

Designs of Second-Order Associated Memory Networks with Threshold Logics: Winner-Take-All and Selective Voting

Toshiro Kubota

Department of Mathematical Sciences, Susquehanna University, Selinsgrove PA, USA

Email: kubota@susqu.edu

Abstract—The capacity of an order- d associative memory model is $O(N^d/\log N)$ where N is the memory size in bit. In contrast, the capacity of the Hopfield network is limited to $O(N/\log N)$. Among higher order associative memory models ($d > 1$), the second order memory ($d = 2$) has attractive properties: a relatively small implementation cost of $O(N^2)$, a small number of spurious states, and the presence of a diagonalization form. Due to these properties, it is of both practical and scientific interests to investigate efficient computational mechanisms of such network.

One disadvantage of higher order associative memory is that it cannot be implemented with simple threshold neurons or McCulloch-Pitts neurons, thus a direct implementation of its computational mechanism on a biological substrate is questionable and its silicon implementation is expensive. In this paper, we propose two approximation models of a second order associative memory using threshold logics. Both are two-layered and employ eigenvalue decomposition of the correlation tensor. The first model uses a winner-take-all mechanism and the second uses a weighted voting by those with significant responses. Architectural-level designs of these memory models are presented. Extensive numerical simulations demonstrate effectiveness of the proposed models in retrieving contents with noisy probe vectors.

Index Terms—neural networks, memory capacity, error correction, architectural level design

I. INTRODUCTION

Associative memory has been consistently an active research area since the works of Kohonen [1], Cohen and Grossberg [2], and Hopfield [3]. Most efforts have been directed to increase the memory capacity, improve the error correction capability, reduce the number of spurious memories, and improve biological feasibility [4] [5] [6] [7]. Advances in this field may reveal important aspects of brain and perception of animals including humans.

Two popular extensions to a Hopfield type network for improving the memory capacity are introduction of self-feedback and relaxation of the synaptic weights to asymmetric ones. In [5], Gorodnichy demonstrated that the size of the attraction basin could be increased with a proper amount of self-feedback in a network trained with the pseudo-inverse learning. Similarly in [8], it was shown

that a small amount of self-feedback could reduce the number of spurious states in Hopfield networks. Moreover, in [7], empirical studies of Hopfield networks with asymmetric additive random weights were conducted. The study showed that the asymmetric weights reduced the number of spurious states while no significant change to the size of attraction basins.

A number of learning algorithms have been suggested to replace the traditional Hebbian and Pseudo inverse learning rules. In [9], the authors proposed a Perceptron type learning algorithm with a goal of maximizing a stability measure of the memory. In [8], additional training vectors were introduced systematically so as to increase the size of the attraction basin with the learning algorithm of [9].

A more drastic departure from the traditional first-order correlation memory has been shown to improve the retrieval performance of the memory. Memory models with higher order correlation have been explored and shown to increase the capacity from $O(N/\log N)$ of the Hopfield model to $O(N^d/\log N)$ where N is the number of neurons in the model and d is the order of the correlation [10] [11] [12]. However, biological feasibility of such systems is questionable as evaluation of a polynomial of order d is required for each memory cell or neuron.

In this paper, we discuss how to arrange the computational mechanism of a higher-order associative memory so that it can be approximated with a collection of simple neuron models. Considering biological feasibility and cost-effective silicon implementation in mind, we impose the following design requirements.

- 1) Each neuron can compute a sum of binary inputs weighted by real numbers.
- 2) Each neuron can perform a generalized threshold operation to the weighted sum to produce a binary output. The operation is defined as

$$\chi(z) = \begin{cases} \Delta_+ & z > \theta \\ \Delta_- & z < -\theta \\ \Delta_0 & \text{otherwise} \end{cases} \quad (1)$$

where Δ_\bullet are either 1, -1 or \pm the current neuron state, and θ is a threshold value.

- 3) A pair of neurons communicate via a single-bit wire.

This paper is based on "Second Order Associative Memory Models with Threshold Logics - Eigen Mode Selections," by T. Kubota, which appeared in the Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), Montreal, Canada, October 2007. © 2007 IEEE.

The model is a slightly generalized version of the McCulloch-Pitts neuron. The threshold operation (1) can accommodate both the original Hopfield model and high-correlation order memories of [10] [11]. To specify the threshold operation, we use symbols $\{+, -, \circ, \uparrow\}$ to describe specific Δ_{\bullet} . The symbols indicate 1, -1, x and $-x$, respectively, with x being the state of the neuron. Furthermore, we compactly write $\chi(z; \theta, \Delta_-, \Delta_+, \Delta_0)$ to fully describe the threshold operation.

When these computational requirements are met, the system can be constructed with a set of simple processing elements that are often considered biologically feasible [13] [14] and amiable to implementation on silicon. Since each input is bipolar (i.e. the value is ± 1), the weighted sum can be implemented with adders. Thus, no multipliers are required in the design. Furthermore, the communication or wiring cost is small and scalable as data transmitted between processing elements are strictly binary. The main objective of this work is in two-fold: to present approximation models of a second-order associative memory and to design architectural level circuits for the approximation models.

In this paper, we consider the following scenario. The memory consists of N memory cells. During a storage phase, the memory is presented with K randomly chosen bipolar vectors of N dimensions (i.e. each element is ± 1). We call them *fundamental vectors*. We consider that the set of fundamental vectors is given prior to a retrieval phase and remains unchanged. During the retrieval phase, the memory is initially set to a bipolar vector of length N , which we call *probe vector*, and the memory performs internal processing and settles to another bipolar vector of length N , which is the memory content retrieved by the probe vector. We only consider auto-association memory, therefore the retrieval is considered correct if the retrieved vector is one of fundamental vectors whose Hamming distance is the smallest to the probe vector.

We use the following notations. Matrices and tensors are written in upper bold letters (\mathbf{C} , for example). Vectors are written in lower bold letters (\mathbf{v}). Scalars are written in non-bold italic letters (x and N). Sub-scripts are used to index elements of a countable set. For example, fundamental vectors are addresses as $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K$. They are also used to index components in vectors, matrices and tensors. A set index precedes a component index. Therefore, j th component of k th fundamental vector is addressed as u_{kj} . Super-scripts are used to specify particular memory cell. We use \mathbf{x} for the state of the memory. Therefore, x^i denotes the bipolar state of the i th memory cell. We use (\mathbf{x}, \mathbf{y}) to denote a dot product of two vectors, \mathbf{x} and \mathbf{y} . Memory models run iteratively and we call each iteration *epoch*. To avoid ambiguity, we may attach the epoch number to the memory state and other associated variables. In the case, a number enclosed in $[]$ indicates the index of the epoch. For example, $x^k[t]$ is the state of k th memory at t th epoch. When $\mathbf{x}[t] = \mathbf{x}[t + 1]$, the memory is said to be *stable*, and otherwise, it is said to be *unstable*.

The memory carry various signals internally, which can be either *bit* or *multi-bit*. A bit signal can be transmitted by a single-bit wire. A multi-bit signal requires a multi-bit bus to be transmitted. A bit number can be either binary or bipolar where the former carries 0 or 1 and the latter carries 1 or -1. When a bit signal is transmitted via a single wire, we need to keep in mind the semantic of the signal: binary or bipolar. The distinction between the two becomes important in designing digital circuits of the memory in Section V.

In our experiments (Section IV), we impose random noise to a probe vector. The noise level is indicated by $Q \in [0, 1.0]$. To add noise, we select $\lfloor QN \rfloor$ bits in the probe vector randomly and flip their states. Finally, we call a correlation memory of order d , d -order memory. In particular, we call 2-order and 3-order memories *second order memory* and *third order memory*, respectively.

The paper is organized as follows. Section 2 provides background materials and brief accounts of other relevant associative memory models. Section 3 describes proposed memory models. Section 4 provides results of numerical experiments. Section 5 provides designs of the proposed memory models and compare them against those of relevant models. Finally, Section 6 provides summary and remarks. An earlier and shorter version of this paper appeared in [15].

II. BACKGROUNDS

A. Correlation Memory Models

The original Hopfield network is a fully connected recurrent one and each neuron stores one bit of information. In this sense, neurons and memory cells are equivalent. It obeys the following update rule.

$$z^i[t] = (\mathbf{w}^i, \mathbf{x}[t]), \tag{2}$$

$$x^i[t + 1] = \chi(z^i[t]; 0, +, -, +). \tag{3}$$

We call z^i the action potential of an i th neuron. The synaptic weights, \mathbf{w}^i , are computed with Hebbian rule without any self-feedbacks. Thus,

$$w_j^i = \sum_{k=1}^K u_{ki}u_{kj} \tag{4}$$

for $i \neq j$ and $w_i^i = 0$.

A d -order memory uses a d -tensor, \mathbf{C}^i , computed with an extension of the above Hebbian rule.

$$\mathbf{C}^i = \sum_{k=1}^K u_{ki}\mathbf{u}_k \times \mathbf{u}_k \times \dots \times \mathbf{u}_k \tag{5}$$

where \times denotes the cartesian product, which are repeated for $d - 1$ times. Typically, any self-feedbacks are set to zero. For example with $d = 2$, i th row, i th column, and diagonal elements are set to zero. In this paper, it is understood that all self-feedbacks are removed from correlation tensors. Hence, the degree of freedom associated

with a correlation tensor is $\binom{n}{d}$. The update of d -order memory is

$$z^i[t] = \mathbf{C}^i \cdot (\mathbf{x}[t] \times \dots \times \mathbf{x}[t]) \quad (6)$$

$$x^i[t+1] = \chi(z^i[t]; \theta, +, -, \uparrow). \quad (7)$$

where the cartesian products in (6) are repeated $d-1$ times and \cdot is an inner product of d and $d-1$ tensors as defined in [16]. When C^i is defined as (5), (6) can be written as

$$z^i[t] = \sum_{k=1}^K u_{ki} (\mathbf{u}_k, \mathbf{x}[t-1])^d \quad (8)$$

The threshold value, θ , is set to 0 for d even and $O(K)$ for d odd [10]. It has been shown that the capacity of an d -order memory is $O(N^d / \log N)$ patterns of length N and the radius of the attraction basin approaches $N/2$ as N approaches ∞ .

The size of a correlation tensor is $O(N^d)$, which reflects the storage requirement imposed on each neuron. This exponential growth of the storage requirement makes a network with a large d prohibitively expensive. Therefore, in practice, we are often limited to $d \leq 3$.

Two vectors (say \mathbf{x} and $\hat{\mathbf{x}}$) are of opposite polarities if $x^i = -\hat{x}^i, \forall i$. It is well known that the Hopfield network cannot distinguish two vectors of opposite polarities [17]. In other words, if \mathbf{x} is a stable/unstable state of the network, then $-\mathbf{x}$ is also stable/unstable. This behavior can be easily observed from the update rule of (2) and (3); Note first that the network is stable if and only if $x^i z^i > \theta, \forall i$. When the sign of every component of \mathbf{x} is flipped, then every z^i will change its sign without changing its magnitude. Therefore, the condition of $x^i z^i > \theta$ being true or false is not affected by the polarity of \mathbf{x} . If \mathbf{x} is stable, then so is $-\mathbf{x}$. If \mathbf{x} is unstable, then so is $-\mathbf{x}$. When \mathbf{x} is one of fundamental vector but $-\mathbf{x}$ is not, then we consider $-\mathbf{x}$ *spurious* as it is stable yet is not one being stored in the memory.

In reference to (8), we can observe that the same behavior holds for a network with an odd d but not for one with an even d . Due to this behavior, the memory recall rate of an even order network tends to be less sensitive to noise imposed on a probe vector. Figure 1 shows the memory recall rates of second order and third order memories with various K 's and Q 's. The size of the network (N) is set to 100. K changes from 10 to 1000. Four levels of Q are tested: 0, 0.1, 0.3, and 0.5. The third order memory shows a larger capacity than the second order one. However the recall rate of the third order memory decreases quickly as the noise level increases. The recall rate of the second order memory is less sensitive to the noise level. The results agree with the above observation.

Due to the aforementioned practical feasibility and good error correction capability, we focus our investigation to a second-order correlation network in this paper. The correlation tensor in this case is an $N \times N$

symmetrical matrix computed by

$$\mathbf{C}^i = \sum_{k=1}^K u_{ki} \mathbf{u}_k \mathbf{u}_k^T. \quad (9)$$

where T is a transpose operator. The update of (6) becomes

$$z^i[t] = \mathbf{C}^i \cdot (\mathbf{x}[t] \times \mathbf{x}[t]) = (\mathbf{C}^i \mathbf{x}[t], \mathbf{x}[t]). \quad (10)$$

Since \mathbf{C}^i is symmetrical, it can be written as

$$\mathbf{C}^i = \sum_{k=1}^R \lambda_k^i \mathbf{v}_k^i (\mathbf{v}_k^i)^T \quad (11)$$

where R is the rank of \mathbf{C}^i , λ_k is an eigenvalue of \mathbf{C}^i , and \mathbf{v}_k is the corresponding ortho-normal eigenvector. This decomposition plays a critical role in developing our network models in the next section.

With (11), (10) can be expressed as

$$z^i[t] = \sum_{k=1}^R \lambda_k^i (\mathbf{v}_k^i, \mathbf{x}[t])^2. \quad (12)$$

In this paper, we call each eigenvector an *eigen-mode* of the second order memory, the sign of λ_k^i the *sign* of the k th eigen-mode of i th neuron, and $\lambda_k^i (\mathbf{v}_k^i, \mathbf{x})$ the *response* of \mathbf{x} to the k th eigen-mode. We further define

$$\phi_k^i(\mathbf{x}) = (\mathbf{v}_k^i, \mathbf{x}) \quad (13)$$

and call it the *unit response* of \mathbf{x} to the k th eigen-mode of the i th neuron.

B. Sparse Distributed Memory

Another important work on associative memory that has a similar implementation structure with second-order memory is Sparse Distributed Memory (SDM) [18].

SDM is comprised of pair-wise address and content blocks where each address holds one content. The address block contains a large collection of randomly assigned bipolar vectors whose length is N for auto-associative memory. For each address, the content block stores a vector of length N . The content vectors are initially set to zero. When a fundamental vector is presented to the auto-associative memory for storage, those addresses whose Hamming distances to the random vector are less than a pre-determined constant (*select radius*) are activated and the bipolar pattern is accumulated to the content block of the activated addresses. When a noisy probe vector is presented to the memory for retrieval, those addresses whose Hamming distances to the vector are less than the select radius are activated, and their contents are accumulated into the output buffer. A retrieved pattern is the accumulated pattern thresholded at zero. The goal of SDN is to model human cortical memories.

A critical parameter of the SDM is the select radius. Kanerva suggests in [19] that the radius be set so that the signal to noise ratio of the memory is maximized. He then go on to suggest the following formula for computing the value.

$$\rho = \Gamma_{N,0.5}^{-1} ((2NK)^{-1/3}) \quad (14)$$

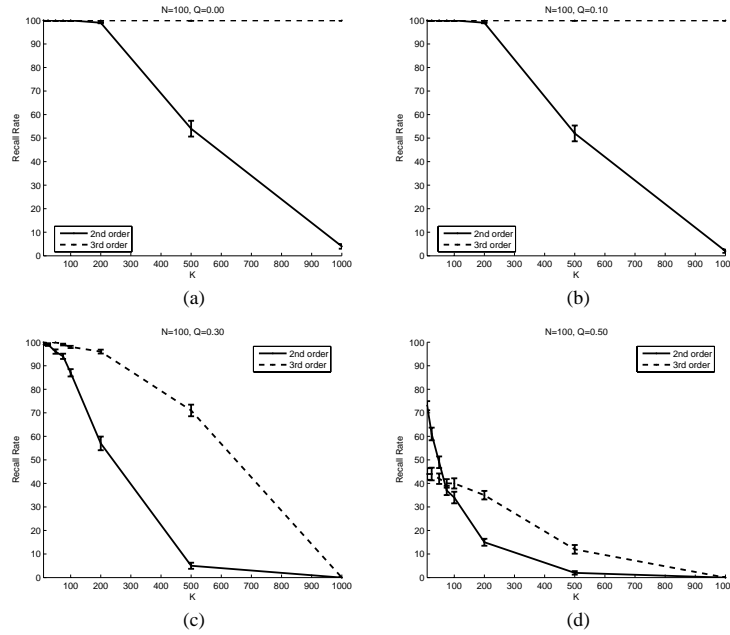


Figure 1. Comparing memory recall rates between second order and third order memories.

where $\Gamma_{n,p}$ denotes the cumulative distribution of a binomial distribution with n trials with p probability of success. In other words,

$$\Gamma_{n,p}(k) = \sum_{j=0}^k \binom{n}{j} p^j (1-p)^{n-j} \quad (15)$$

Since the binomial distribution is a discrete one, ρ is the number of successes whose cumulative probability according to $\Gamma_{N,0.5}$ is closest to $(2NK)^{-1/3}$.

III. MEMORY MODELS

We are interested in implementing a second order associative memory using a set of generalized threshold logics as defined in Section I. Note that the Hopfield network is implementable with the neuron models, while the second order memory is not. The latter involves multiplying two multi-bit numbers, which violates the imposed computational requirements.

For now, let's relax the requirements and allow neurons to multiply two multi-bit numbers and to transmit multi-bit data. Then, Equation (12) suggests a two-layered design for each one bit portion of the second-order memory. The first layer consists of R neurons. Each is assigned a distinct eigen-mode of (12), and computes the unit response of its eigen-mode to the current state, $\mathbf{x}[t]$. The second layer consists of a single neuron that computes the square of each incoming input, computes the sum of the first layer responses weighted by respective eigenvalues of the eigen-modes, and then takes a threshold of the result to produce a bipolar state. The result of the second layer neuron becomes the state of the bit. This is exactly the result of carrying out (12).

Figure 2 illustrates an overview of this two-layered network for a single bit memory. The first layer neurons

(Φ^i) compute

$$y^i = \phi_k^i(\mathbf{x}) = \sum_k (\mathbf{v}_k^i, \mathbf{x}). \quad (16)$$

The second layer neuron (Ψ^i) computes

$$z^i = \sum_k \lambda_k^i (\phi_k^i(\mathbf{x}))^2 = \sum_k \lambda_k^i (y_k^i)^2, \quad (17)$$

$$x^i = \chi(z^i; 0, +, -, \uparrow). \quad (18)$$

This model violates the requirements at three places: to take the square of a multi-bit number (y_k^i), to multiply two multi-bit numbers ($(y_k^i)^2$ and λ_k^i) and to transmit a multi-bit number ($y_k^i = \phi_k^i(\mathbf{x})$) from the first layer to the second layer. Despite the violations, the model does suggest a computational strategy for carrying out (12) by a collection of neurons with each computing weighted sum of inputs. Thus, the model is more amiable to the first computation requirement given in Section I.

In what follows, we introduce two approximation models that alleviate these violations and can be implemented on the two layer network of Figure 2. In regard to the two layer network, we introduce the following terminology. We say a first layer neurons is *active* if its unit response exceeds the threshold level of the neuron, and is *inactive* otherwise. Furthermore, when the neuron is active, then its eigen-mode is considered *significant* with respect to the current memory state.

The two-layered network of Figure 2 will also serve as the basis for memory implementations that will be detailed in Section V.

A. Model 1 - Winner-Take-All

Our first model is based on the following conjecture: it is likely that the sign of the most strongly responding

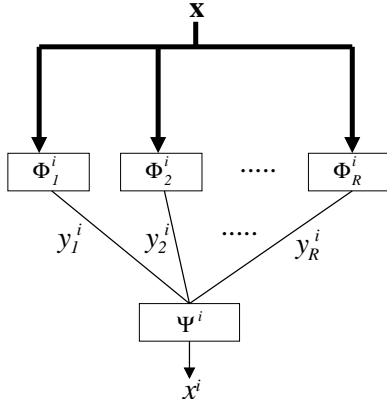


Figure 2. A two-layer architecture of the second order network model.

eigen-mode coincides with the sign of z^i in (12). Thus,

$$sgn \left(\sum_k \lambda_k (\mathbf{v}_k^i, \mathbf{x})^2 \right) \approx sgn (\lambda_{\tilde{k}}^i), \quad (19)$$

where

$$\tilde{k} = \arg \max_k \left| \sqrt{|\lambda_k^i|} (\mathbf{v}_k^i, \mathbf{x}) \right|. \quad (20)$$

Equations (19) and (20) suggest the following computational scheme using the same network architecture shown in Figure 2; The j th neuron in the first layer for i th neuron computes

$$y_j^i[t] = \chi(\phi_j^i(\mathbf{x}[t]); \theta, +, +, \uparrow), \quad (21)$$

and the second layer neuron computes

$$z^i[t] = \sum_{j=1}^R \sqrt{|\lambda_j^i|} y_j^i[t] \quad (22)$$

$$x^i[t+1] = \chi(z^i[t]; 0, +, -, \uparrow) \quad (23)$$

with θ being determined by the second layer neuron as described next. We call this approximation model the *winner-take-all* model of the second-order memory or WTA for short.

We need to provide a mechanism such that only one neuron in the first layer survives the threshold operation. We can implement such competition in different ways, and such competitive (winner-take-all) behavior has been observed in biological systems as well [20] [21]. To comply with our computation requirements, we implement the competition as a binary search of an optimum threshold value of θ . The search is controlled by a feedback signal from the second layer to the first layer in the following manner.

Let β denote the feedback signal, which is binary. When more than one neurons in the first layer are active, β is set to 1, while when no neurons are active, β is set to 0. Then θ obeys the following rule.

$$\theta = \begin{cases} \theta + \delta & \beta = 1 \\ \theta - \delta & \beta = 0 \end{cases}, \quad (24)$$

$$\delta = \delta/2. \quad (25)$$

At the beginning of the search, θ and δ are initialized to

$$\theta = \frac{\sum_j |v_{kj}^i| \sqrt{|\lambda_k^i|}}{2}, \quad \delta = \theta^0/2. \quad (26)$$

This binary search will quickly find an optimal threshold with which only one first layer neuron is active.

B. Model 2 - Weighted Significant Voting

As K increases, \mathbf{C}^i approaches full-ranked and its spectrum (or the set of eigenvalues) approaches uniform. As a result, the approximation of (19) using a single eigen-mode becomes insufficient. Note that

$$\sum_j [\phi_j^i(\mathbf{x})]^2 = N \quad (27)$$

where \mathbf{x} is any bipolar vector. This is because the length of \mathbf{x} does not change by rotation of the coordinate system. Thus, unit responses, $\{\phi_j^i(\mathbf{x})\}_i$, are not independent with one another.

If a probe vector is responding strongly to one mode, say k , $\phi_k^i(\mathbf{x})^2 \approx N$ and $\phi_j^i(\mathbf{x})^2 \approx 0, \forall j \neq k$. This is the case assumed by WTA model. When either K or Q grows large, a probe vector tends to respond strongly to more than one eigen-modes. Assume M eigen-modes are responding strongly with their unit responses being more than θ . Then, due to (27), their unit responses are constrained by $\theta < \phi_k^i < \sqrt{N-(M-1)\theta^2}$. As M increases, the permitted range of ϕ_k^i becomes smaller and smaller. Consequently, ϕ_k^i shares similar values and

$$\sum_j \lambda_j^i (\phi_j^i)^2 \propto \sum_{j \in \Omega(\mathbf{x})} \lambda_j^i. \quad (28)$$

where $\Omega(\mathbf{x})$ is a set of eigen-modes responding significantly to a given \mathbf{x} . This observation leads to our second model, which is to approximate the second-order memory with

$$x^i[t+1] \approx sgn \left(\sum_{j \in \Omega(\mathbf{x}[t])} \lambda_j^i \right) \quad (29)$$

The approximation can be implemented with the architecture of Figure 2 using the following rules. The first layer neurons compute

$$y_j^i[t] = \chi(\phi_j^i(\mathbf{x}[t]); \theta, +, +, -), \quad (30)$$

and the second layer neuron computes

$$z^i[t] = \sum_j \lambda_j^i y_j^i[t] + \sum_j \lambda_j^i, \quad (31)$$

$$x^i[t+1] = \chi(z^i[t]; 0, +, -, \uparrow). \quad (32)$$

The bias term $\sum_j \lambda_j^i$ in (31) is to cancel inputs from insignificant first layer neurons. This term is not necessary if y_j^i is interpreted as binary instead of bipolar as seen in Section V. We call this approximation model *weighted significant voting* model of the second order memory or WSV for short.

An important component of WSV model is the threshold value of θ . Note that when N is large, the constraint

of (27) on each ϕ_j^i is weak and ϕ_j^i has approximately a Chi-square distribution of 1 degree of freedom. Then the mean and standard deviation are 1 and 2, respectively, and are independent of N . We can set θ to the mean plus the standard deviation times some positive constant. In this paper, we are concerned with networks of $N \geq 10$, and consider the Chi-square model applicable under such condition.

IV. EXPERIMENTS

In this section, we provide numerical experiments for evaluating the performance of proposed memory models in terms of memory recall rates. A set of K randomly generated bipolar patterns are stored in the memory. To evaluate the performance of memory retrieval, one of the fundamental vectors is selected randomly and noise is added to it to form a probe vector. A recall is considered successful if the retrieved pattern is one of stored patterns that are closest in terms of Hamming distance to the probe vector. Therefore, the Hamming network [22] provides a 100% recall rate and an upper bound of the performance.

To impose noise on a probe vector, we randomly select $\lfloor NQ \rfloor$ bits and flip their values. Thus, when $N = 100$ and $Q = 0.1$, exactly $QN = 10$ bits have their values flipped. Four levels of noise are used in the experiments. They are $Q = 0, 0.1, 0.3, \text{ and } 0.5$.

For each experiment, the recall rate presented is the average of 20 independent experiments, in which 30 probe vectors are used to compute the each recall rate. An accompanying error bar provides a standard error of the mean recall rate estimate.

A. Threshold in Weighted Significant Voting

First, we examine the performance of WSV with respect to a threshold level, θ . We tested 8 levels between 1.5 and 3.25 with 0.25 increments. Results are shown in Figure 3. Neither a single setting performs the best overall nor a simple rule predicting the performance is apparent. However, we make the following observations:

- Results are similar for $1.75 \leq \theta \leq 3$ in all the conditions tested.
- When K/N is small ($K < 2N$), a higher θ in the range of $2.75 \leq \theta \leq 3.0$ performs better.
- When K/N is large ($K \geq 2N$), a lower θ in the range of $1.75 \leq \theta \leq 2.5$ performs better.

We can explain the behavior as follows. When K/N is small, there tend to be a few eigen-modes that are dominant. Since the sum of the squared response is always N , the unit responses of these eigen-modes are relatively high. Thus, we can set θ high to extract them reliably. When K/N is large, a larger number of eigen-modes respond strongly and their unit responses are not as high as when K/N is small. Thus, θ needs to be lowered to include them into the approximation. Based on these observations, we fix $\theta = 2.5$ in the remaining of our experiments. However, this choice does not appear to be critical.

B. Comparisons with randomized experiments

Second, the number of neurons in the network (N) is set to 100 and the number of fundamental vectors (K) is varied from 10 to 200. For comparisons, we implemented the Hopfield network, second order memory and SDM, and tested them under the same condition with our approximation models. The number of addresses in SDM is set to N^2 to make the storage requirements comparable to our models. The select radius of SDM is set using the formula described in Section II-B.

The results are shown in Figure 4. Due to its limited capacity, the performance of the Hopfield network is low compared to others. SDM performs well when no noise is present to a probe vector; The recall rate is the highest when $N \leq 50$ and comparable to that of the second order memory for $N > 50$. However, it does not retain the performance as Q increases. The performance of WSV is consistently better than that of the WTA, except for $Q = 0.5$ and $K < 50$. The performance of WSV model is highly comparable to the second order model for $K \leq N$. Although the second order model retains high recall rates as K becomes more than N , both WTA and WSV fail to do so. The results suggest that the approximation models are capable of storing patterns as many as N .

Next, K is fixed to 100, and N is varied from 10 to 100. The results are shown in Figure 5. Observations similar to ones for Figure 4 can be made. When there is no noise in a probe vector, SDM has the highest recall rate of close to 100%. The performance degrades quickly as noise is being introduced. Recall rates of the Hopfield network are consistently low due to its limited capacity. The performance of WSV is consistently better than that of WTA, except for $N < 50$. The performance of WSV is comparable with the second order model when $Q \leq 0.1$ and $N \approx K$.

Some descriptive statistics of mean recall rates normalized by corresponding mean recall rates of the second order memory are shown in Table I. The normalized mean recall rate is 1.0 if the mean recall rate is the same with that of the second order memory and less than 1 if it is less than that of the second order memory. Shown in the table are the mean, median, first quartile (1st Q), third quartile (3rd Q), and standard deviation (Std). The mean of SDM is deceptively high because of a few entries that are more than 3. Overall, the proposed approximation models are most comparable to the performance of the second order memory. The results also suggest better and more stable performance of the weighted significant voting than the winner-take-all model.

V. ARCHITECTURE

In this section, we look at our second objective: to introduce architectural level designs of the memory models considered in Section IV. Rough designs of the models have been described in Sections II and III. We will make them more precise so that translation of the algorithms to hardware become straightforward.

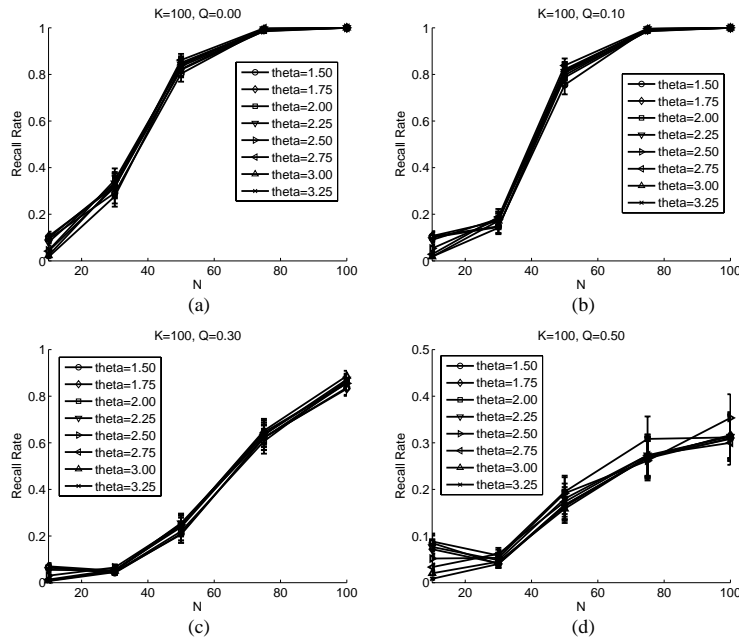


Figure 3. Comparing memory recall rates of the Weighted Significant Voting with various θ s.

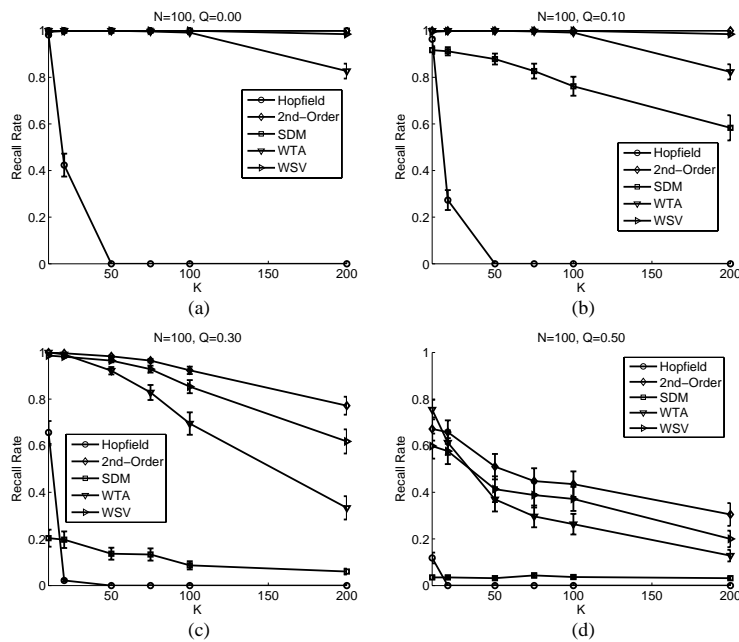


Figure 4. Comparing memory recall rates with $N = 100$ and various K .

TABLE I.
DESCRIPTIVE STATISTICS OF THE NORMALIZED MEAN RECALL RATE.

Method	Mean	Median	1st Q	3rd Q	Std
Hopfield	0.09	0.00	0.00	0.00	0.226
SDM	0.834	0.686	0.138	1.00	0.870
WTA	0.751	0.782	0.550	0.992	0.238
WSV	0.764	0.850	0.531	0.987	0.232

One subtle but critical issue to the designs is the semantic of a bit signal. We want to allocate a single bit to encode the state of a single memory cell (x_j^i), which

is bipolar (± 1). However, most conventional computing platforms use the binary representation (0/1). To utilize the rich existing computing conventions, therefore, the bipolar signals need to be converted into the binary representations internally. Fortunately, all designs considered in this section interface the state vector (\mathbf{x}) with either bit-word or bit-bit multiply-accumulator (MAC), and we can build-in the bipolar-binary conversion into the MAC with little overhead. More details of MAC designs are given below.

The following conventions are used throughout in illustrations. A bit signal is shown by a gray arrow. When the

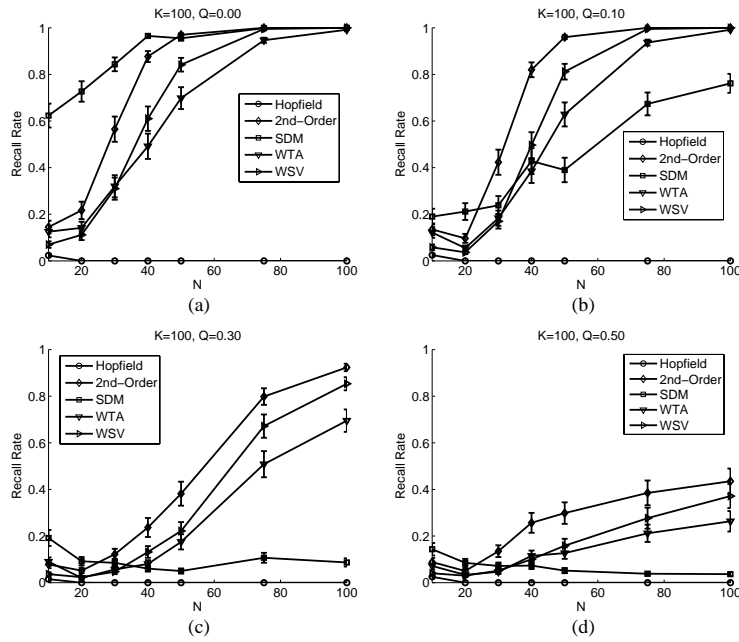


Figure 5. Comparing memory recall rates with $K = 100$ and various N .

signal is interpreted as bipolar, the arrow accompanies a small line segment perpendicular to the arrow. A multi-bit signal is shown by a black arrow. A vector signal carrying number of either bit or multi-bit signals is shown by a thick arrow. The shade of the thick arrow is dependent on the type of the signal. A bit vector signal is in gray, a multi-bit vector signal is in black. For our discussion, we assume that eigenvalues and eigenvectors of the decomposition (11) are represented with L -bits multi-bit signals. Figure 6 shows the different kinds of arrows used in this section.

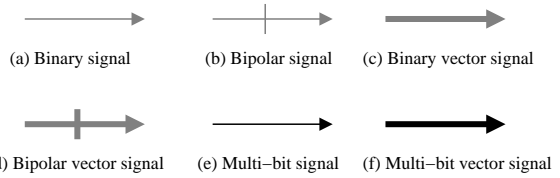


Figure 6. Different signal types and corresponding arrow representations.

The designs deploy a number of multiply-accumulate (*MAC*) units which computes an inner product of two vectors. Figure 7 shows five different types of MAC units used in our designs. Each type is shown its block representation (left) and architectural-level design (right). Figure 7(a) shows a binary-binary MAC, where both inputs are binary vector signals. It can be implemented with N multiplexers and one N -input accumulator (shown as N-AC in the figure). The accumulator can be built with $N - 1$ adders of at most $\log_2 N$ bits. It does not require any multipliers, thus can be implemented inexpensively. Figure 7(b) shows a bipolar-bipolar MAC where both inputs are bipolar vector signals. As for the binary-binary MAC, it can be implemented with N multiplexers and one N -input accumulator. However, in this case, the output of the accumulator needs to be multiplied by 2 and then subtracted by N . This arithmetic implements a binary-bipolar conversion.

Figure 7(c) shows a word-binary MAC, which takes one multi-bit vector and one binary vector. Like the binary-binary MAC, it requires N multiplexers and an N -input accumulator, but does not require multipliers. The N -input accumulator is more expensive than that of Figure 7(a) and (b) as additions involve at most $L \log_2 N$

bits.

Figure 7(d) shows a word-bipolar MAC, which takes one multi-bit vector and one bipolar vector. It is similar to the word-binary MAC of Figure 7(c) except that the inputs to multiplexers are the multi-bit input and its negative. For a sign-magnitude format, computing the negative of a multi-bit number is as simple as to invert the sign bit. For a 2's complement number, it is to take the 2's complement. When the multi-bit vector is fixed, we can employ an alternative approach, which can reduce the implementation cost for a large N . This alternate scheme is shown in Figure 7(e). Instead of computing the negative of each multi-bit number, it converts a word-binary MAC to a word-bipolar MAC using a similar strategy with the bipolar-bipolar MAC of Figure 7(b). In our designs, we employ this alternative approach in implementing word-bipolar MACs.

Finally, Figure 7(d) shows a word-word MAC, which takes two multi-bit vectors as inputs. It requires N L -bit multipliers and one N -input $2L \log_2 N$ accumulator. Therefore, it is considerably more expensive than the other two.

A. Second-Order Model

First model to be considered is the second order correlation memory implemented with the eigen decomposition

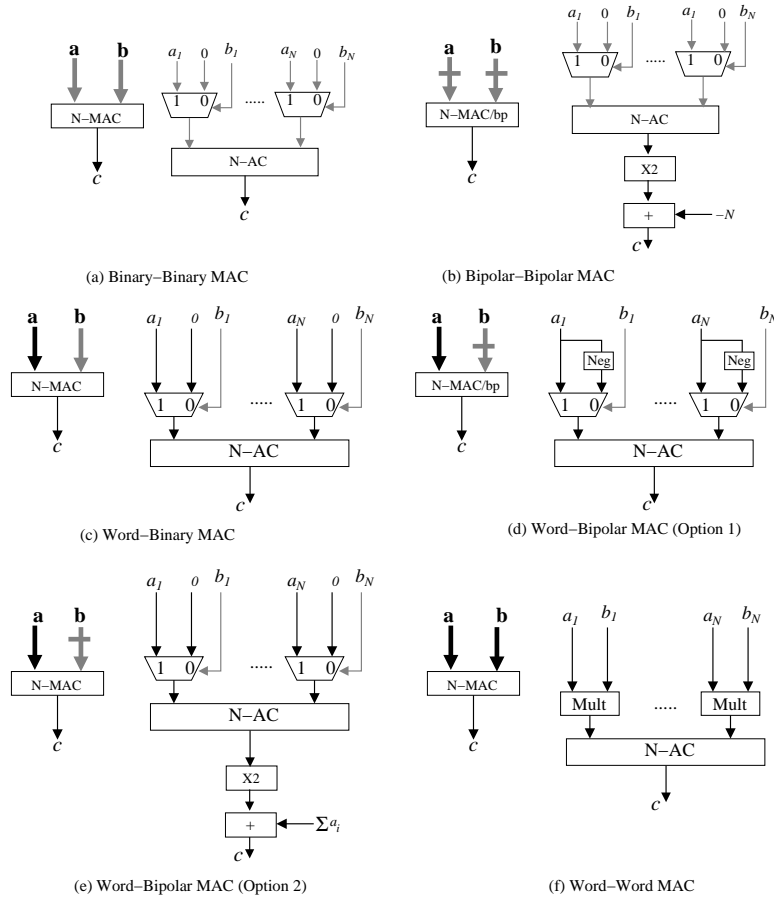


Figure 7. Various types of multiply-accumulate units used in the memory designs.

of Equation (12). Figure 2 shows its architectural-level design of the i th memory cell.

The j th neuron (Φ_j^i) in the first layer computes a dot-product between \mathbf{x} and \mathbf{v}_j^i to compute the unit response. The output of Φ_j^i is delivered to the second layer neuron via an L -bit bus. The second layer neuron (Ψ^i) first squares each input and takes a dot-product between the squared inputs and $\lambda^i = [\lambda_1^i, \lambda_2^i, \dots, \lambda_R^i]$.

B. Sparse Distributive Memory Model

Next model to be considered is the Sparse Distributive Memory (SDM). Figure 9 shows its architectural-level design of the i th memory cell. The architecture departs slightly from the two-layered one shown in Figure 2. It requires an additional layer at the very front as indicated with a set of shaded blocks (η .s). We call this layer the address layer, as it implements the address block of the memory. The address layer is shared among all memory cells in the memory. Each neuron in the address layer computes a dot-product between \mathbf{x} and a random vector (\mathbf{r}_j) associated with the neuron. The result of the dot-product is thresholded with the select radius (ρ) to produce a binary signal ξ_j , which is used to activate the corresponding entry in the content block. See Section II-B for details of the address and content blocks. Note that there are $N \times N$ neurons in the content block as determined in Section IV.

The j th neuron in the first layer (Φ_j^i) after the address layer contains a single multiplexer which selects either 0 or the i th entry of the content memory at the j location (a_j^i). It requires at most $\log_2 2K$ bits to store a_j^i as a_j^i can take values between $[-K, K]$. Therefore, we consider the output of Φ_j^i to be $\log_2 2K$ bit long. When $\xi_j = 0$, 0 is selected, and when $\xi_j = 1$, a_j^i is selected. The output from $N \times N$ neurons in the first layer is collected in the second layer neuron, Ψ^i . It needs to calculate the sum of the $N \times N$ numbers, each being $\log_2 2K$ bits long. This process requires $N \times N - 1$ adders of at most $2 \log_2 N \log_2 2K$ bits. The result of the additions is thresholded at 0 to produce the next i th state of the memory.

Costly parts of the design are $\log_2 K$ -bit data-paths connecting the first layer to the second layer, and the $N \times N$ -MAC in Ψ^i .

C. Winner-Take-All Model

We now consider proposed models to approximate the second order correlation memory. First, we consider WTA. Figure 10 shows an architectural-level design of the model. It obeys the two-layer model of Figure 2 except two one-bit feedback signals (β and γ) from the second to the first layer.

The first layer neurons (Φ_j^i) computes a dot-product between $\sqrt{|\lambda_j^i|} \mathbf{v}_j^i$ and \mathbf{x} . The result is then thresholded at

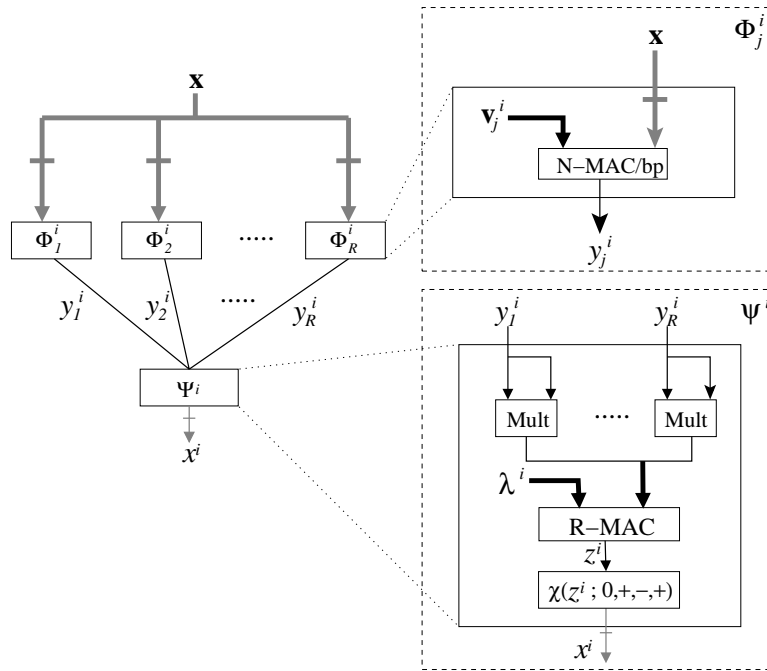


Figure 8. Architectural-level design of the second order correlation memory.

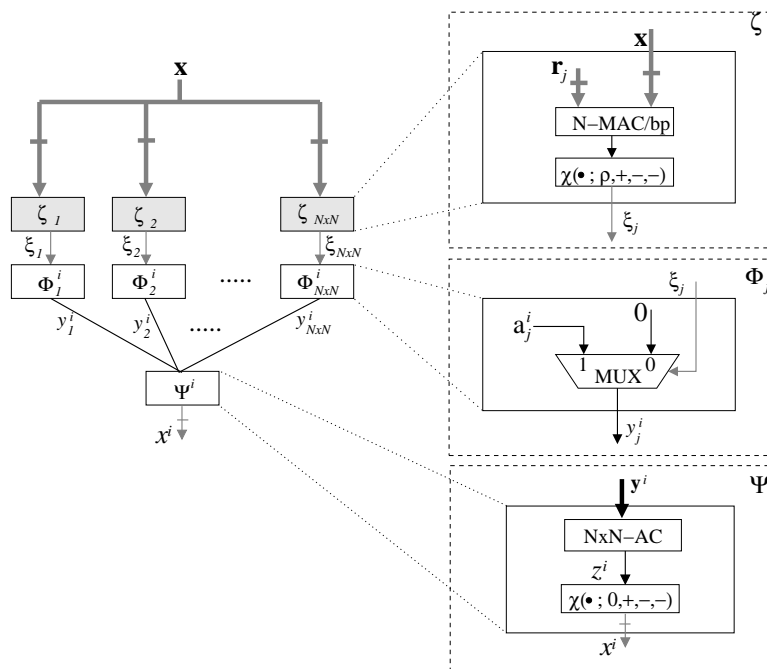


Figure 9. Architectural-level design of the sparse distributed memory.

θ whose value is determined in the θ -Adj block shown in Figure 10. When the result of the dot-product is more than θ or less than $-\theta$, then the neuron outputs 1. Otherwise it outputs 0. The θ -Adj block implements the following equation.

$$\theta = \begin{cases} \theta = \theta + \delta & \text{if } \beta = 1 \text{ and } \gamma = 0 \\ \theta = \theta - \delta & \text{if } \beta = 0 \text{ and } \gamma = 0 \\ \theta = \theta_0 & \text{if } \gamma = 1 \end{cases}, \quad (33)$$

$$\delta = \begin{cases} \delta/2 & \text{if } \gamma = 0 \\ \theta_0/2 & \text{if } \gamma = 1 \end{cases}. \quad (34)$$

Thus, the block can be implemented with an adder and either a shift register or an adder for computing the division by 2. Note that θ_0 is an initial threshold value, which should be set to $\sum_k |v_{jk}^i| \sqrt{|\lambda_j^i|}$, the maximum possible value of the dot-product.

The second layer neuron takes a binary vector of R -bits as the input. The main goal of the neuron is to provide β^i until only a single bit in the R -bit input vector is 1, and when this condition is achieved, set γ to 1. To do so, the sum of the R -bits in the input vector is computed in

an R-bit accumulator (shown as R-AC in the figure). The accumulator requires $N - 1$ adders that are at most $\log_2 R$ bits long. Denote the result of the accumulator c^i . Then, c^i is passed to both β -Gen block and a comparator ($= 1$?). The β -Gen block basically implements the following equation.

$$\beta^i = \begin{cases} 0 & \text{if } c^i = 0 \\ 1 & \text{if } c^i > 0 \end{cases} . \quad (35)$$

The comparator implements the following equation.

$$\gamma^i = \begin{cases} 0 & \text{if } c^i \neq 1 \\ 1 & \text{if } c^i = 1 \end{cases} . \quad (36)$$

Thus, $\gamma^1 = 1$ signals that a winner (\tilde{k} in Eq. (20)) of the competition is found. The signal is then used to latch the sign of $\lambda_{\tilde{k}}^i$, which is computed by a dot-product between \mathbf{y}^i and a vector whose k th entry is the sign of λ_k^i .

The second layer neuron requires, therefore, two comparators (one for detecting $c^i = 0$ and the other detecting $c^i = 1$), one R-AC whose adders are at most $\log_2 R$ bits, one binary-binary R-MAC, and a latch. Communication between first layer and second layer neurons involve only bit signals.

D. Weighted Significant Voting Model

Finally, we consider WSV. It obeys the two-layer network of Figure 2. There are R neurons in the first layer. The j th first layer neuron (Φ_j^i) computes a unit response of \mathbf{v}_j^i to \mathbf{x} using a word-binary N-MAC. The result is thresholded at a pre-defined threshold value of θ . When the unit response is more than θ or less than $-\theta$, the neuron outputs 1. Otherwise, it outputs 0. The neuron can be completely implemented with one word-binary N-MAC and one threshold logic.

The second layer neuron (Ψ^i) first computes a dot-product between the input binary vector and $\boldsymbol{\lambda}^i = [\lambda_1^i, \lambda_2^i, \dots, \lambda_R^i]$. The result is then thresholded at 0. Again, the neuron can be completely implemented with one word-binary R-MAC and one threshold logic. The communication among neurons involves only bit signals.

VI. CONCLUSIONS

The paper introduced two associative memory models that are derived as an approximation to a second order correlation model. Both are constructed from an eigenvalue decomposition of a correlation tensor of the second order model. The winner-take-all model selects the eigen-mode with maximum response, while the weighted significant voting selects those that are deemed significant in terms of the unit response. The memory can be implemented with a collection of simple processing units without multipliers, interconnected by a single wire per input/output to transmit its state.

Our numerical experiments suggest better and more stable performance of the weighted significant voting than the winner-take-all model. The design of the weighted significant voting is simpler than the winner-take-all model, as no feedbacks are necessary. Thus, we can conclude

that the weighted significant voting is preferable to the winner-take-all model for implementing a second order memory.

No theoretical analysis of the approximation models are available at this time. Future works include studying the convergence properties of the models, and deriving the capacity of the models and the size of the attraction basin. With a decent amount of capacity, robustness against noise, and relatively small computational cost, the proposed memory models are valuable alternatives to constructing associative memories.

Another limitation of this study is that it does not consider on-line storage of fundamental vectors. The storage phase is done off-line with a set of fundamental vectors, and eigen-modes are pre-computed and pre-stored in the circuits. For the memory models to become widespread use, they need to be able to update the memory contents with new fundamental vectors with minimum user interactions in a computationally efficient manner.

REFERENCES

- [1] T. Kohonen, *Associative memory*. Berlin: Springer-Verlag, 1977.
- [2] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, no. 5, pp. 815–826, Sept./Oct. 1983.
- [3] J. J. Hopfield, "Neural nets and physical systems with emergent collective computational properties," *Proceedings of the National Academy of Science USA*, 1982.
- [4] V. Gimenez-Martinez, "A modified hopfield auto-associative memory with improved capacity," *IEEE Transactions on Neural Networks*, vol. 11, no. 4, pp. 867–875, 2000.
- [5] D. O. Gorodnichy, "The optimal value of self-connection," in *Proceedings International Joint Conference on Neural Networks*, Washington DC, July 1999.
- [6] O. de Jesus, J. Horn, and M. Hagan, "Analysis of recurrent network training and suggestions for improvements," in *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, July 2001, pp. 2632–2637.
- [7] Z. Chengxiang, C. Dasgupta, and M. P. Singh, "Retrieval properties of a hopfield model with random asymmetric interactions," *Neural Computation*, vol. 12, no. 4, pp. 865–880, 2000.
- [8] D.-L. Lee and T. C. Chuang, "Designing asymmetry hopfield-type associative memory with higher order hamming stability," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1464–1476, November 2005.
- [9] W. Krauth and M. Mezard, "Learning algorithms with optimal stability in neural networks," *Journal of Physics A: Mathematical and General*, vol. 20, pp. L745–L752, 1987.
- [10] S. S. Venkatesh and P. Baldi, "Programmed interactions in higher-order neural networks: The outer-product algorithm," *Journal of Complexity*, vol. 7, no. 4, pp. 443–479, Dec. 1991.
- [11] D. Burshtein, "Long-term attraction in higher order neural networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 42–50, Jan. 1998.
- [12] P. Baldi, "Neural networks, orientations of the hypercube and algebraic threshold functions," *IEEE Transactions on Information Theory*, vol. 34, no. 3, pp. 523–530, 1988.

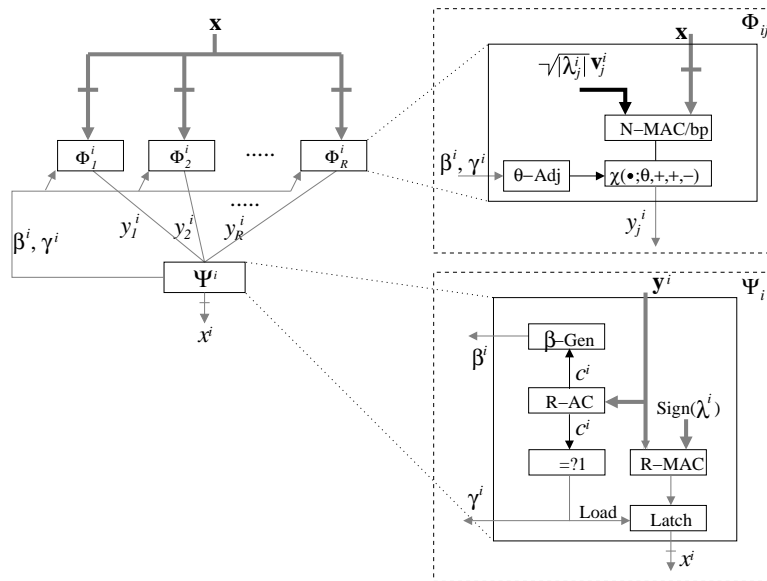


Figure 10. Architectural-level design of the winner-take-all memory.

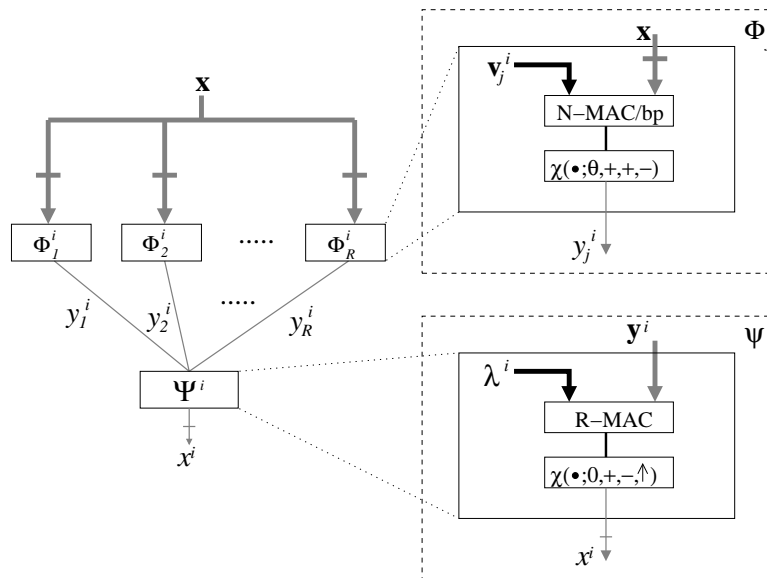


Figure 11. Architectural-level design of the weighted significant voting memory.

[13] U. Bauer, M. Scholz, J. B. Levitt, K. Obermayer, and J. S. Lund, "A biologically-based neural network model for geniculocortical information transfer in the primate visual system," *Vision Research*, vol. 39, no. 613-629, 1999.

[14] C. Koch, *Biophysics of computation. Information Processing in Single Neurons*. New York: Oxford University Press, 1999.

[15] T. Kubota, "Second order associative memory models with threshold logics -eigen mode selections," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Montreal, Quebec, October 2007, pp. 1884-1889.

[16] L. De Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253-1278, 2000.

[17] J. Bruck and V. P. Roychowdhury, "On the number of spurious memories in the hopfield model," *IEEE Transactions on Information Theory*, vol. 36, no. 2, pp. 393-397, 1990.

[18] P. Kanerva, *Sparse Distributed Memory*. MIT Press, 1988.

[19] —, *Associative Neural Memories: Theory and Implementation*. New York: Oxford University Press, 1993, ch. 3, pp. 50-76.

[20] T. Fukai and S. Tanaka, "A simple neural network exhibiting selective activation of neuronal ensembles: From winner-take-all to winners-share-all," *Neural Computation*, vol. 9, no. 1, pp. 77-97, 1997.

[21] Z. H. Mao and S. G. Massaquoi, "Dynamics of winner-take-all competition in recurrent neural networks with lateral inhibition," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 55-69, 2007.

[22] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, Apr. 1987.

Toshiro Kubota received the BS degree in instrumentation engineering from Keio University, Japan, in 1988, and the MS and PhD degrees in electrical engineering from the Georgia Institute of Technology, Atlanta, GA in 1989 and 1995, re-

spectively. From 1996 until 2004, he was at the University of South Carolina, Columbia, SC. From 2004 to 2006, he was at the Siemens Medical Solutions, Malvern, PA. Currently, he is an Assistant Professor in the Department of Mathematical Sciences at the Susquehanna University, Selinsgrove, PA. His research interests include computer vision, image processing, remote sensing, and neural networks. Dr. Kubota is a member of the IEEE, the IEEE Computer Society, the IEEE Signal Processing Society, and the ACM.