}{dt} = -u_i\left(\frac{1}{R_t} + \sum_{j=1}^{N}\frac{1}{R_{ij}}\right) + \sum_{j=1}^{N}\frac{\gamma_{ij}v_j}{R_{ij}} \qquad (3.2)$$

Comparing (2.1) and (3.2) we have

$$\tau = \frac{C}{\left(\frac{1}{R_t} + \sum_{j=1}^{N}\frac{1}{R_{ij}}\right)} \text{ and } \quad w_{ij} = \frac{(\gamma_{ij}\frac{1}{R_{ij}})}{\left(\frac{1}{R_t} + \sum_{j=1}^{N}\frac{1}{R_{ij}}\right)} \qquad (3.3)$$

Rearranging (3.3), we have,

$$\frac{1}{R_{ij}} = \frac{|w_{ij}|}{R_t\left(1 - \sum_{j=1}^{N}|w_{ij}|\right)} \qquad (3.4)$$

Therefore, once R_t and C are chosen, and the weight matrix calculated using (2.6), all the circuit parameters can be calculated using 3.4. To sum up, in the HNN model OPAMPS model nonlinear neurons and resistors model the *fixed connections*.

In the circuit implementation of the network with adaptive weights (AHNN), the fixed coupling resistances, R_{ij}'s, of Fig.2 are replaced by voltage-controlled resistances as shown in Fig. 3. The capacitor voltage V_c controls the weight W_{ij} between the i^{th} and j^{th} neuron. Applying KCL at node V_c,

$$C_w\frac{dV_c}{dt} = \frac{V_iV_j - V_c}{R_w} \qquad (3.5)$$

$$R_wC_w\frac{dV_c}{dt} = -V_c + V_iV_j \qquad (3.6)$$

It is apparent that eqn. 3.6 is identical to eqn 2.3. Also, $W_{ij} \propto v_c$ and R_wC_w is the time constant of relaxation for this layer. A four quadrant transconductance multiplier is used to implement the multiplication of local variables V_i and V_j in eqn 3.6, output of which charges the weight control capacitance. So far,using the multiplier, resistor and capacitor we have implemented the learning rule and obtained a voltage across the capacitance which establishes the value of the synaptic connection between the I^{th} and j^{th} neurons respectively. Synaptic weight W_{ij} is a conductance between output of j^{th} neuron and input of i^{th} neuron which varies depending on the learning rule (the Capacitance voltage) .It can be modeled by using a complimentary pair of enhancement mosfets. Let us consider how the capacitor voltage V_c controls the weight w_{ij} of the (i, j) synapse. If §V_j is applied as the drain voltage of the MOS transistor, where § is a small constant, then for $V_{DS} \le V_{GS} - V_T$ the device operates in the linier region of its characteristics. Under this condition for $V_c \rangle V_{th}$, the drain current is given by the equation,

$$I_{DS} = \frac{KW}{L}\left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}\right] \qquad (3.7)$$

If $V_{DS} << V_{GS} - V_T$, $I_{DS} = \frac{KW}{L}\left[(V_{GS} - V_T)V_{DS}\right]$

$$R_{ij} = R_{DS} \cong \frac{L}{KW(V_{GSi} - V_T)} \qquad (3.8)$$

Where K is the transconductance, W and L are channel width and length, respectively, V_T is transistor threshold voltage.

As V_c can take both positive and negative values, depending on output of four quadrant multiplier, a complimentary pair of P and N channel transistors have to be used in order to model both positive and negative weights. Voltage controlled conductance can also be implemented in several other ways using switched capacitor, switched ladder and CMOS switches. But after trying several possibilities we preferred this scheme as, the value of R_{ds} can be scaled by the channel length to width ratio, L/W as apparent from 3.8. The switch normally remains off if $V_{gs} \langle V_{th}$, but then with increasing V_{gs} greater than V_{th} the value of R_{ds} drops quickly. When applying this configuration as a tunable resistance we should however keep in mind that, ideally, linear resistances of values as in equation 3.8 are only obtainable at the origin i.e. $V_{ds} = I_{ds} = 0$. Otherwise slight nonlinearity due to the quadratic term in equation 3.7 is always present, and the weights can only be approximately considered as linier resistances. This should not be a serious problem, however, since no great precision of weight values is usually required for a neural network operating in a recall mode [10].

IV SIMULATION RESULTS

The objective of our simulations is to investigate the consequence of weight adaptations in retrieval phase on the network performance as an associative memory. The immediate goal is not to maximize the performance but to achieve a good understanding of the network dynamics.

Experiment #1 (P/N Ratio): One of the most important performance characteristics of an associative memory is its storage capacity. 100 Neuron electronic models (N=100) of HNN and the AHNN were put on various storage/ retrieval tasks. It is well-known from the theory of HNN that increasing the number of stored patterns (P) beyond a critical value rapidly impairs network performance [2]. The number of stored patterns, P, is varied from 1 to 40 and the retrieval error is calculated by taking an average of 40 trials.

Firstly we define network's performance. In all the simulations, the network's initial state, **V**(0), is a noisy version of a stored pattern; S^p. Noise is added by flipping a

fixed fraction (14%) of bits in S^p. Ideally, for $\mathbf{V}(0)$ close to S^p, the final state, $\mathbf{V}_{final}$, of the network must equal S^p. Since this does not always happen, we define the error in network performance as the root mean square value of $||\mathbf{V}_{final} - S^p||$ over many trials. In the HNN the weight matrix encoding the stored patterns is calculated using (2.6) and is converted into resistances using (3.4). In the case of AHNN the weight matrix is loaded as initial conditions on the capacitance of the weight dynamics and is allowed to evolve in time simultaneously with the activation dynamics during retrieval phase. Typically the circuit is simulated for 130 μsec, by which time it is found that the opamp (Neuron) outputs saturate near $\pm V_{cc}$. Fig 6 shows the variation of the error in retrieval for HNN and AHNN as a function of number of patterns (P) stored simultaneously. Note that increasing the number of stored patterns beyond approx. 14% of N rapidly impairs network performance of HNN.This is a known result from HNN theory. It is also evident from the graph that the adaptive model guarantees a significantly higher P/N ratio (approx. 0.26 as compared to 0.14 of HNN). Note that the deterioration in fruitful retrieval of the stored patterns is also much slower in AHNN as compared to the HNN.

Experiment #2 (Character Recognition): We consider a simple numerical character recognition task. The memory patterns 0 to 9 are stored in the form of 10 X 10 two dimensional images. In the adaptive network, the weight matrix initially has all the stored character patterns encoded in itself. That is, the initial weight matrix is calculated using Hebbian rule [5] and is fed as initial conditions on the capacitances of the learning block, *unlike the Hopfield network in which the weights converted into fixed resistances using eqn 3.4.* The network is initialized from the neighborhood of any of the stored patterns, say S^p. The output image retrieved is reconstructed once the network stabilizes. Figure 7 shows comparative retrieval performance the HNN and AHNN, with all the ten numeric characters stored and noise kept constant at 14 percent (by flipping bits at random positions). It is readily apparent from the figure that adapting weights during retrieval can indeed result in better recognition of the stored characters. Also, weights in the adaptive model finally settle approximately at the weight values separately calculated for the stored pattern S^p only. In some cases we also observe that output patterns appear to be combinations of several patterns (character 4) for the HNN but are correctly identified by AHNN. Sometimes even patterns that are completely inverted were obtained. (This is acceptable as the Lyapunov energy function has identical values for complimentary states).When the initial conditions are given as a mixture of numbers, for example, an initial condition containing, say, 15% of character 1, 15% of character 3 and 70%of character 5 along with 25% bit flipping, the retrieval performance of the adaptive model was found to be much better. Experiments on character recognition using the AHNN along with Kohonen's self organizing maps are also being carried out and for reasons of space those experimental results could not be included in this work.

V. Automatic SPICE Netlist Generation

To test the performance in SPICE, even 50 neuron models had more than forty thousand nodes and it was impossible to use a schematic editor. So a new scheme was devised, in which there is a Visual *C++ program which automatically generates the SPICE netlist.* The user specifies the number of neurons N, No of patterns to be stored and the amount of corruption i.e. noise level. Provision for random generation of patterns to be stored as well as random generation of noise is also made. The weight matrix is calculated and corresponding values of weights are assigned as initial conditions on capacitors. Output of the program is a text file and can be simulated in SPICE as a foreign Netlist.

VI. DISCUSSION

An interesting feature of our work is the idea of adapting weights during retrieval, which goes against the basic tenets of connectionism. The central dictum of connectionism is that the knowledge contained in a neural or connectionist network is encoded in the network weights [1]. But in our case, the weights change, losing information in the process, but ultimately resulting in better performance. In this procedure the weights are used as transitory, intermediary variables. Stability of such systems can be proved by associating a Lyapunov energy function with the network dynamics.

Simulations with the implemented electronic analog demonstrate that by using an appropriate adaptive law for the weights during retrieval, the storage capacity can be enhanced significantly. Analog VLSI implementations of learning neural networks offer improvements over both software simulations and digital circuits emulating analog operation, opening up scope for many interesting experiments. In an earlier work we showed that there exists close, graded relationship between energy consumption (dissipation) and informational work in an electronic implementation of the Hopfield neural network. Contrarily, efforts to minimize energy dissipation led to performance deterioration [1, 2]. Most of the neurodynamical systems turn out to be dissipative [6] and one may wonder whether there are fundamental physical reasons why neural memories need to be dissipative. The possibility of linking computation with energy expenditure also becomes extremely relevant in the light of certain neuroscience data. In the brain there is a tight coupling between 4 quantities – 1) neural activity, 2) local cerebral blood flow, 3) local glucose metabolism, and 4) temperature changes [15]. Studies also show that external sensory stimuli evoke transient, local temperature changes in the brain [16]. Hence it would also be a worthwhile exercise to explore the energetic costs of increase in computational reliability of the adaptive model (AHNN) presented in this article.

VII. Mathematical Appendix

The energy function E in (2.5) is a Lyapunov function of the system (2.1,2.2,2.3).

Proof: Using Chain rule and the property,

$$\frac{d}{dV_i}\left(Gi\int_0^{V_i} g^{-1}(V)dV\right) = G_i u_i$$

and taking the time derivative of E, we have,

$$\frac{dE}{dt} = -\left(\sum_i \frac{dV_i}{dt}\left(\sum_j W_{ij}V_j - G_i u_i\right)\right) - \left(\sum_i\sum_j V_i V_j \frac{dW_{ij}}{dt} - \sum_i\sum_j W_{IJ}\frac{dW_{ij}}{dt}\right)$$

$$= -\left(\sum_i \frac{dV_i}{dt} C \frac{dU_i}{dt}\right) - \left(\sum_i\sum_j \left(-W_{ij} + V_i V_j\right)\frac{dW_{ij}}{dt}\right)$$

$$= -\left(\sum_i \frac{dV_i}{dt} C \frac{dU_i}{dt}\right) - \left(\sum_i\sum_j \left(\frac{dW_{ij}}{dt}\right)^2\right)$$

$$\leq 0$$

as $\frac{dV_i}{dt}$ *and* $\frac{du_i}{dt}$ *have identical signs.*

Therefore the function E in (2.5) is a global Lyapunov function for the system (2.1,2.2,2.3).

VIII. FIGURES

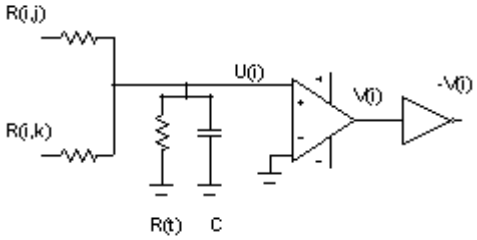

Figure 1. Basic electrical model of Hopfield neuron.

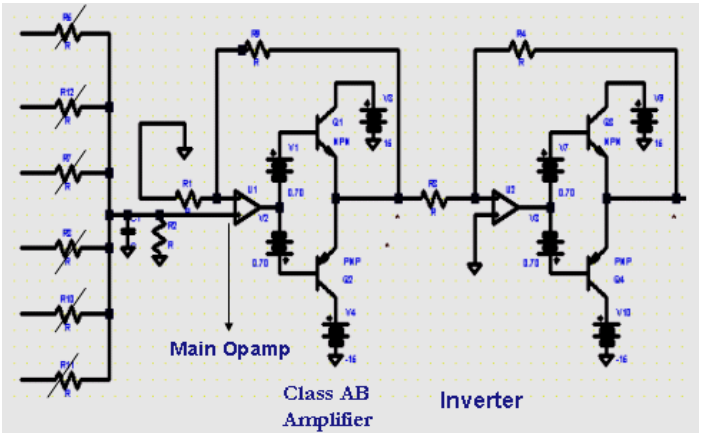

Figure 2: Modified electrical model of the Hopfield neuron in order to simulate large networks.

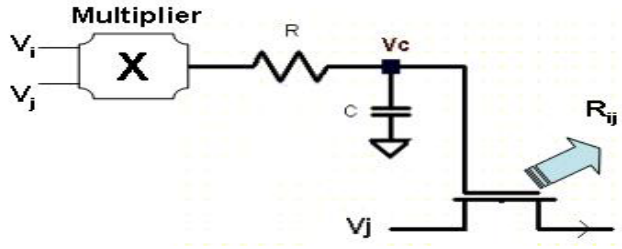

Figure 3: Equivalent circuit of weight adaptation dynamics

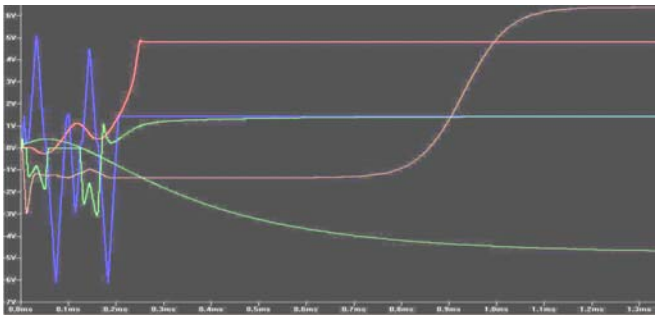

Figure 4: Weights changing and stabilizing slowly during retrieval

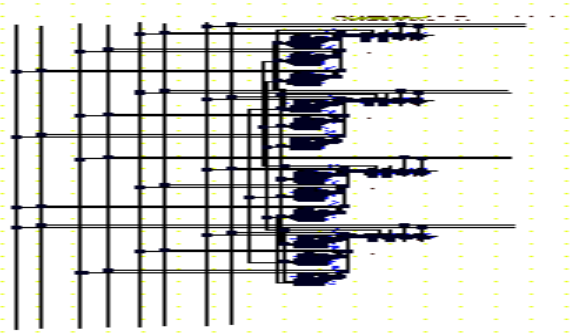

Figure 5: Four Neuron Adaptive model. (50 and 100 neuron models were simulated by generating foreign SPICE netlists using VC++).

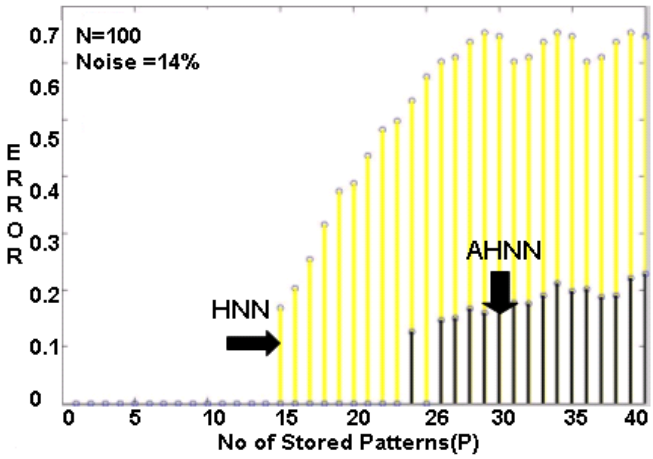

Figure 6: Shows the variation of the relative error in retrieval for 100 neuron HNN and AHNN electronic models as a function of number of patterns (P) stored simultaneously. Note the increase in P/N ratio and graceful degradation of performance in AHNN.

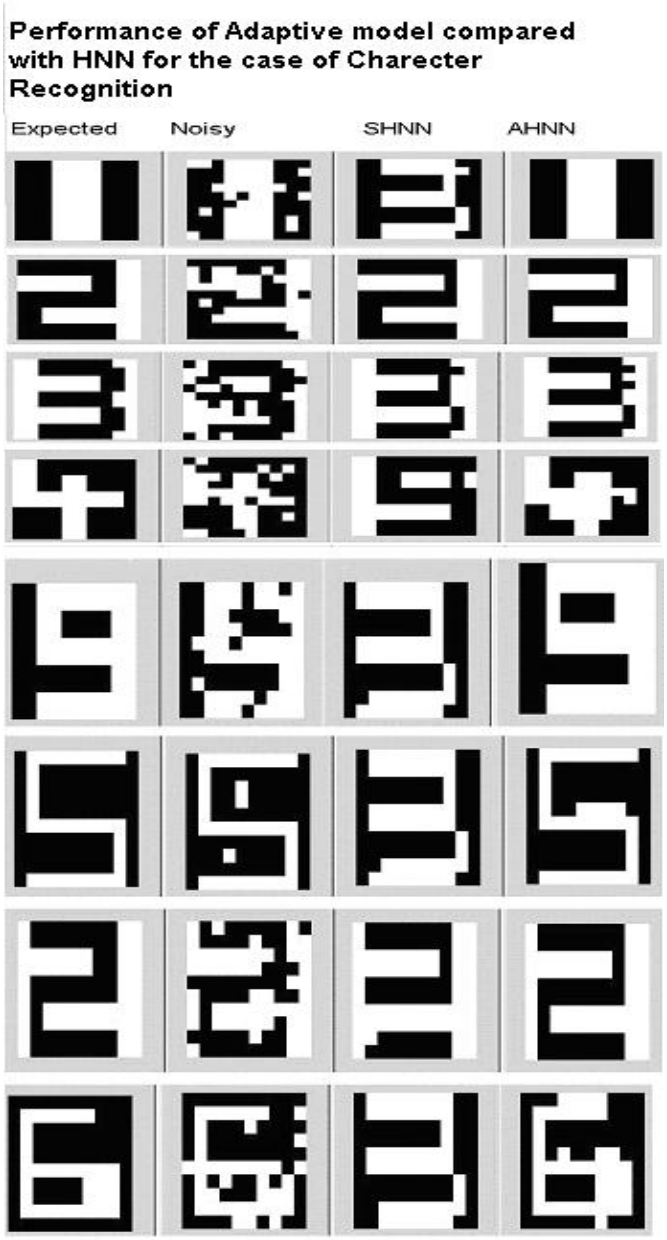

Figure 7. Comparative retrieval performance of the HNN and AHNN, with eight numeric characters stored simultaneously. Output images were reconstructed once the network stabilizes. Average of 40 trials was taken in order to thoroughly verify the recognition ability.

IX. REFERENCES

1. Udayshankar, M., Vishwanathan M., & Chakravarthy V.S., Inevitable Energy Costs of Storage Capacity Enhancement in an Oscillatory Neural Network, Proceedings of 46th IEEE International Midwest Symposium on Circuits and Systems, December, Cairo, Egypt, 2003.
2. Kumar, A., V. Manmohan, M. Udayshankar, M. Vishwanathan, V.S. Chakravarthy. "Link between Energy and Computation in a Physical Model of Hopfield Network," ICONIP-2002, 18-22, November, Singapore, 2002.
3. J.J.Hopfield. Neural networks and physical systems with emergent collective computational abilities, *Proc of the Nat'l Acad of Sci* USA, 79:2554-2558, 1982.
4. M. Cohen & S. Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks, IEEE Trans. Sys. Man Cybernetics. 13:pp815-826, 1983.
5. J.J.Hopfield. Neurons with graded response have collective computational properties like those of two state neurons. *Proc Nat'l Acad Sci* USA 79:2554-2558, 1984.
6. S .Haykin. Neural Networks: A Comprehensive Introduction, Prentice Hall India, 1999.
7. B.Kosko.Feedback stability and unsupervised learning. In Proc.of IEEE Intl.Conf. on Neural Networks: Vol1, 141-152.San Diego: IEEE Press,1988.
8. J.Zurada. Introduction to Artificial Neural Systems, Jaico Publishers, 1994.
9. P.Baldi, S.Venkatesh. The number of stable points for spin glasses and neural networks of higher orders. Physical review letters, 58.913-916, 1987.
10. G.E.Hinton, T.J.Sejnovski, D.H.Ackley. Boltzman Machines: Constraint satisfaction networks that learn.Technicalreport, CMU-CS-84-119, Caringe-Mellon Univ., Pittsburg, PA, 1984.
11. C.A Mead. Analog VLSI and Neural Systems, MA: Addison-Wesley, 1989.
12. E Vittoz. Analog VLSI Signal processing: Why, Where and How? Journal of VLSI Signal processing, Kluwer, 1994.
13. H.P.Graf, L.D.Jackel. Analog electronic neural network circuits, IEEE Circuits and devices magazine, 1989.
14. H.C.Card. Hebbian plasticity in MOS Synapses, IEE proceedings - F, Volume 138,No 1 ,February 1991.
15. L .Sokoloff, *Metabolic Probes of Central Nervous System Activity in Experimental Animals and Man*, Sunderland, MA: Sinauer Associates, 1984.
16. J.G.McElligott and R Melzack, "Localized Thermal Changes evoked in the brain by Visual and Auditory Stimulation," Experimental Neurology, vol. 17, pg 293-312, 1967