A Two-Level Hamming Network for High Performance Associative Memory

Nobuhiko Ikeda

Department of Computer Science & Electronic Engineering
Tokuyama College of Technology
Yamaguchi, 745 Japan
n-ikeda@tokuyama.ac.jp

Paul Watta

Department of Electrical & Computer Engineering
University of Michigan-Dearborn
Dearborn, MI 48128
watta@umich.edu

Metin Artiklar and Mohamad H. Hassoun

Department of Electrical & Computer Engineering
Wayne State University
Detroit, MI 48202
Hassoun@brain.eng.wayne.edu

Abstract

This paper presents an analysis of a two-level decoupled Hamming network, which is a high performance discrete-time/discrete-state associative memory model. The two-level Hamming memory generalizes the Hamming memory by providing for local Hamming distance computations in the first level and a voting mechanism in the second level. In this paper, we study the effect of system dimension, window size, and noise on the capacity and error correction capability of the two-level Hamming memory. Simulation results are given for both random images and human face images.

1. Introduction

The design of high performance associative memory has long been a goal of artificial neural net researchers (Hassoun 1993). A practical low-cost device which could store and retrieve information similar to the human “content addressable” memory would be revolutionary and would have numerous applications in such areas as image processing, face, character and voice recognition, data bases, control systems, and robotics.

Unfortunately, although many neural models have been proposed, there is still today no practical associative memory available as an off-the-shelf item. Existing models of associative memory suffer from one or more of the following serious design flaws: limited capacity, low or unquantifiable error correction capability, a large number of spurious memories, and impractical hardware implementation.

The problem of limited capacity is self-explanatory: The purpose of an associative memory (or any memory for that matter) is to store information. So, no matter how mathematically elegant, an associative memory which can’t store much information is practically useless. The problem of unquantifiable error correction is also very serious, but is usually overlooked. In terms of the hardware implementation problem, difficulties arise either from requiring complicated hardware or else requiring an excessive number of interconnection weights (for example, fully interconnected architectures).

Presently, for the case of discrete-time/discrete-state systems, the associative memory model which comes closest to meeting all essential design criteria is conceptually the simplest: the Hamming associative memory. But although the Hamming associative memory has huge capacity (exponential in the input dimension), offers precise bounds on error correction capability, and has no spurious memories, it suffers from impractical hardware implementation and slow retrieval speed.

Recently, several models of associative memory which generalize the operation of the Hamming associative memory have been introduced: the *grounded Hamming memory* (Watta, Wang, and Hassoun, 1997), the *cellular Hamming memory* (Watta, Hassoun, and Akkal, 1997), the *decoupled Hamming memory*, and the *two-level decoupled Hamming memory* (Ikeda, Watta, and Hassoun, 1998). These models retain the high performance properties of the Hamming net, but allow for a much more practical hardware implementation and faster retrievals by utilizing local Hamming distance measures.

This paper presents an analysis of the most promising of these generalized memories: the two-level decoupled Hamming memory. We derive the capacity of this model as a function of system dimension, local window size, and incident noise. The results indicate that the two-level network can tolerate large amounts of uniform random noise.

The remainder of this paper is organized as follows. Section 2 reviews the operation of the Hamming associative memory. In Section 3, the Hamming memory is mapped onto a parallel and distributed processing (PDP) framework involving local Hamming distance computations. This direct PDP implementation is called the decoupled Hamming memory, but suffers from a serious spurious memory problem. Section 4 discusses how this spurious memory problem can be eliminated by introducing a voting mechanism and a two-level network architecture. Section 5 presents a theoretical analysis of the capacity and error correction of the two-level decoupled Hamming memory for memory sets consisting of uniform random patterns. In Section 6, simulation results on random memory patterns are given which correlate well with the theoretical predictions. Section 7 gives simulation results on a more practical memory set consisting of human face images. Finally, Section 8 summarizes the results and discusses future extensions of this work.

2. The Hamming Associative Memory

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* (corrupted) 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 design phase of the Hamming memory involves simply storing all the patterns of the fundamental memory set. In the recall phase, for a given input memory key $\mathbf{x} \in \{0, 1\}^N$, the retrieved pattern is obtained as follows

(1) Compute the Hamming distances $d_k = d(\mathbf{x}, \mathbf{x}^k)$, $k = 1, 2, \ldots, m$

(2) Select the minimum such distance $d_{k^*} = \min\{d_1, d_2, \ldots, d_m\}$

(3) Output the fundamental memory $\mathbf{y} = \mathbf{x}^{k^*}$ (closest match)

It can be shown that the Hamming memory has exponential capacity and very large error correction capability (Chou, 1989; Hassoun and Watta, 1996). In fact, the performance of the Hamming memory can be shown to be optimal in the sense of classical statistical pattern recognition (Hamming, 1986).

This important fact—that no other memory can outperform the Hamming memory—indicates that the best place to start in formulating a high performance associative memory is with the Hamming memory itself. Unfortunately, there are several serious disadvantages of this model. First, the memory retrievals are slow because the memory key has to be compared to each fundamental memory. Second, each bit of the output pattern depends on each and every input bit. Hence, a direct hardware implementation of the Hamming memory would require full interconnectivity, and hence is impractical.

In the following sections, we develop and analyze an associative memory model which localizes the Hamming distance computations, making it suitable for implementation on parallel and distributed processing systems.

3. The Decoupeld Hamming Memory

Notice that for the 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, resulting in the computation of *local Hamming distance* measures.

The *decoupled Hamming associative memory* localizes the Hamming distance computation by partitioning the input vector into nonoverlapping modules or local 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 local windows: $\{X_1, X_2, \ldots, X_w\}$ 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 local window 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 local windows. Figure 1 shows the structural difference between (a) the full Hamming memory and (b) the decoupled Hamming memory.

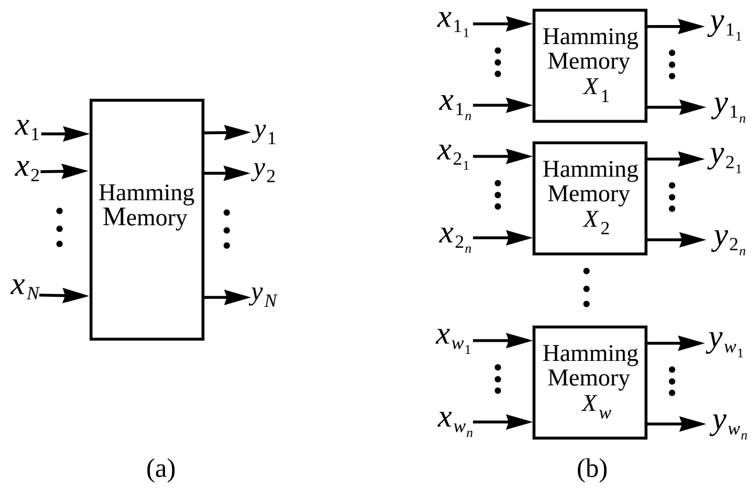


Figure 1. Structure of (a) the full Hamming and (b) the decoupled Hamming memory.

Each window is a local Hamming memory and has its own local memory set, which is obtained by partitioning each fundamental memory $\mathbf{x}^k$ 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 of $\mathbf{x}^k$ 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 2(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.

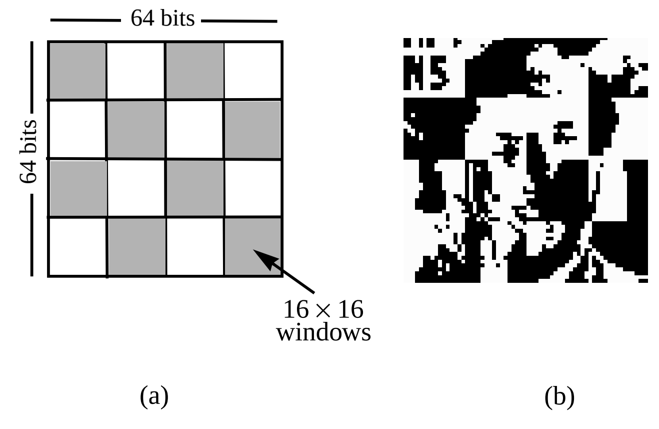


Figure 2. (a) Structure of local memories arranged as a grid of nonoverlapping 16×16 windows in a 64×64 image. (b) A spurious memory.

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 2(b). The full Hamming network, on the other hand, never retrieves spurious memories.

4. 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 3(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 $\mathbf{x}^{I^*}$; 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.

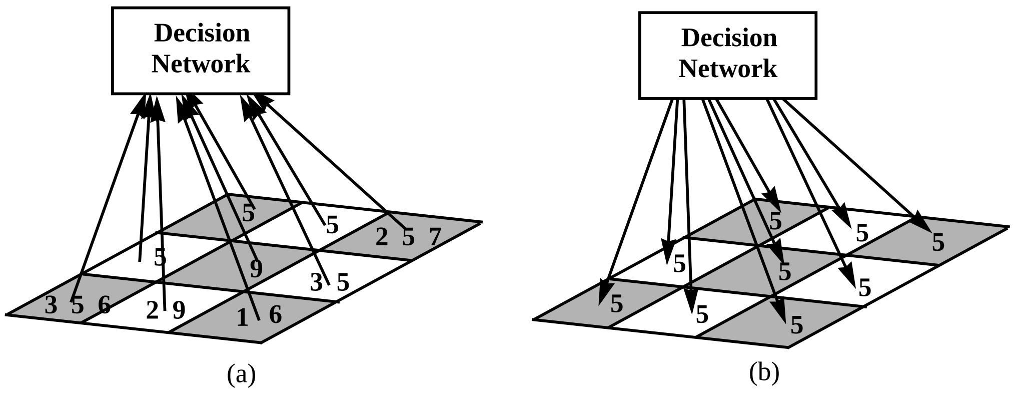


Figure 3. Structure of the two-level decoupled Hamming network. Numbers in (a) represent the index of the closest matching pattern(s), and (b) shows the result after the voting.

For example, in Figure 3(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* (Bishop, 1995) and *sensor fusion (*Ho, et al. 1994), 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. In the analysis which follows, we assume that the decision network uses a simple majority rule.

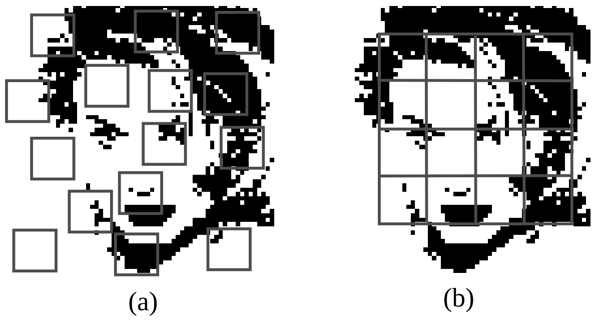


Figure 4. Selective coverage of input image with (a) random placement and (b) strategic placement of modules.

Note that the two-level decoupled Hamming memory can still function even if the entire image is not covered by local windows. For example, instead of covering the image completely as shown in Figure 2(a), we can cover, say $2/3$ of the image as shown in Figure 4(a). Here, the local memories are randomly scattered over the entire image. In the case of human face recognition, a priori information can be exploited to place the covering modules for optimal performance. For example, Figure 4(b) shows a partial covering of the image in the expected nose and eyes region (center of image).

Assuming a majority rule for the decision network, and as with the single layer decoupled Hamming network, it is easy to see that the two-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 two-level decoupled Hamming memory with majority rule decision network also reduces to the full Hamming network in the other extreme case: $w = N$. In this case, each pixel (module) of the image chooses those memory set images which have the same pixel value. The memory set image closest to the test image (in a Hamming distance sense) will necessarily get the most votes among all the modules, and hence the output of the two-level network will be the same as that of the Hamming network.

The most important performance measure of an associative memory is its capacity, or how many patterns it can reliably store (Hertz, Krough, and Palmer, 1991). Typically, statistical methods are used to derive capacity measures (Willshaw, Buneman, and Longuet-Higgins, 1969; Palm, 1980; Buckingham and Willshaw, 1992; Kawamura and Hirai, 1997).

The two-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 (Hopfield, 1982), and other recording algorithms for the same single-layer Hopfield-type neural structure (Hassoun, 1993, 1995).

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, Pulkki and Taneli, (1996)].

5. Capacity Analysis of the Two-Level Decoupled Hamming Memory

In this section, we derive the capacity of the two-level decoupled Hamming memory. Figure 5 shows a schematic of the fundamental memory set and an input memory key. In this case, the memory patterns are shown as 2-dimensional images; however, in our analysis, we still assume the memory elements consist of (one-dimensional) column vectors.

In the previous section, we discussed how the memory key and each of the fundamental memories are modularized into n-dimensional windows. One such n-window is highlighted in the memory key shown in Figure 5. The corresponding n-window is highlighted in each of the fundamental memories, as well.

We assume that the memory set consists of uniformly random vectors. That is, each bit of a fundamental memory has a 50 percent chance of being 1 and 50 percent chance of being 0. In addition, it is assumed that the memory key is a corrupted version of one of the fundamental memories. In particular, the memory key is obtained by adding an amount ρ of uniform random noise to one of the fundamental memories—called the *target memory*; i.e., with probability ρ, each bit of the target image is flipped from its original value. Each of the remaining $m - 1$ fundamental memories will be called a *non-target* memory, or *other* memory (short for “other than the target”).

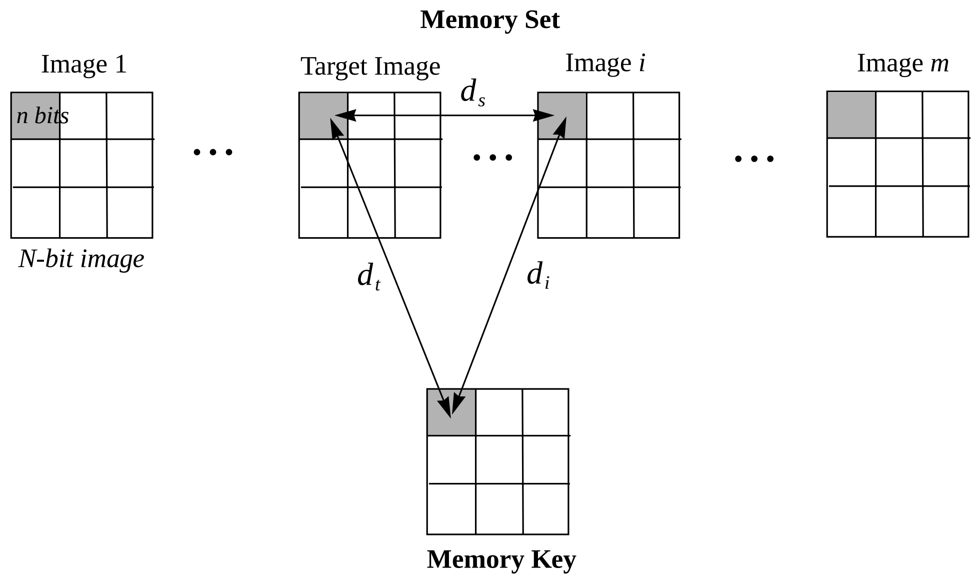


Figure 5. The fundamental memory set and memory key for the 2-level decoupled Hamming memory. With respect to the highlighted *n*-windows, the Hamming distance between the memory key and the target window is denoted d_t, the Hamming distance between the memory key and the *i*th non-target memory window in the memory set is denoted d_i, and the Hamming distance between the target and a non-target memory window is denoted d_s.

The analysis will proceed by first computing the probability that the given local window votes for the target memory and the non-target memories, then the number of votes for the target memory and non-target memories will be computed, and finally, the capacity will be estimated by computing the probability that the target memory gets the highest number of votes.

5.1 Probability of Voting for the Target and a non-Target Memory

In reference to Figure 5, let us fix a window in the memory key (say, the highlighted window), and let us fix the corresponding window in each of the fundamental memories. In addition, of the $m - 1$ non-target memories, let us focus our attention on a single one of them, say the *i*th memory.

Let d_t denote the Hamming distance between the highlighted local window of the memory key and the corresponding window of the target memory, let d_i denote the Hamming distance between the highlighted local window of the memory key and the corresponding window of the *i*th (non-target) image in the memory set, and let d_s denote the Hamming distance between the highlighted local window of the target image and the corresponding window of *i*th image.

Clearly, since each of the fundamental memories is a uniform random binary vector, then the probability that $d_s = j$ bits (where $0 \le j \le n$) follows a binomial distribution of the form

$$Prob(d_s = j) = \binom{n}{j}\left[\frac{1}{2}\right]^j\left[\frac{1}{2}\right]^{n-j} = \binom{n}{j}\left[\frac{1}{2}\right]^n \qquad (1)$$

where $\binom{n}{j}$ is the number of combinations of *n* items chosen *j* at a time, and is given by

$$\binom{n}{j} = \frac{n!}{j!(n-j)!}$$

Also, since the memory key is obtained by uniformly perturbing the target image with an amount of noise ρ, then the probability that $d_t = k$ bits ($0 \le k \le n$) also follows a binomial distribution of the form

$$Prob(d_t = k) = \binom{n}{k}\rho^k(1-\rho)^{n-k} \qquad (2)$$

Another quantity that will be of interest is $Prob(d_i = j | d_t = k)$. That is, assuming $d_t = k$ bits, we want to compute the probability that $d_i = j$ bits. The target image and the *i*th memory image are created completely independently of each other. If some of the bits of the target image are subsequently flipped (which is how the memory key is created), it is still independent of the *i*th memory image; hence $Prob(d_i = j | d_t = k)$ has the same binomial distribution as $Prob(d_i = j)$:

$$Prob(d_i = j | d_t = k) = Prob(d_i = j) = \frac{\binom{n}{j}}{2^n} = \binom{n}{j}\left[\frac{1}{2}\right]^n \qquad (3)$$

Note that each of the other non-target memories in the memory set follows the same distribution, since each of the fundamental memories was created independent and uniformly random. Hence, Equation 3 holds for each non-target memory *i* in the memory set.

Now for which memory will the highlighted window vote? Well, for each $k \in \{0, 1, \ldots, n\}$ the target memory gets a vote if $d_t = k$ **and** $d_{i'} \geq k$ for *all* other $m - 1$ memories (here, i' ranges over all indices $1, \ldots, m$ excluding the index of the target image). The probability that this occurs is given by

$$P_t = P_t(n, \rho, m) = \sum_{k=0}^{n} Prob(d_t = k)\left[\sum_{j=k}^{n} Prob(d_i = j | d_t = k)\right]^{m-1}$$

Substituting Equations 2 and 3 into the above Equation, we get

$$P_t(n, \rho, m) = \sum_{k=0}^{n} \binom{n}{k} \rho^k (1-\rho)^{n-k} \left(\sum_{j=k}^{n} \binom{n}{j} \left[\frac{1}{2}\right]^n\right)^{m-1}$$

On the other hand, for each $k \in \{0, 1, \ldots, n\}$ the *i*th (non-target) memory gets a vote if $d_t = k$ **and** $d_i \leq k$ **and** $d_{i'} \geq d_i$ for all other $m - 2$ memories in the memory set (here, i' ranges over all indices $1, \ldots, m$ excluding index *i*—for the *i*th memory—and the index of the target memory). Hence, the probability that the *i*th image gets a vote $P_i = P_i(n, \rho, m)$ is given by

$$P_i = \sum_{k=0}^{n} Prob(d_t = k)\left\{\sum_{j=0}^{k} Prob(d_i = j | d_t = k)\left[\sum_{s=j}^{n} Prob(d_i = s | d_t = k)\right]^{m-2}\right\} \tag{4}$$

Substituting Equations 2 and 3 into the above Equation, we get

$$P_i(n, \rho, m) = \sum_{k=0}^{n} \binom{n}{k} \rho^k (1-\rho)^{n-k} \left[\sum_{j=0}^{k} \binom{n}{j} \left[\frac{1}{2}\right]^n \left(\sum_{s=j}^{n} \binom{n}{s} \left[\frac{1}{2}\right]^n\right)^{m-2}\right]$$

5.2 Number of Votes for the Target and non-Target Images

Thus far, we computed the probability that a single window will vote for the target and/or one of the non-target memories. The decision network of the two-level decoupled Hamming memory counts up the votes for each of the *w* windows covering the memory key and then chooses the fundamental memory with the most votes. Hence, the question here is: What is the total number of votes received by the target and each of the non-target memories?

Let N_t denote the total number of votes received by the target memory and N_i the number of votes received by the *i*th non-target memory. Both N_t and N_i are random variables which follow a binomial distribution. The expected value and variance of the number of votes for the target are given by

$$\mu_t = E[N_t] = P_t \frac{N}{n}$$

$$\sigma_t^2 = V[N_t] = P_t(1 - P_t)\frac{N}{n}$$

Similarly, the expected value and variance for the number of votes received by the *i*th non-target memory is given by

$$\mu_i = E[N_i] = P_i \frac{N}{n}$$

$$\sigma_i^2 = V[N_i] = P_i(1 - P_i)\frac{N}{n}$$

By the central limit theorem, and assuming a large number of windows, the probability distribution of the random variables N_t and N_i approaches a normal distribution. In this case, the (approximate normal) density function f_t for the number of votes for the target image is given by

$$f_t(x) = \frac{1}{\sqrt{2\pi}\sigma_t} e^{\frac{-(x - \mu_t)^2}{2\sigma_t^2}} \qquad (5)$$

and the density function f_i for the number of votes for the *i*th non-target memory can be approximated by

$$f_i(x) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{\frac{-(x-\mu_i)^2}{2\sigma_i^2}} \tag{6}$$

In the normal approximations of Equations (5) and (6), we assume $N \to \infty$, $w \to \infty$, and n is held constant. Such approximations are useful in obtaining numerical values for N_t and N_i.

5.3 Estimation of Memory Capacity

In the previous subsection, we determined the expected number of votes for the target image and the expected number of votes for the *i*th non-target image. Of course the two-level decoupled Hamming memory retrieves the correct (target) image when N_t is larger than N_i for *all* of the $m - 1$ non-target memories in the fundamental memory set. That is, N_t must be larger than N_1 and N_2 and . . . and N_{m-1}, where we have assumed, without loss of generality, that the target memory is the *m*th memory (last memory) in the fundamental memory set. To compute the probability of correct retrieval, then, we must determine the maximum of the "other" or non-target votes; hence, we define a quantity N_i^{max}

$$N_i^{max} = max\{N_1, N_2, \ldots, N_{m-1}\} \tag{7}$$

It can be shown that the maximum of a collection of continuous random variables is also a random variable; furthermore, the cumulative distribution function (cdf) of the max random variable is given by the product of the individual cdf's of the random variables being maximized (Port, 1994).

In this case, we have $m - 1$ random variables which follow the continuous (and approximate) probability density functions $f_1, f_2, \ldots, f_{m-1}$ given in Equation 6. Suppose that the corresponding cumulative distribution functions are denoted by $F_1, F_2, \ldots, F_{m-1}$, respectively. Then, since each of these $m - 1$ distributions are identical, we have

$$F_{max}(x) = F_1(x)F_2(x)\ldots F_{m-1}(x) = [F_i(x)]^{m-1} \tag{8}$$

The probability density function of N_i^{max} can be obtained by differentiating Equation 8

$$f_{max}(x) = \frac{d}{dx}F_{max}(x) = (m-1)[F_i(x)]^{m-2}f_i(x) \tag{9}$$

Using the density function in Equation 9, the expected value of N_i^{max} can be computed as follows:

$$\overline{N}_i^{max} = E[N_i^{max}] = \int x f_{max}(x)dx \tag{10}$$

Finally, the probability of correct retrieval is the probability that $N_t > \overline{N}_i^{max}$, i.e., $P_{cor} = Prob(N_t > \overline{N}_i^{max})$ which is computed below

$$P_{cor}(n, N, \rho, m) = \int_{\overline{N}_i^{max}}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_t} e^{\frac{-(x-\mu_t)^2}{2\sigma_t^2}} dx \tag{11}$$

Using the standard normal distribution, Equation 11 can be recast as

$$P_{cor}(n, N, \rho, m) = \int_{\frac{\overline{N}_i^{max}-\mu_t}{\sigma_t}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} dz \tag{12}$$

And using the standard error function

$$erf(x) = \int_0^x \frac{2}{\sqrt{\pi}} e^{-z^2} dz$$

Equation (12) can be written as

$$P_{cor}(n, N, \rho, m) = \frac{1}{2}\left[1 + erf\left(\frac{\mu_t - \overline{N}_i^{max}}{\sqrt{2}\sigma_t}\right)\right] \tag{13}$$

Equation 13 gives the probability that the target image will be retrieved as a function of system dimension *N*, local window size *n*, noise level ρ, and total number of stored patterns *m*.

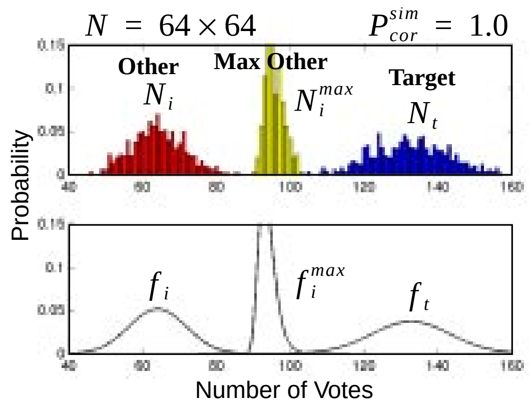
Numerical estimates of the capacity m^* of the memory, then, can be determined by fixing values for *n*, *N*, and ρ, and computing Equation 13 for increasing values of *m*.

6. Simulations

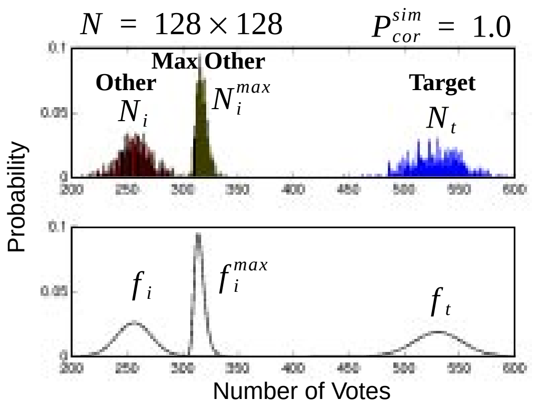
6.1 Expected Number of Votes

For the simulations of the two-level decoupled Hamming memory presented in this section, it is assumed that all memory patterns are 2-dimensional images and are generated randomly from a uniform distribution. As with the above theoretical analysis, it is assumed that one of the memory images (the target image) is selected and corrupted with an amount ρ of uniform random noise ($0 \le \rho \le 0.5$). This corrupted image is used as the memory key, and it is desired that the system produce the target image at the output; i.e. retrieve the target image.

Figure 6(a) shows a comparison of the simulation and theoretical results for the expected number of votes for the target N_t, an arbitrary other image (other than target) N_i, and the maximum among these other images N_i^{max}. For these simulations, $m = 10,000$ images, the noise level is set at $\rho = 0.4$, and the local window size is $n = 2 \times 2$. The top Figure shows the simulation results, and the bottom plot shows the corresponding theoretical distributions for these quantities, as given in Equations 5, 6, and 9. In (a), the image size is set at $N = 64 \times 64$, while in (b), the image size is $N = 128 \times 128$.



(a)



(b)

Figure 6. Comparison between simulation and theory for the case of a 2×2 window size, input noise $\rho = 0.4$, memory size of $m = 10000$, and an image size of (a) 64×64 and (b) 128×128.

The simulation results were obtained as follows. The corrupted target memory (with 40% random noise) was input to the two-level Hamming memory, and the number of votes received by the target, a randomly chosen other (non-target) image, and the maximum among the non-target images were recorded. This process was repeated 500 times. In each case, if the system retrieved the target image, the retrieval was considered a success; otherwise it was counted as a failure. The probability of correct retrieval was simply the number of successes out of 500 trials (Watta, Ikeda, Artiklar, Subramanian, and Hassoun, 1999).

Clearly, the theoretical results provide a good model for the underlying distributions. Note that in both cases, there is sufficient separation in the distributions of N_t and N_i to successfully perform the classification, and hence the probability of correct retrieval is unity: $P_{cor} = 1$.

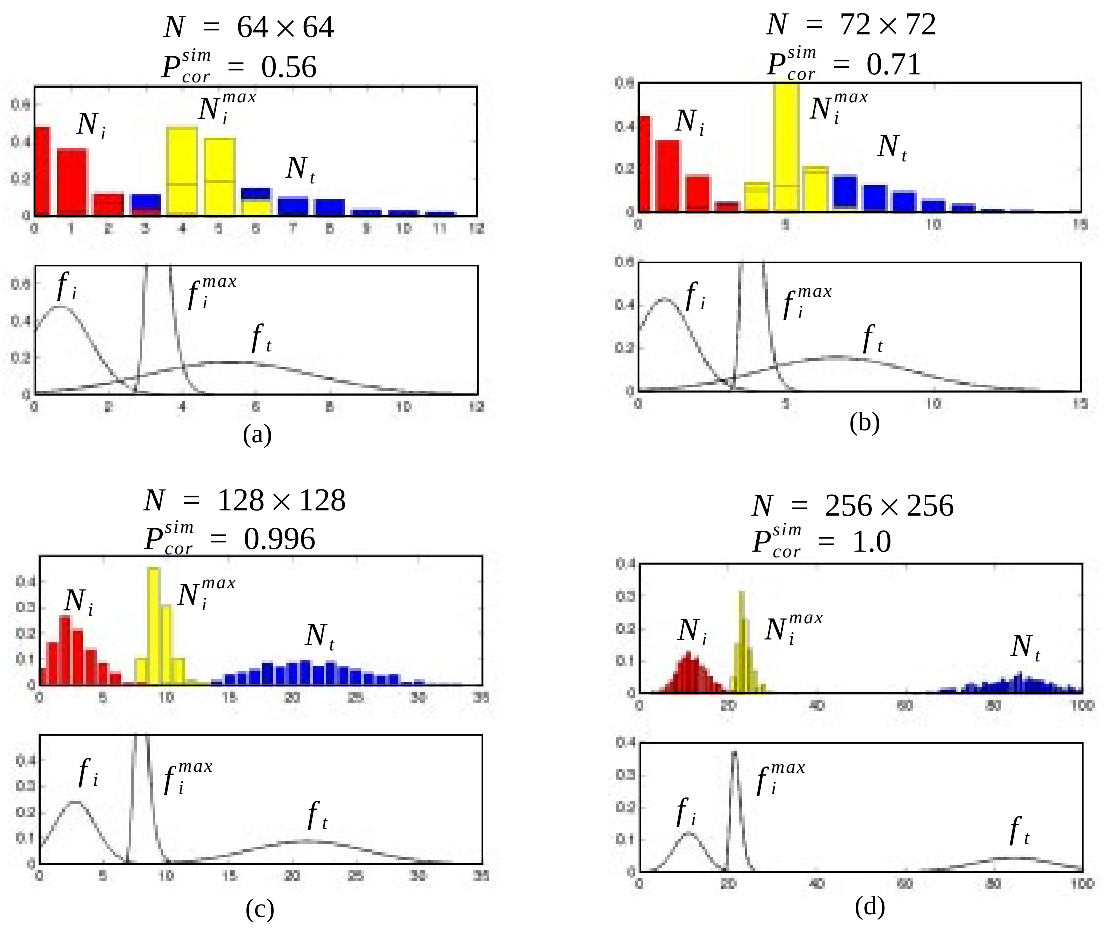


Figure 7. Comparison between simulation and theory for the case of a 4×4 window size and noise of $\rho = 0.4$ and a memory size of $m = 1000$, and an image size of (a) 64×64, (b) 72×72, (c) 128×128, and (d) 256×256.

Figure 7 shows a case where there is overlap between the distributions. Here, the number of images in the memory set is $m = 1000$, the input noise is $\rho = 0.4$, and the local window size is $n = 4 \times 4$. Figure 7(a) shows the results in the case of an image size of $N = 64 \times 64$. Here, the probability of correct retrieval is $P_{cor}^{sim} = 0.56$. Figure 7(b) shows the results for $N = 72 \times 72$. In this case, the probability of correct retrieval is $P_{cor}^{sim} = 0.71$. Similarly, (c) and (d) show the results for larger image sizes: $N = 128 \times 128$, and $N = 256 \times 256$. As expected, for a fixed number of memory patterns, as the image size increases, the separation between N_i and N_t increases.

6.2 Probability of Correct Retrieval

Fixing the system dimension at $N = 64 \times 64$ and the noise level at $\rho = 0.4$, the simulations of the previous section were repeated for various values of the number of fundamental patterns m from $m = 10$ to $m = 10,000$. Figure 8 shows the probability of correct retrieval vs. the number of stored patterns for various local window sizes. The dashed line gives the simulation result, and the solid lines gives the theoretical values from Equation 13.

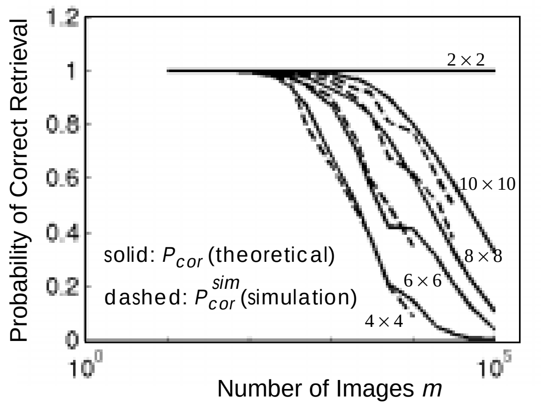


Figure 8. Probability of correct retrieval vs. number of stored patterns *m*.

Note that when the window size does not divide the image size evenly (as in the 6×6 and 10×10 windows), then some overlap of neighboring windows is necessary to cover the whole image. In our simulations, we simply overlap the right-most modules and lower-most modules if needed. So to cover the 64×64 image requires a 7×7 array of the 10×10 windows.

As noted earlier, the two-level Hamming network reduces to the full Hamming memory in both of the extreme cases for the window size. This is illustrated in Figure 8, where the small 2×2 neighborhood size gives very good performance. As the neighborhood size increases, though, the performance degrades. In particular, the 4×4 window gives the worst performance. By increasing the window size above 4×4, the performance improves.

Figure 9 shows the probability of correct retrieval when only portion of the image, as shown in Figure 4, is covered by local windows (here, $N = 64 \times 64$, $\rho = 0.4$, and n is set at the worst possible value: $n = 4 \times 4$). The dashed line gives the simulation result, and the solid lines gives the theoretical values. Using fewer windows amounts to a reduction in the dimension of the system and, as expected, the capacity decreases.

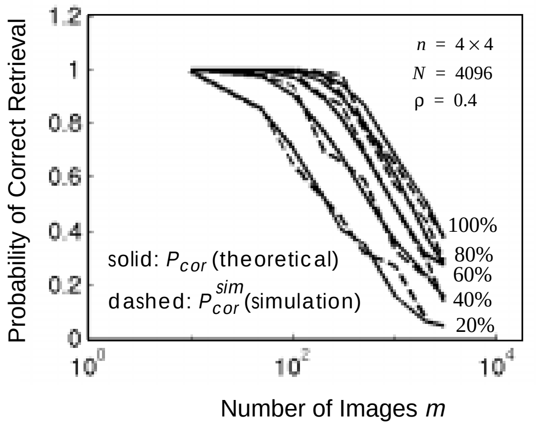


Figure 9. Probability of correct retrieval for different percent coverages: 20%, 40%, 60%, 80%, and 100%.

6. 3 Capacity and Error Correction

The notion of capacity that will be used here is as follows: Assuming an input noise level of ρ, how many input patterns can be reliably stored? (Hertz et al, 1991; Kawamura and Hirai, 1997). This measure of capacity is based on the error correction capability of the associative memory. That is, if some percentage of bits of the input pattern are corrupted, can the memory still retrieve the correct pattern?

Plots of the capacity of the two-level decoupled Hamming memory can be generated as follows. First, values for n, N, and ρ are fixed. Then, starting with a small value for m, P_{cor} is computed using Equation 13. Initially, with such a small m, P_{cor} is a near 1.0. As m is slowly increased, P_{cor} decreases in value. This process of decreasing m is continued until P_{cor} falls below 0.99. In this case, the largest value of m which gives $P_{cor} \geq 0.99$ is taken as the capacity m^* of the memory.

Figure 10 shows a plot of the capacity of the 2-level decoupled Hamming memory vs. the input noise level for various window sizes: 4×4, 6×6, 8×8, and 10×10. For these experiments, the image size was fixed at $N = 64 \times 64$. As expected, the capacity decreases as the noise increases.

Figure 11 shows a plot of capacity vs. image size for various values of input noise: $\rho = 0.3$, 0.35, 0.4, and 0.45. Again, the image size is fixed at 64×64.

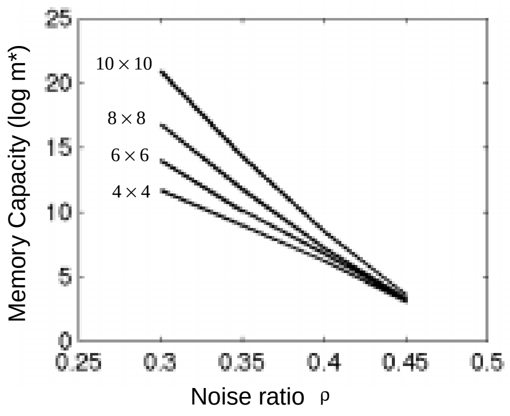


Figure 10. Capacity vs. noise level for window sizes: 4×4**,** 6×6**,** 8×8**, and** 10×10**.**

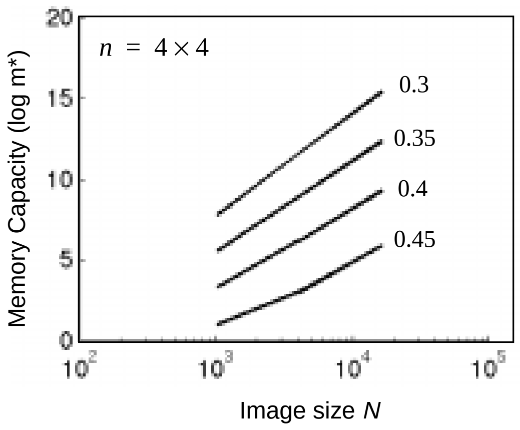


Figure 11. Capacity vs. image size for various noise levels: 0.3, 0.35, 0.4, 0.45.

7. Correlated Memory Sets

The theoretical and simulation results given in the previous sections assumed that the patterns to be stored consisted of random binary patterns. In practice, though, one usually wants to store highly correlated patterns, such as images of faces, fingerprints, voice waveforms, etc. To test the performance of the two-level decoupled Hamming memory on such correlated memory sets, we constructed a database of 200 binary face images (Watta, Artiklar, Masadeh, and Hassoun, 2000) of size 82x115. Some samples of these face images are shown in Figure 12. In addition, Figure 12 also shows the face images corrupted with various amounts of uniform random noise. Notice that at 25 and 40% noise, the recognition problem becomes difficult for humans.

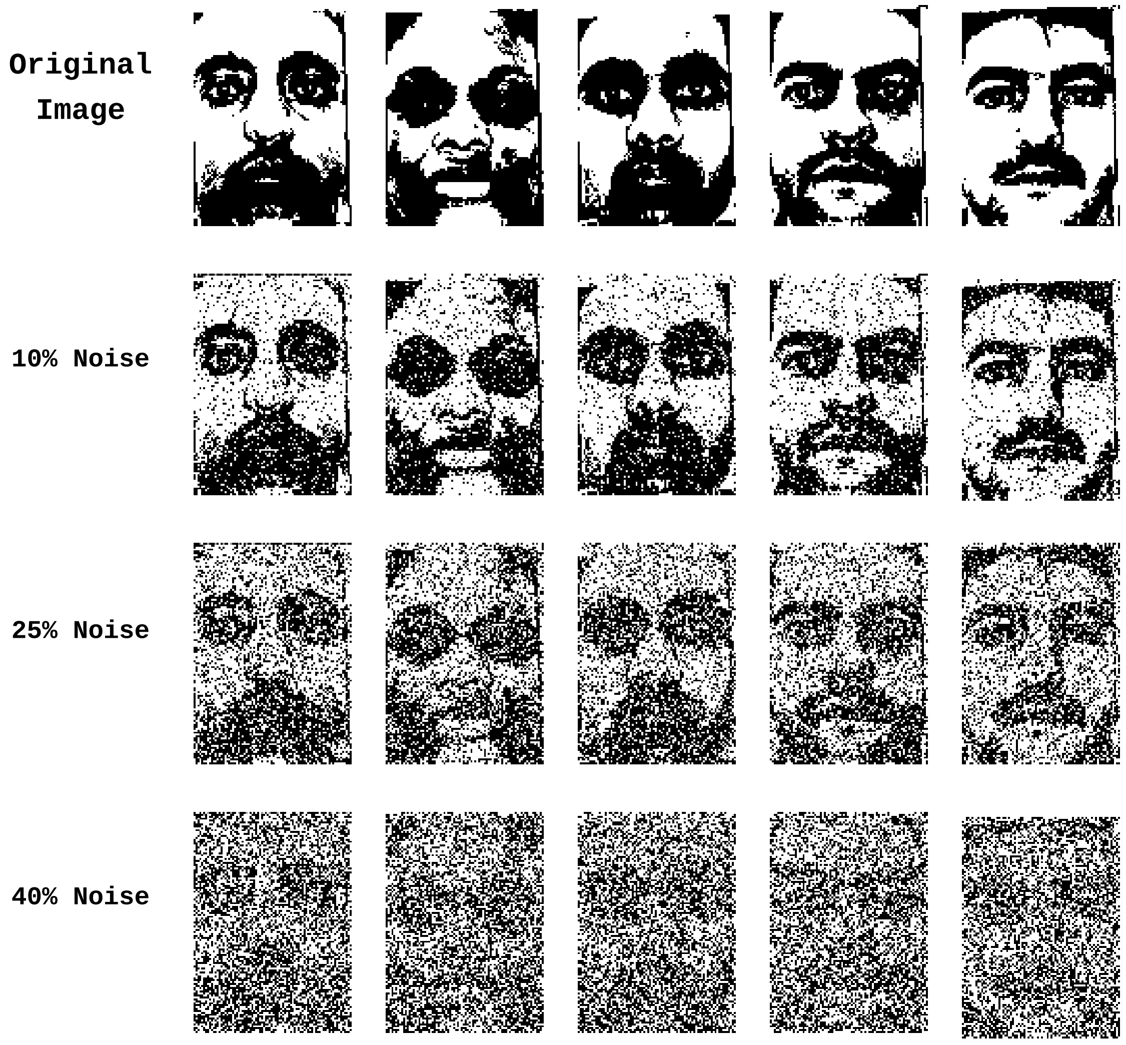


Figure 12. Samples of 82x115-dimensional face images in the 200-person memory set. The original images are shown in the top row, and subsequent rows the patterns corrupted with various amounts of uniform random noise.

For comparison with results presented in the previous section, the face images were cropped to a size of 64×64 and 72×72 pixels, as shown in Figure 13.

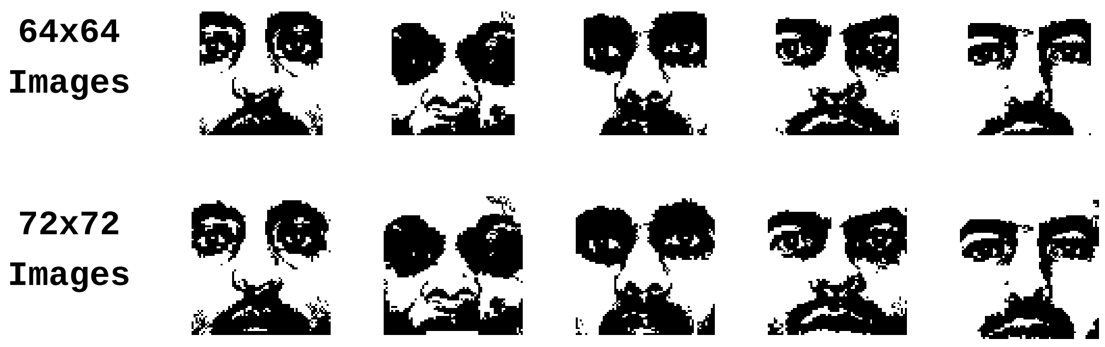


Figure 13. Samples of 642x64 and 72x72-dimensional face images in the 200-person memory set.

Table 1 shows the probability of correct retrieval over the 200-faces memory set using a noise level of 40%. As with the binary images, the performance of the memory is high for small and large windows and deteriorates slightly for intermediate-sized windows. Note that for noise levels lower than 40%, the probability of correct retrieval is 100% for all window sizes and all 3 image sizes.

N \ n	2x2	3x3	4x4	9x9	15x15	20x20	25x25
64x64	100	96	90	86	100	100	100
72x72	100	98	95	93	99	100	99
82x115	100	99	100	100	100	100	100

Table 1. Probability of correct retrieval (in percent) on the 200-image memory set of human faces.

The results in Table 1 were obtained in a similar fashion to the experiments on the random images. Each of the 200 face images was corrupted with 40% noise and input to the two level Hamming memory. If the output of the Hamming memory was the original person, then the trial was counted as a success; otherwise it was counted as a failure. The probability of correct retrieval was then the number of success over 200. The above process was repeated 25 times and the results averaged over those 25 times.

8. Summary

In this paper, we investigated the computational capabilities of a class of associative memory models called the *two-level decoupled Hamming associative memory.* This memory model is a generalization of the Hamming associative memory, and allows for local distance measures and a voting mechanism. We successfully formulated a theoretical analysis of this model which characterized its memory capacity and error correction capability in the case of random memory patterns. In particular, an expression was derived for the probability of correct retrieval as a function of pattern size N, window size n, noise level ρ, and number of patterns stored, m: $P_{cor} = P_{cor}(n, N, \rho, m)$.

Simulation results were shown to be in close agreement with the theoretical results. More importantly, the capacity of the two-level decoupled Hamming memory was shown to be substantially larger than other single layer neural net memories, such as the Hopfield network. In fact, the two-level decoupled Hamming network can perform well even when the entire input image is not covered with local windows. The simulations results in Figure 8 showed a gradual decrease in performance as fewer and fewer local windows were used.

As expected, the two-level Hamming memory performs best when a single local window is used: $w = 1$ or when the maximal number of windows are used: $w = N$ (each pixel is a local window). For intermediate window sizes, the capacity of the two-level memory is not as large. Interestingly, the worst performing local window size was found to be of size 4×4 (for random input images).

In the case of correlated memory patterns, simulation results showed that the two-level decoupled Hamming memory can store and correctly retrieve several hundred human face images, even in the presence large amounts of noise (40% bit errors).

The two-level Hamming network (and Hamming network) achieves associative memory by using a template matching approach, where the input image is compared to each of the prototype images. 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 (Chellappa, Wilson, and Sirohey, 1995). 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 (1973): ". . . 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. The severity of the storage/speed dilemma has diminished in recent years with the availability powerful and low cost PCs, and hence the template matching approach of the Hamming network warrants further consideration.

It is easy to see that there are applications, however, where the two-level Hamming memory requires *less* storage than neural net-based approaches. Consider the simplest type of associative neural memory: a single layer Hopfield-type network (Hopfield, 1982 and 1984). Suppose we want to store a 512×512 gray-level image of each of the 9,000 undergraduate students at the University of Michigan-Dearborn. In this case, the two-level decoupled Hamming memory requires about 2.5 GB of storage (of size character), whereas the Hopfield weight matrix requires the storage of about 68×10^9 weights. Since weights are usually stored as floating point numbers, the Hopfield weight matrix would require about 136 GB of memory (assuming 2 bytes for each floating point number). Clearly, whenever $mN < N^2$, the two-level decoupled Hamming memory requires less storage than a single layer neural associative neural memory (ANM). Of course, the single layer ANM is the simplest network architecture possible, and more sophisticated multilayer associative neural memories usually require considerably more than N^2 weights.

In future papers, we will extend the two-level Hamming network to the case of gray level images and present additional results on the use of this memory for human face recognition problems.

Acknowledgments

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

References

1. Bishop, C. (1995). *Neural Networks for Pattern Recognition*, Oxford University Press, New York.

2. Buckingham, J. and Willshaw, D. (1992). “Performance Characteristics of the Associative Net,” *Network*, **3**, 407-414.

3. R. Chellappa, C. Wilson, and S. Sirohey, “Human and Machine Recognition of Faces: A Survey,” *Proceedings of the IEEE*, **83**(5), 705-740, 1995.

4. Chou, P. (1989). “The Capacity of the Kanerva Associative Memory,” *IEEE Transactions on Information Theory*, **35**(2), 281-298.

5. R. Duda and P. Hart, “Pattern Classification and Scene Analysis, John Wiley & Sons, New York, 1973.

6. Hamming, R. (1986). *Coding and Information Theory.* Prentice-Hall, Englewood Cliffs, NJ.

7. Hassoun, M. H., ed. (1993). *Associative Neural Memories: Theory and Implementation*. Oxford

University Press, New York.

8. Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks*, MIT Press, Cambridge, Mass.

9. Hassoun, M. and Watta, P. (1996). "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., pp. 583-587.

10. Hertz, J., Krough, A., and Palmer, R. (1991). *Introduction to the Theory of Neural Computation*, Addison-Wesley, Reading, MA.

11. Ho, T., Hull, J., and Srihari, S. (1994). "Decision Combination in Multiple Classifier System," *IEEE Transactions on Neural Networks*, **16**(1), 66-75.

12. Hopfield, J. (1982). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences*, USA, **79**, 2445-2558.

13. Hopfield, J. (1984). "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proc. Natl. Acad. Sci., USA, Biophys*. **81**, 3088-3092.

14. Ikeda, N., Watta, P., and Hassoun, M. (1998). "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, pp. 486-491.

15. Kawamura and Hirai, (1997). "Storage Capacity Analysis on a Model of Human Associative Processing, HASP" *Systems and Computers in Japan*, **23**(1), 24-33.

16. Kohonen, T. (1989). *Self-Organization and Associative Memory.* Springer-Verlag, Berlin.

17. Kosko, B. (1988). "Bidirectional Associative Memories," *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-18**, 49-60.

18. Palm, G. (1980). "On Associative Memory," *Biological Cybernetics*, **36**, 19-31.

19. Port, S. (1994). *Theoretical Probability for Applications*, John Wiley and Sons, New York.

20. Pulkki, V. and Taneli, H. (1996). "An Implementation fo 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., pp. 1345-1349.

21. Watta, P. B., Akkal, M., and Hassoun, M. H. (1997). “Decoupled Voting Hamming Associative Memory Networks” *Proceedings of the IEEE International Conference on Neural Networks*, ICNN’97, June 9-12, 1997, Houston, Texas, pp. 1188-1193.

22. P. Watta, M. Artiklar, A. Masadeh, and M. H. Hassoun, (2000). “Construction and Analysis of a Database of Face Images which Requires Minimal Preprocessing,” *Proceedings of the IASTED Conference on Modeling and Simulation MS’2000*, May 15-17, 2000, Pittburg Pennsylvania, 465-469.

23. Watta, P. B. and Hassoun, M. H. (1996). “Efficient Realization of the Grounded Hamming Associative Memory,” *Proceedings of the World Congress on Neural Networks*, WCNN’96, Sept. 15-20, San Diego, CA, pp. 763-767.

24. Watta, P., Ikeda, N., Artiklar, M., Subramanian, A., and M., and Hassoun, M. H. (1999). “Comparison Between Theory and Simulation for the Two-Level Decoupled Hamming Associative Memory,” *Proceedings of the IEEE International Conference on Neural Networks*, IJCNN’99, July 10-16, 1999, Washington, DC.

25. Watta, P., Wang, and Hassoun, M. (1997). “Recurrent Neural Nets as Dynamical Boolean Systems with Application to Associative Memory,” *IEEE Transactions on Neural Networks*, **8**(6).

26. Willshaw, D., Buneman, O., and Longuet-Higgins, H. (1969). “Non-holographic Associative Memory,” *Nature*, **222**, 960-962.