A Weighted Voting Model of Associative Memory: Experimental Analysis

Xiaoyan Mu
Department of Electrical and Computer Engineering
Rose-Hullman Institute of Technology
Terre Haute, IN 47803
mu@Rose-Hulman.edu

Paul Watta
Dept. Electrical and Computer Engineering
University of Michigan-Dearborn
Dearborn, MI 48128
watta@umich.edu

Mohamad H. Hassoun
Dept. Electrical and Computer Engineering
Wayne State University
Detroit, MI 48202
hassoun@eng.wayne.edu

Abstract

In a related paper [1], a weighted voting RAM-based associative memory model was proposed, and a theoretical analysis of its performance on binary and random memory sets was given. In this paper, we give an experimental analysis of the weighted voting memory using both binary-random memory sets and more practical memory sets consisting of grayscale face images. The results show that the weighted voting memory offers higher performance over the voting memory on both types of memory sets.

I. Introduction

The associative memory problem is stated as follows: We are given a *fundamental memory set* of desired associations: $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$, where $\mathbf{x}_i \in X \subset R^N$ and $\mathbf{y}_i \in Y \subset R^L$, $i = 1, 2, \ldots, m$. The task is to design a system which robustly stores the fundamental associations, such that

(1) When presented with $\mathbf{x}_i$ as input, the system should produce $\mathbf{y}_i$ at the output.

(2) When presented with a *noisy* (corrupted, distorted, or incomplete) version of $\mathbf{x}_i$ at the input, the system should also produce $\mathbf{y}_i$ at the output.

(3) When presented with input $\mathbf{x}$ that is *not sufficiently similar* to any of the inputs in the memory set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, the system should reject the input.

The meaning of the words *noisy* and *not sufficiently similar* are application dependent. For example, in an image processing application, translation, rotation, and scale variations are common types of image distortion. In order to achieve all 3 of these specifications, the system should have a rejection mechanism capable of rejecting patterns with low signal-to-noise ratio.

As suggested in [1], to measure how well the system performs on all 3 of these tasks, two types of tests are used: *identification test* and *watch list test*. In both cases, we start with the given fundamental memory set. In the identification test, the system is tested with noisy versions of the fundamental patterns. For a given input, the task is to identify which of the fundamental patterns it should be associated with. In this case, no rejection state is needed. The measure of merit here is the *identification rate* (IR), which is the probability that a given input will be matched with the correct stored memory.

In the watch list test, the system must have a rejection mechanism. Here, two test sets are required: TS_G, which contains noisy versions of the fundamental patterns, and TS_N, which contains spurious or imposter input patterns which should **not** be associated with any of the fundamental patterns. For the TS_G test set, there are two measures of merit: the *detection and identification rate* (DIR), which is the percentage of input patterns that are correctly matched with the known fundamental patterns, and the *false rejection rate* (FRR), which is the percentage of patterns that are rejected by the system [3]. For the TS_N test set, there is only one measure of interest: the *false acceptance rate* (FAR), which gives the percentage of imposter patterns that are incorrectly matched with a fundamental pattern. Ideally, we would like the system to reject all of the imposter patterns and hence achieve a false acceptance rate of 0%.

Of course there is a trade-off between the detection and identification rate and the false acceptance rate. It is desirable to design systems with a tunable parameter or threshold T which allows one to control the trade-off between DIR and FAR. A *receiver operating characteristic* (ROC) curve can be constructed which shows how DIR and FAR vary as a function of T. Such an ROC curve is an appropriate measure of memory capacity for systems designed to meet all 3 of the above design specifications.

II. The Voting Associative Memory

The weighted voting memory is an extension of the voting memory described in [2]. In the voting associative memory, the N-dimensional input $\mathbf{x}$ and each memory pattern $\mathbf{x}_k$ are partitioned into a collection of non-overlapping windows of size n. For notional simplicity, we will assume that n divides N, hence N/n is an integer.

For the input (memory key), let $\mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[N/n]$ denote the data in each window. That is, $\mathbf{x}[i]$ is the portion of $\mathbf{x}$ contained in the ith window, etc. The database patterns are partitioned in the same way: $\mathbf{x}_k[1], \mathbf{x}_k[2], \ldots, \mathbf{x}_k[N/n]$, $k = 1, 2, \ldots, m$. The partitioned database patterns can be stored in a RAM-type network, where the ith RAM holds all the database patterns associated with the ith window: $\mathbf{x}_1[i], \mathbf{x}_2[i], \ldots, \mathbf{x}_M[i]$. Figure 1 shows the architecture of a RAM network with 9 windows arranged in a 3×3 structure.

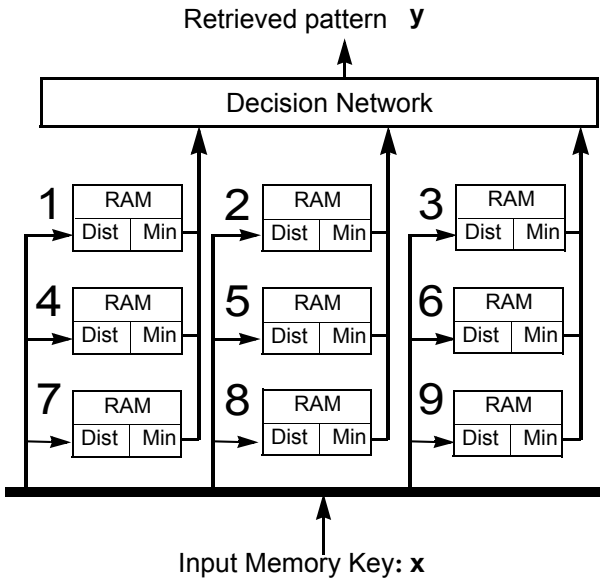

Figure 1. Structure of the voting associative memory.

The voting network requires a distance measure to be computed locally at each window. Let d be a distance measure between two n-dimensional vectors. Any suitable distance function can be used. For example, for binary memory patterns, the Hamming distance can be used; for real-valued patterns, the city-block distance can be used. In either case, the (local) distance between $\mathbf{x}_k[i]$ and $\mathbf{x}[i]$ is given by:

$$d(\mathbf{x}_k[i], \mathbf{x}[i]) = \sum_{j=1}^{n} \left| \mathbf{x}_k[i]_j - \mathbf{x}[i]_j \right|$$

where $\mathbf{x}_k[i]_j$ and $\mathbf{x}[i]_j$ denote the jth component of $\mathbf{x}_k[i]$ and $\mathbf{x}[i]$, respectively.

At each window, we compute a *local distance* $d_1, \ldots, d_m$ between the input key and all the memory patterns. The smallest distance is found, say d_k, and then the local window casts a vote for memory pattern k. The decision network examines the votes of all the windows and chooses the memory pattern that received the most votes.

It is easy to introduce a rejection mechanism in the voting model. We simply use a threshold T to indicate whether the number of votes received by the best matching pattern is sufficiently large. In the case that the number of votes received is less than T, then the input is rejected.

III. The Weighted Voting Associative Memory

The weighted voting model operates as follows. Again, we compute local distance measures at each window. But instead of just choosing the smallest one, we sort all the distances and assign a rank to each. Let the memory set pattern that has the smallest (local) distance be assigned $rank = 1$. The pattern with the next smallest distance will have $rank = 2$, etc. The distance computations and ranking of memory set patterns are done independently by each local window. Hence the weighted voting model has the same parallel structure as the voting model shown in Figure 1. The only difference is that each window now requires a sorting operation and not a simple min select.

The decision network tallies up the votes as follows. For each memory set pattern $\mathbf{x}_k$, let $N_{\mathbf{x}_k}^{(1)}$ be the number of windows that were found to have rank 1, $N_{\mathbf{x}_k}^{(2)}$ the number of windows that have rank 2, etc. In general, let $N_{\mathbf{x}_k}^{(r)}$ denote the number of windows that have rank r, $r = 1, 2, \ldots, m$. The total number of votes $N_{\mathbf{x}_k}$ assigned to pattern $\mathbf{x}_k$ is a weighted sum of the number of windows at each ranking $N_{\mathbf{x}_k}^{(1)}, N_{\mathbf{x}_k}^{(2)}, \ldots, N_{\mathbf{x}_k}^{(m)}$ that $\mathbf{x}_k$ received:

$$N_{\mathbf{x}_k} = \alpha_1 N_{\mathbf{x}_k}^{(1)} + \alpha_2 N_{\mathbf{x}_k}^{(2)} + \ldots + \alpha_m N_{\mathbf{x}_k}^{(m)} \qquad (1)$$

where the weights $\alpha_1, \alpha_2, \ldots, \alpha_m$ are used to adjust the relative importance of each ranking. There are many ways to choose the weights. In [4], theoretical values for the weights are derived for binary and random memory using: $\alpha_r = P(\omega_t | rank = r)$. For more practical memory sets, the weights can be determined empirically using a training phase and an additional set of training data.

Once the weighted vote is computed, the output is determined similar to the voting memory. The pattern which received the largest weighted vote is selected as the candidate output. If the number of votes exceeds the threshold, the candidate is accepted as the system output; otherwise, the input pattern is rejected.

II. Model Parameters

Both the voting memory and the weighted voting memory have the following main parameters:

N	dimension of input patterns $\mathbf{x}_i$
n	window size
m	number of memory patterns

Table 1. System-level parameters of the voting and weighted voting associative memory models.

For the case of binary and random memory sets, an additional noise parameter ρ can be used to indicate how noisy the input patterns are. That is, each of the input bits are corrupted with probability ρ.

When probing the system with a noisy version of one of the fundamental memory set patterns (i.e., a pattern from TS_G), there is one *target* memory set pattern (to which the input should be associated). All other patterns in the memory set are said to be *non-target* patterns. When probing the system with an imposter pattern from TS_N, all of the memory set patterns are non-target patterns.

At the local level (i.e., at a single window), what is of interest is how often the target memory gets a vote:

P_t	Probability that the target pattern gets a vote when probed with an element of TS_G.
P_i	Probability that a non-target pattern gets a vote when probed with an element of TS_G.
$P_{i'}$	Probability that a (non-target) pattern gets a vote when probed with input from TS_N.

Table 2. Local-level probabilities.

The higher-level decision network is responsible for tallying the votes and determining the system output. The relevant quantities are listed in Table 3.

For the identification problem, the probability of correct retrieval is simply the probability that the number of votes received by the target image exceeds that received by the maximum non-target image:

$$P_{id} = Prob(N_t > N_i^{max}) \tag{2}$$

In the case of the watchlist test, the input is rejected when the number of votes fails to exceed the threshold; hence the detection and identification rate (DIR) is the probability

$$P_{dir} = Prob(N_t > N_i^{max} \cap N_t > T) \tag{3}$$

N_t	Number of votes for the target pattern when probed with an element of TS_G.
N_i	Number of votes for the ith non-target pattern when probed with input from TS_G.
N_i^{max}	Number of votes received by the best-matching of all the non-target patterns N_i.
$N_{i'}$	Number of votes for the ith non-target pattern when probed with input from TS_N.
$N_{i'}^{max}$	Number of votes received by the best-matching of all the non-target patterns $N_{i'}$.

Table 3. Decision network (higher level) quantities.

Finally, the probability of false acceptance is the probability that one of the memory patterns receives more than T votes:

$$P_{far} = P(N_{i'}^{max} \geq T) \tag{4}$$

For memory sets consisting of binary and random patterns, it is possible [*] to derive theoretical expressions for all of the quantities listed in Tables 2 and 3, as well as the performance measures P_{id}, P_{dir}, and P_{far}, as a function of the system parameters; for example, $P_{dir} = P_{dir}(n, N, m, \rho)$.

III. Random Patterns: Number of Votes

For the first (identification) experiment, the following parameters were chosen: memory set size: $m = 150$, pattern dimension: $N = 300 \times 300$, window size: $n = 10 \times 10$, and input noise: $\rho = 0.48$. Figure 2 shows the results for the voting network. Here, Figure 2(a) shows a histogram of the number of votes (experimentally observed) received by the target pattern $\tilde{N}_t$, a typical non-target pattern $\tilde{N}_i$, and the non-target pattern with the most votes $\tilde{N}_i^{max}$. Figure 2(b) shows the corresponding theoretical results.

Figure 3 shows the resulting histograms for the weighted voting model. In Figure 3(a), the results of the simulation are given, (b) shows the corresponding theoretical results using the discrete model. Comparing with Figure 2, we see that for the same parameters, the weighted voting model offers better performance because the distributions for N_t and N_i^{max} are more separated. For the voting model, the correct retrieval rate was experimentally found to be $\tilde{P}_{id} = 78.0$, with theoretical prediction $P_{id} = 79.7$. For the weighted voting model, the experimental result was $\tilde{P}_{id} = 99.6$ and the corresponding theoretical result is $P_{id} = 99.6$.

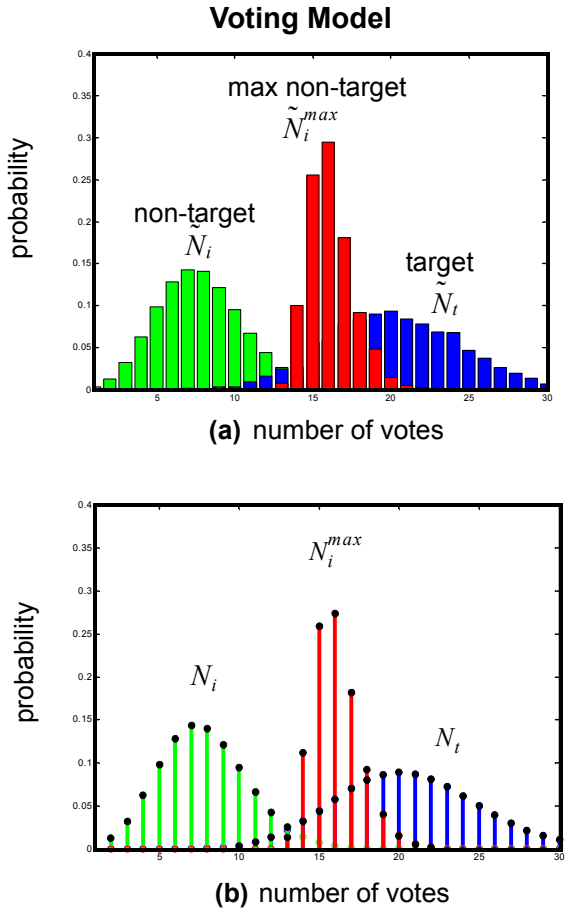

Figure 2. Results of the voting model with parameters: $N = 300 \times 300$, $n = 10 \times 10$, $\rho = 0.48$, and $m = 150$. (a) simulation results and (b) theoretical results.

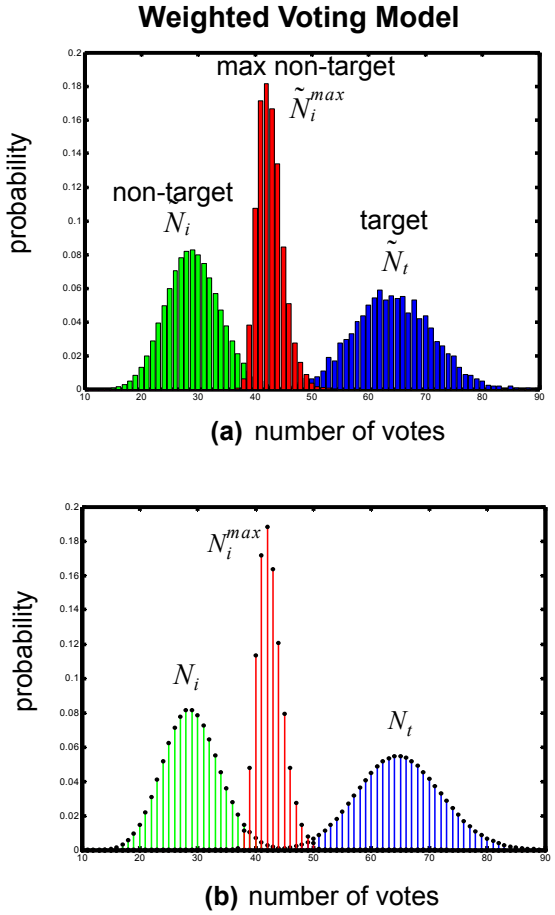

Figure 3. Results of the weighted voting model with parameters: $N = 300 \times 300$, $n = 10 \times 10$, $\rho = 0.48$, and $m = 150$. (a) simulation results and (b) a theoretical results.

IV. Random Patterns: Identification

For the experiments in this section, the parameters were set at: pattern dimension: $N = 60 \times 60$, window size: $n = 5 \times 5$, and input noise: $\rho = 0.4$. Figure 4 shows how the identification rate for both the voting and weighted voting memory varies as the memory set size is increased from $m = 100$ to $m = 1000$.

In this Figure, the experimental results are shown as circles and the theoretical results are shown as a dashed line. There is close agreement between the two.

Clearly, the performance of the voting model degrades sharply as the number of memory patterns increases. The weighted voting model, though, is much more robust and is able to achieve a retrieval rate of about 95% when storing 1000 memory patterns. The results here pertain to a (rather large) noise level of $\rho = 0.4$. For smaller noise levels, the identification rates are even higher.

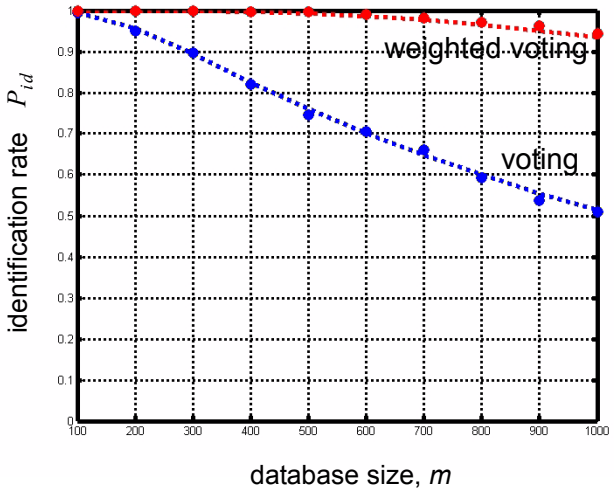

Figure 4. Effect of database size on identification rate for the voting model (lower curve) and the weighted voting model (upper curve).

The selection of the window size n is an important consideration for both the voting and weighted voting models. Figure 5 shows how the retrieval rate varies as a function of window size for (a) the voting model and (b) the weighted voting model.

As noted in [2], the performance of the voting models is best when using very small window sizes (for example, 2×2) or large window sizes (for example, larger than 10×10). Performance suffers when intermediate window sizes are used (such as 5×5 or 6×6).

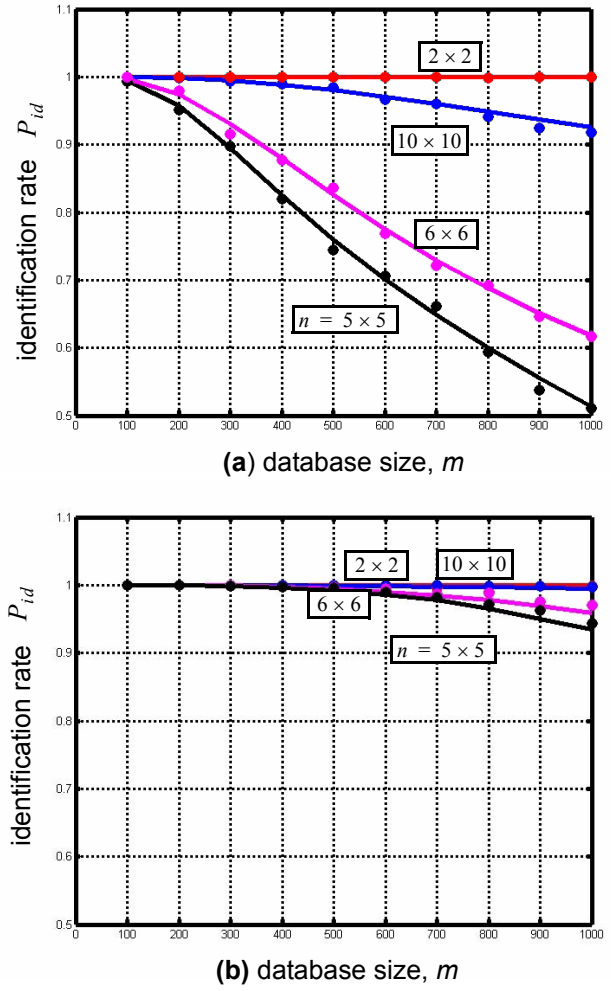

Figure 5. Effect of window size on the retrieval rate for (a) the voting model and (b) the weighted voting model. Both experimental results (circles) and theoretical results (solid line) are shown. Here $N = 60 \times 60$ and $\rho = 0.4$.

V. Random Patterns: Watchlist

An ROC curve showing how both P_{dir} and P_{far} vary for various memory set sizes is shown in Figure 6. In (a), the results for the voting model are given. Here, the solid line is the experimental result and the dashed line is the theoretical result. The results for the weighted voting model are shown in (b). The performance of the weighted voting model is very good, and hence most of the results are clumped at the top of the graph.

Recall that ROC curves are constructed by using various values of the system threshold T. For the left-most part of each curve, a large value of the threshold is used. As the threshold is reduced, the detection and identification rate increases, but at the expense of a higher false acceptance.

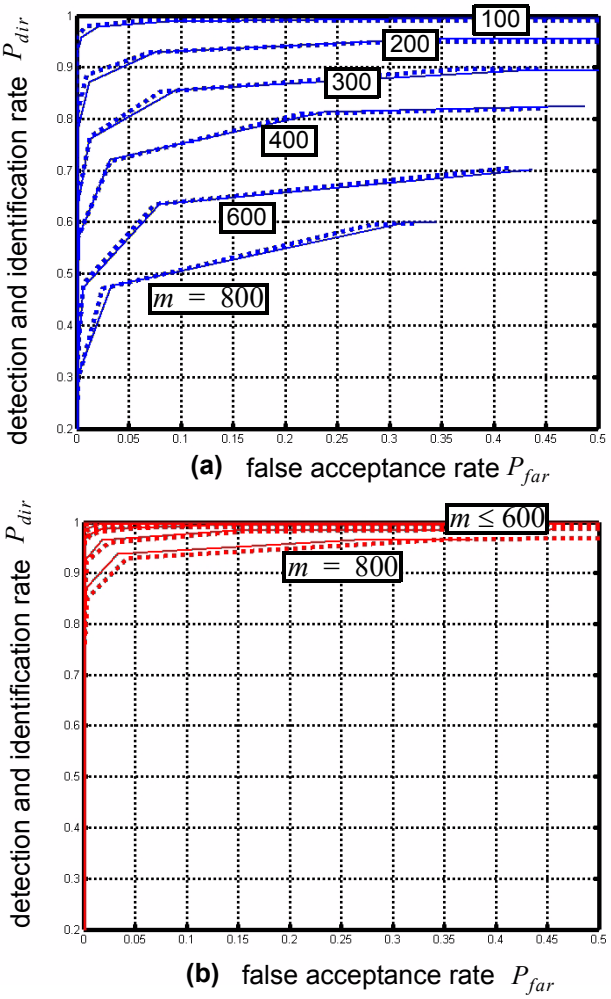

Figure 6. ROC curves showing how P_{dir} and P_{far} vary as a function of the threshold (implicit in the graph) and memory set size for (a) the voting model and (b) the weighted voting model. Here, $N = 60 \times 60$, $n = 5 \times 5$ and $\rho = 0.4$.

VI. Grayscale Patterns: Watchlist

One of the advantages of the voting and weighted voting models is that they can easily operate in heteroassociative mode and handle grayscale data. In the context of human

face recognition, the fundamental memory set consists of a database of face images, where the input $\mathbf{x}_i$ is an image of the ith person, and the associated output $\mathbf{y}_i$ is the corresponding name of the person.

In the face recognition community, the FERET database is commonly used to benchmark performance. This database contains a set of frontal images **fa** containing 1196 people. Here, **fa** is used as the memory set. Another set of frontal images **fb** (containing 1195 people) are used as the test set. For each image in **fa** and **fb**, the position of the eyes is used to crop and scale the face to a standard size (hence, this qualifies as a partially automated test in the FERET terminology).

Note that since the FERET database has only 1 image per person in the memory set, there is not enough training data to compute weights for the weighted voting memory. In this case, we used the following heuristic to set the weights:

$$\alpha_r \equiv \frac{1}{r}$$

Here, the weight assigned to the best matching pattern (the pattern with rank 1) is 1, the weight for the second best pattern is $1/2$, third best is $1/3$, etc. We have found empirically that this choice of weights works fairly well for the face recognition problem.

For both the voting and weighted voting systems, the window size was set at 13×13. In order to make the voting models more robust to small shifts and rotations, we applied an elastic graph matching-type procedure [5] to optimize the position of each local window. That is, when computing the local distance between the input and a memory image, we compute the distance in the standard position of the window as well as nearby positions (by perturbing the position of the window a few pixels in each direction). Among these candidate distances, the smallest one is chosen and used as the local distance.

The results of the watchlist test on the full FERET **fb** test set are shown in Figure 7 for both the voting (dashed line) and weighted voting (solid line) models. Note that since an additional set of stranger images are not available, we used a leave-one-out strategy to measure the false acceptance rate. That is, the first memory image is removed from the memory set (and hence is now a stranger) and used as input to the network. After the output is obtained, the first image is put back into the memory set. The same procedure is used for all the other memory images.

VII. Conclusions

The simulations reported here show that the weighted voting memory consistently outperforms the voting memory. The price to pay for this increased performance is in terms of *computation time*: the weighted voting memory requires a local sorting operation, while the voting memory only requires a min-select operation, and *training*: the weighted voting memory requires training data and a training phase in order to compute the weights, while the voting memory does not require such a training phase.

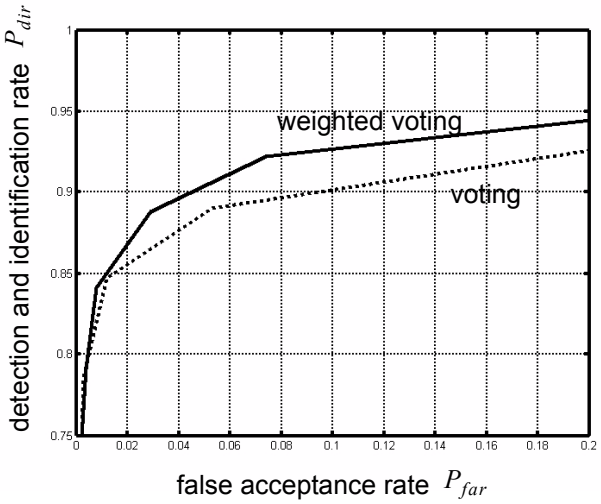

Figure 7. ROC curve for the full FERET database. The results of the voting and weighted voting models are shown.

The weighted voting strategy outlined here is a quite general and can be used with other types of features; for example, eigenfaces, wavelets, etc. In future work, we will study the performance of the weighted voting memory when using other types of features. In addition, we will look into ways of making the computation more efficient.

References

[1] Mu, X., Watta, P., and Hassoun M. (2004). "A Weighted Voting Model of Associative Memory: Theoretical Analysis," submitted for publication in the *Proceedings of IJCNN 2005*, Montreal, Canada.

[2] Ikeda, N., Watta, P., Artiklar, M., and Hassoun, M. (2001). "Generalizations of the Hamming Net for High Performance Associate Memory," *Neural Networks*, 14(9), 1189-1200.

[3] Phillips, P., Moon, H., Rizvi, S., and Rauss, P. (2000). "The FERET Evaluation Methodology for Face-Recognition Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(10), 1090-1104.

[4] Mu, X., (2004). "Automated Face Recognition: A Weighted Voting Method," *Ph.D. Dissertation*, Dept. Electrical and Computer Engineering, Wayne State University, Detroit, MI, 48202.

[5] Wiskott, L., Fellous, J., Kruger, N., and von der Malsburg, C. (1997). "Face recognition by Elastic Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, **17**(7), 775-779.