
Neural Processing Letters **13:** 183–194, 2001.

Generalizations of the Hamming Associative Memory

PAUL WATTA[1] and MOHAMAD H. HASSOUN[2]
[1]*Department of Electrical & Computer Engineering, University of Michigan-Dearborn, Dearborn, MI 48128; USA; e-mail: watta@umich.edu*
[2]*Department of Electrical & Computer Engineering, Wayne State University, Detroit, MI 48202, USA; e-mail: hassoun@brain.eng.wayne.edu*

Abstract. This Letter reviews four models of associative memory which generalize the operation of the Hamming associative memory: the grounded Hamming memory, the cellular Hamming memory, the decoupled Hamming memory, and the two-level decoupled Hamming memory. These memory models offer high performance and allow for a more practical hardware realization than the Hamming net and other fully interconnected neural net architectures.

Key words: artificial neural network, associative memory, capacity, error correction, Hamming net

1. Introduction

Existing models of associative memory suffer from one or more of the following serious flaws:

- Limited capacity
- Low or unquantifiable error correction capability
- Large number of spurious memories
- Impractical hardware implementation.

In terms of hardware implementation, difficulties arise either from requiring complicated hardware or else requiring an excessive number of interconnection weights (for example, fully interconnected architectures). In this letter, we describe a class of associative memory models which can overcome some of these design flaws. These memory models generalize the operation of the Hamming associative memory by allowing for a ground state and local distance measures. Four different models are discussed:

(1) Grounded Hamming Memory
(2) Cellular Hamming Memory
(3) Decoupled Hamming Memory
(4) Two-level decoupled Hamming Memory.

The grounded Hamming memory is similar to the Hamming associative memory, but allows for a ground state which will attracts all states with very low signal-to-noise ratio. The cellular Hamming memory utilizes local Hamming dis-

tance measures rather than a global Hamming distance measure, leading to a cellular network architecture which is more amenable to VLSI hardware implementation and fine-grained parallel implementations. The decoupled Hamming memory also uses local Hamming distance computations, but requires less hardware than the cellular Hamming memory because the local windows do not overlap. Finally, the two-level decoupled Hamming memory combines the decoupled Hamming distance computations with a higher-level decision making stage.

These associative memory models have been individually described and analyzed in the literature in recent years [1–7]. The purpose of this letter is to give a complete summary and comparison of the various models, highlighting their advantages and limitations in terms of memory capacity and hardware implementation.

In the following, we consider the binary autoassociative memory problem. In this case, the given fundamental memory set is of the form $\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^m\}$, where each pattern $\mathbf{x}^i$ is an N-bit binary vector, i.e. $\mathbf{x}^i \in \{0, 1\}^N$, $i = 1, 2, \ldots, m$. The task is to design a system which associates every fundamental pattern with itself. That is, when presented with $\mathbf{x}^i$ as input, the system should produce $\mathbf{x}^i$ at the output. In addition, when presented with a *noisy* version of $\mathbf{x}^i$ at the input, the system should also produce $\mathbf{x}^i$ at the output.

Let the Hamming distance between two binary vectors $\mathbf{x}$ and $\mathbf{y}$ (of the same dimension) be denoted as $d(\mathbf{x}, \mathbf{y})$.

The Hamming associative memory [2] is a static model, and operates as follows: For any memory key $\mathbf{x} \in \{0, 1\}^N$, the retrieved pattern is obtained by computing the Hamming distances $d_k = d(\mathbf{x}, \mathbf{y}^k)$, selecting the minimum such distance d_{k^*}, and outputting the fundamental memory $\mathbf{x}^{k^*}$ (closest match). The most attractive feature of this model is its exponential capacity [8] and large error correction capability. There are two serious disadvantages of this model. First, there is no provision for a ground state. Second, the hardware implementation is cumbersome because the computation of the required Hamming distances is a global and sequential operation, and hence not immediately suitable to parallel and distributed processing systems. The following sections detail generalizations of the Hamming memory which overcome these limitations.

2. The Grounded Hamming Memory

The operation of the grounded Hamming associative memory model [5] is similar to the Hamming associative memory, but provides for a ground state (the zero state) to attract all outlier or 'garbage' inputs. That is, the grounded Hamming associative memory outputs the closest fundamental memory when there is a sufficiently close match between the memory key and one of the fundamental patterns; otherwise, when no such match occurs, the grounded Hamming memory converges to the ground state. The Hamming memory, on the other hand, has no such ground state, and will produce a fundamental memory at the output no matter how noise-corrupted the memory key. Clearly, there are applications which require such a

'no decision' state in the presence of excessive noise. Figure 1 shows conceptually the difference between the Hamming associative memory and the grounded Hamming memory in terms of state space basins of attraction.

The first step in the design of the grounded Hamming memory is to choose some desired level of *error correction*. The easiest way to do this is to choose a value for p, the number of correctable bit errors. If the number of correctable bit errors is to be uniform for all memories in the fundamental memory set, then p must be chosen such that the following condition is satisfied

$$1 \leqslant p < \tfrac{1}{2}\min\{d(\mathbf{x}^i, \mathbf{x}^j)\}, \tag{1}$$

where $i \neq j \in \{1, \ldots, m\}$. Once p is chosen, the grounded Hamming memory operates as follows. Given an input memory key $\mathbf{x}$, we compute $d(\mathbf{x}, \mathbf{x}^i)$ for each $i = 1, 2, \ldots, m$. If there is an index i^* which satisfies $d(\mathbf{x}, \mathbf{x}^{i^*}) \leqslant p$, then the output of the memory is $\mathbf{x}^{i^*}$; otherwise, the output of the memory is the ground state $\mathbf{0}$. Note that by the condition established in (1), if i^* exits, then it is unique.

In Watta, Wang, and Hassoun [7], it was shown that an explicit Boolean expression for the grounded Hamming memory operation can be derived. In this case, the system may be implemented with digital hardware, as shown schematically in Figure 2. Here, each $\lambda_{\mathbf{x}^i}(\mathbf{x})$ block is a collection of AND and OR gates; see [7] for details. Unfortunately, this design is not practical for high dimensional systems because it requires an excessive number of gates.

A more efficient design of the grounded Hamming memory may be obtained by using linear threshold gates (LTGs). The neural architecture for this system is shown in Figure 3. Here, each of the mLTGs is 'tuned' to respond to a single fundamental memory. In this case, we require at most $m(N+1)$ LTGs, which scales linearly in N. Note the incredible savings in hardware for this LTG-realization of the grounded Hamming memory as opposed to the digital logic realization, which required an exponential number of gates. In this sense, and as mentioned in [9] in a different context, the LTG is *exponentially more powerful* than digital logic gates.

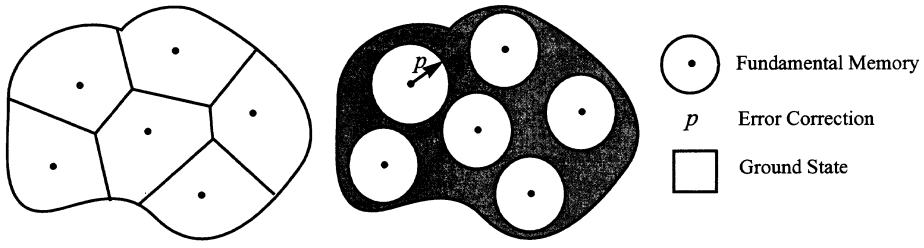

Figure 1. Basins of attraction of the (a) the Hamming memory and (b) the grounded Hamming memory.

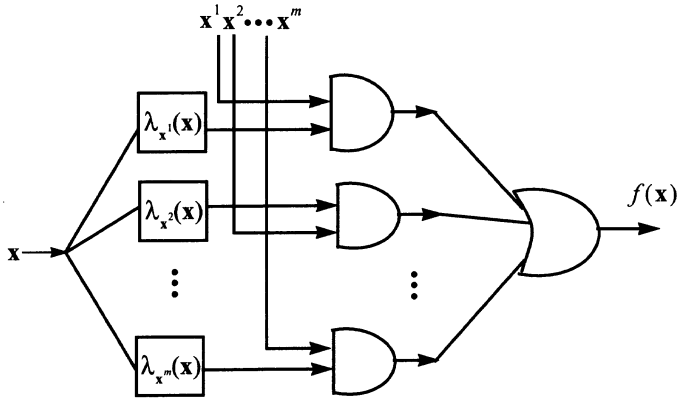

Figure 2. Circuit diagram of the grounded Hamming memory. Each $\lambda_{\mathbf{x}^i}(\mathbf{x})$ is a block of AND and OR gates.

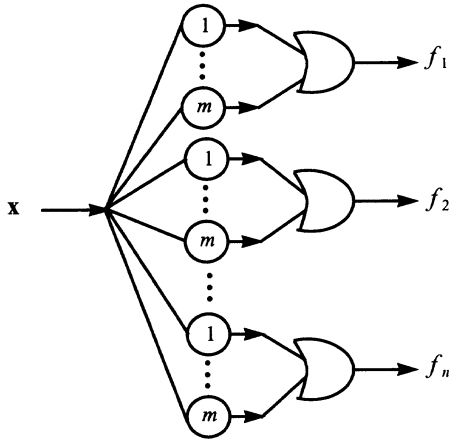

Figure 3. LTG-realization of the grounded Hamming memory.

Another attractive feature of the architecture shown in Figure 3 is the ease with which the size of the basin of attraction for all fundamental memories can be changed. Here, to change p, we need only adjust the threshold of each neuron in the hidden layer. In fact, in this case, we don't require a uniform value of p for all memories: we can define a nonuniform error correction for each fundamental memory, $p_j, j = 1, 2, \ldots, m$, where

$$1 \leqslant p_j < \tfrac{1}{2}\min\{d(\mathbf{x}^j, \mathbf{x}^k): k = 1, \ldots, m, k \neq j\}. \tag{1}$$

The state space structure for the grounded Hamming memory with nonuniform error correction is shown schematically in Figure 4. Note that the grounded Hamming memory with uniform error correction uses the most conservative value of p; i.e. $p = \min\{p_j : k = 1, \ldots, m\}$. The nonuniform grounded Hamming memory, though, is able to handle (for some images) much more noise.

3. Cellular Hamming Memory

Notice that for the grounded Hamming net formulated above, each output bit is a function of the entire input vector; i.e. $y_i = y_i(x_1, x_2, \ldots, x_N)$ for each $i = 1, 2, \ldots N$, Substantial savings in hardware may be achieved by restricting the dependence of each output to a small fraction of all possible inputs. giving rise to the notion of a *local Hamming distance* measures. Since our application will be for image processing. we will suppose that the memory input pattern is 2-dimensional. In this case, the cellular Hamming model is actually a two-dimensional *nonuniform* cellular automaton in which neighboring pixels interact locally, as shown in Figure 5(a). Here, the pixel interconnectivity structure is shown as a 2×2 Von Neumann-type neighborhood, but larger neighborhoods, such as the 3×3 Moore neighborhood. or extended Moore neighborhoods may be formed, as well.

Previous studies of 2-dimensional cellular automata [10] concentrated on uniform systems in which each automaton (pixel element) employs the same transition or updating function. Here, to obtain the operation of associative recall, each automaton may employ a different transition function, formulated as follows. First, each pixel senses the state of the pixels in its neighborhood. For example, suppose the pixel marked with a circle in Figure 5(b) is chosen for updating and suppose we employ a 3×3 neighborhood (shown as the shaded region surrounding the updating pixel). The updating pixel updates itself as follows:

1. Compute the Hamming distance between its neighborhood configuration and the corresponding pixels in all the memory patterns. For example, in Figure 5(b), the distances $\{d_1, d_2, \ldots, d_m\}$ are computed.

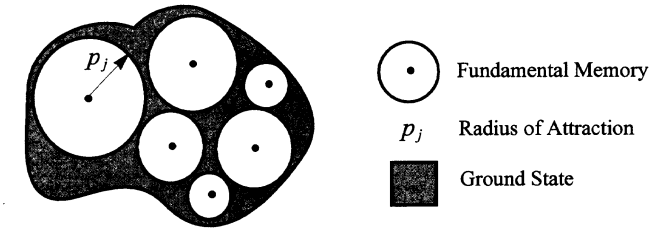

Figure 4. The grounded Hamming memory with nonuniform error correction.

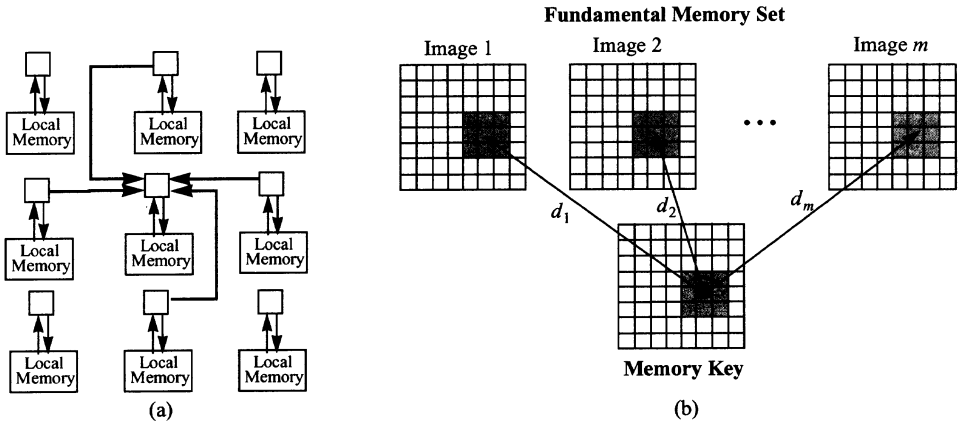

Figure 5. (a) Structure of local interactions for the local Hamming net, assuming a 2×2 neighborhood; (b) Updating of the memory key by local Hamming distance computations (here, a 3×3 neighborhood is used).

2. Choose the smallest such distance d_{k^*}.
3. Assume the value of the center pixel in that closest pixel pattern.

A simple heuristic may be employed in case of a tie in the minimum Hamming distance computation. For example, if there is a tie between two images and if the two fundamental memories agree at pixel (i, j), then we simply assign the pixel to the common value. If, however, they disagree at (i, j), then heuristically, we can keep the input image pixel at its present value. Similar rules can be formulated in the case of a tie among several memory patterns.

There are several ways in which the dynamics can be computed. For *parallel update,* all the nodes in the network update their state at each time instant. For *sequential update,* only one node is updated at each time instant following some fixed ordering of the nodes. *Block sequential update* is a mixture of sequential and parallel update in which a partition is formed on the set of nodes, and the updating of the partitions follows sequentially, while the updating of the nodes within each partition block is performed in parallel. For *random update,* only one randomly chosen node is updated at each time instant.

It is important to note that the full Hamming net can be viewed as a special case of this memory, corresponding to the case where each pixel has a neighborhood consisting of the entire image. We expect the quality of memory retrievals will be best for larger neighborhoods, approaching the (optimal) performance of the Hamming net in the limit of maximum neighborhoods.

4. Decoupled Hamming Associative Memory

The local Hamming memory uses overlapping windows and hence requires a lot of hardware. It is possible, though, to use nonoverlapping windows, giving rise to the *decoupled Hamming associative memory*.

The decoupled Hamming associative memory localizes the Hamming distance computation by partitioning the input vector into nonoverlapping modules or windows, and performing the Hamming memory operation on each module independently. To be precise, suppose we partition the N input variables $X = \{x_1, x_2 \ldots, x_N\}$ of our memory into w modules: $\{X_1, X_2, \ldots, X_w\}$ such that $X_i \subset X, \cup X_i = X$, and $X_i \cap X_j = \emptyset, i \neq j$. To simplify notation, assume that each module has the same number of variables, denoted n. In this case, we have $|X_i| = n, i = 1, 2, \ldots, w$, where $w = N/n$ is the total number of windows or modules. Figure 6 shows the structural difference between (a) the full Hamming memory; and (b) the decoupled Hamming memory.

Each module is a local Hamming memory and has its own local memory set, which is obtained by partitioning each fundamental memory $\mathbf{x}^i$ into w memory subvectors: $\mathbf{x}^k = [\mathbf{x}^k_{(1)}, \ldots \mathbf{x}^k_{(w)}]$, where the ith subvector $\mathbf{x}^k_{(i)} \in \{0, 1\}^n$ contains the components $\mathbf{x}^k$ specified by the variables in the ith module X_i. In this case, we can associate with each module its own local memory set of the form $\mathbf{x}_{(i)} = \{\mathbf{x}^1_{(i)}, \ldots \mathbf{x}^m_{(i)}\}$.

The decoupled Hamming memory operates as follows: The memory key $\mathbf{x}$ is partitioned in the same fashion as the fundamental memories: $\mathbf{x} = [\mathbf{x}_{(1)}, \ldots \mathbf{x}_{(w)}]$, and the w module Hamming memories independently (and in parallel) operate on each of the subvectors of $\mathbf{x}$, computing the Hamming distances $d(\mathbf{x}_{(i)}, \mathbf{x}^k_{(i)}), k = 1, 2, \ldots, m$, and outputting the closest matching pattern.

In the case of 2-dimensional patterns, there are many different topologies possible for the layout of the local Hamming memories. For example, the local Hamming memories may be arranged by row, by column, or in a checkerboard arrangement, as shown in Figure 7(a). Here, the 64×64 binary image is covered with nonoverlapping 16×16 windows in a checkerboard-type layout. Each local Hamming memory then computes 256-bit Hamming distances as opposed to 4096-bit Hamming distances for the entire image.

One clear advantage of the decoupled Hamming memory over the full Hamming memory is retrieval speed. Since all modules can perform their computations in parallel, a w-fold speedup in retrieval time can be achieved by dedicating a processor to each module. A disadvantage of this stringent parallelism, though, is that the decoupled Hamming memory may retrieve a pattern which was not part of the memory set; i.e. spurious memories are possible. For image processing applications, for example, the decoupled memory may converge to an image which is predominantly one of the fundamental images, but contains scattered 'chunks' of other images, as shown in Figure 7(b). The full Hamming network, on the other hand, never retrieves spurious memories.

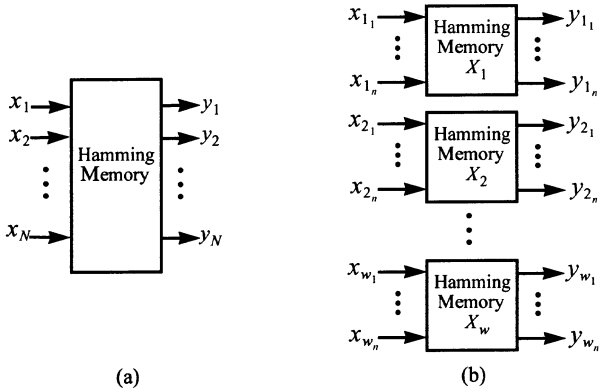

Figure 6. Structure of (a) the full Hamming; and (b) the decoupled Hamming memory.

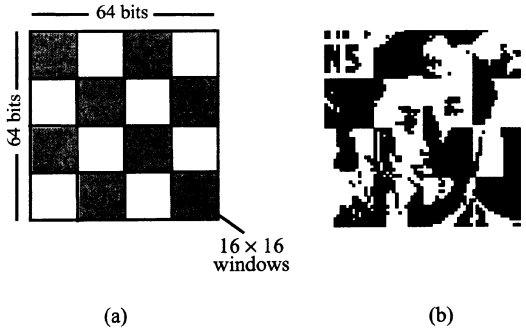

Figure 7. (a) Structure of local memories arranged as an array of nonoverlapping 16 × 16 windows; (b) A spurious memory.

5. The Two-Level Decoupled Hamming Associative Memory

To avoid the spurious memory problem of the previous section, a two-level structure can be used which consists of a decoupled Hamming memory along with a higher-level decision network. The architecture of this memory (in the case of 2-dimensional memory patterns) is shown in Figure 8(a). Here, each local Hamming memory or module X_i computes the closest matching pattern and sends the index I_i of the best match pattern to the decision network. The decision network examines the indices $I_1, I_2, \ldots, I_w$, of all the modules and computes a single best match index I^*. Each memory module then outputs its portion of the fundamental memory

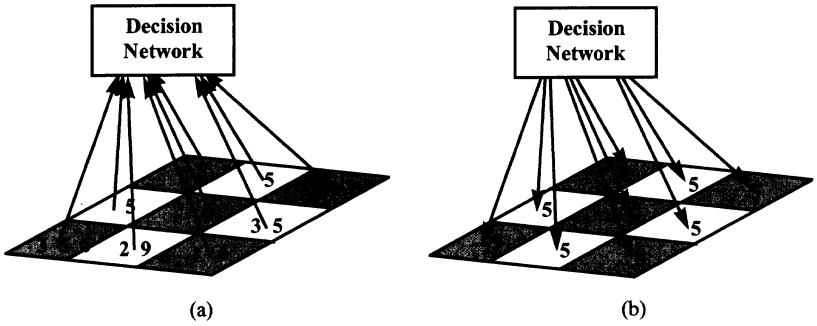

Figure 8. Structure of the two-level decoupled Hamming network. Numbers in (a) represent the index of the closest matching pattern(s); and (b) shows the result after the voting.

$\mathbf{x}^{I^*}$; that is, each module outputs $\mathbf{x}^{I^*}_{(i)}$, $i = 1, 2, \ldots, w$. Since the decision network forces all modules to output the same fundamental memory, the spurious memory problem of the previous section is eliminated.

For example, in Figure 8(a), the window in the upper left hand corner of the image best matches image 5 in the memory set, while the window in the lower right hand corner best matches images 1 and 6 (there is a tie in the Hamming distance). The decision network examines all the votes from the local windows, determines that 5 is the most prevalent, and forces all windows to output its portion of image 5, as shown in (b).

There are many ways to design the decision network. In the simplest case, a majority rule is used, in which I^* is chosen to be the most frequent index among $I_1, I_2, \ldots, I_w$. Utilizing the emerging theory of *classifier combination* [11] and *sensor fusion* [12], more sophisticated decision rules can be formulated. In this case, it may be desirable for each module to send an ordered list of, say, the best 3 indices $I_{i_1}, I_{i_2}, I_{i_3}$, to the decision network. For very noisy patterns, the second and third choices of each module may contain useful information which can be exploited with an appropriate combination scheme.

As with the single layer decoupled Hamming network, it is easy to see that the 2-level decoupled Hamming network reduces to the full Hamming network in the case of a single module: $w = 1$. But unlike the single layer model, the 2-level decoupled Hamming memory also reduces to the full Hamming network in the other extreme case: $w = N$. So the 2-level decoupled Hamming network achieves the optimal performance of the Hamming memory for both the maximum and minimum number of modules.

For intermediate window sizes, the capacity of the two-level decoupled Hamming memory is not as large as the full Hamming memory. But even so, the two-level decoupled Hamming memory with intermediate window size has a much higher capacity and much more error correction than most of the standard neural-based

associative memories, such as the correlation recorded Hopfield network [13], and other recording algorithms for the same single-layer Hopfield-type neural structure.

Besides its performance advantages over standard neural net models, the two-level decoupled Hamming net is ideal for parallel hardware implementation. Since the first level is modularized, the computation can be done in parallel. Indeed, special purpose hardware consisting of a dense array of digital signal processors already exists which can perform the required computations efficiently (see, for example, [14]).

6. Summary

In a template matching (Hamming network) approach to pattern recognition, the input image is compared (using a suitable metric) to each of the prototype images, and the output is simply the best match image. The sequential computation of comparing the input to each prototype is reminiscent of the serial processing that humans do when asked to identify people whom they do not know [15]. In a comparison between a feature based approach and a template matching approach for human face recognition problems, Poggio and Brunelei [16] found that template matching provided superior performance, and in fact, gave 100% recognition on their data set of face images.

The training time for template matching is O(1) complexity, and simply consists of storing all the images in memory. In other algorithms, such as neural net-based approaches, the training time is extremely long, but the retrieval time can be very quick.

An obvious advantage of template-based schemes is that it is easy to add new individuals by simply storing additional images or delete an individual by deleting the undesired prototype images. Another advantage is that template-based systems are not limited to producing just a single output, but can produce an ordered list containing the best matching, say k, individuals.

The main disadvantages of template-based methods were pointed out in Duda and Hart [17]: '. . . the complete set of samples must be stored, and must be searched each time a new feature vector is to be classified.'

Back in 1973 when this classic text was published, memory and processing speed were indeed serious constraints. In fact, it would take almost 10 years before PCs were developed with a clock speeds of a few MHz, 1 MB of RAM, and a few MB of hard disk space. Today, however, affordable PCs are available with clock speeds of 700 MHz, 384 MB of RAM, and 50 GB of hard disk storage. Clearly, the severity of the storage/speed dilemma has diminished in recent years with the availability these powerful and low cost computers, and hence the template matching approach of the Hamming network warrants further consideration.

As outlined in this paper, by combining the Hamming net computation with the parallel and distributed processing methods of neural networks, high performance

associative memories can be designed. In future work, we will apply these generalized Hamming models to the practical problem of human face recognition.

Acknowledgement

This work was supported by the National Science Foundation (NSF) under contract ECS-9618597.

References

1. Artiklar, M., Hassoun, M. H. and Watta, P.: Application of a postprocessing algorithm for improved human face recognition, *Proceedings of the IEEE International Conference on Neural Networks*, IJCN'99, July 10–16, 1999, Washington (1999).
2. Hassoun, M. H. and Watta, P. B.: The Hamming associative memory and its relation to the exponential capacity DAM *Proceedings of the IEEE International Conference on Neural Networks*, ICNN'96, June 3–6, Washington, D.C. (1996), pp. 583–587.
3. Ikeda, N., Watta, P. and Hassoun, M. H.: Capacity analysis of the two-level decoupled Hamming associative memory, *Proceedings of the International Joint Conference on Neural Networks*, IJCNN'98, May 4–9,1998, Anchorage, Alaska (1998), pp. 486–491.
4. Watta, P. B., Akkal, M. and Hassoun, M. H.: Decoupled voting Hamming associative memory networks, *Proceedings of the IEEE International Conference on Neural Networks,* ICNN'97, June 9–12, 1997, Houston, Texas (1997), pp. 1118–1193.
5. Watta, P. B. and Hassoun, M. H.: Efficient realization of the grounded Hamming associative memory, *Proceedings of the World Congress on Neural Networks,* WCNN'96, Sept. 15–20, San Diego, CA (1996), 763–767.
6. Watta, P. B., Ikeda, N., Artiklar, M., Subramanian, A. and Hassoun, M. H.: Comparison between theory and simulation for the two-level decoupled Hamming associative memory, *Proceedings of the IEEE International Conference on Neural Networks,* IJCN'99, July 10–16, 1999, Washington (1999).
7. Watta, P., Wang, K. and Hassoun, M. H.: Recurrent neural nets as dynamical Boolean systems with application to associative memory, *IEEE Transactions on Neural Networks* **8**(6) (1997), 1268–1280.
8. Chou, P.: The capacity of the Kanerva associative memory, *IEEE Transactions on Information Theory* **35**(2) (1989), 281–298.
9. Siu, K., Roychowdhury, V. and Kailath, T.: *Discrete Neural Computation: A Theoretical Foundation,* Prentice-Hall, Englewood Cliffs, New Jersey (1995).
10. Wolfram, S.: *Cellular Automata and Complexity: Collected Papers*, Addison-Wesley, Reading, Massachusetts (1994).
11. Bishop, C.: *Neural Networks for Pattern Recognition*, Oxford University Press, New York (1995).
12 Ho, T., Hull, J. and Srihari, S.: Decision combination in multiple classifier system, *IEEE Transactions on Neural Networks* **16**(1) (1994), 66–75.
13. Hopfield, J.: Neurons with graded response have collective computational properties like those of two-state neurons, *Proc. Natl. Acad. Sci., USA, Biophys.* **81** (1984), 3088–3092.
14. Pulkki, V. and Taneli, H.: An implementation of the self-organizing map on the CNAPS neurocomputer, *Proceedings of the IEEE International Conference on Neural Networks*, ICNN'96, June 3–6, Washington, D.C. (1996), pp. 1345–1349.
15. Chellappa, R., Wilson, C. and Sirohey, S.: Human and machine recognition of faces: A survey, *Proceedings of the IEEE* **83**(5) (1995), 705–740.

16. Brunelli, R. and Poggio, T.: Face recognition: Features vs. templates, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(10) (1993), 1042–1053.
17. Duda, R. and Hart, P.: *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York (1973).