Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 95 (2005) 537–544
Information Processing Letters
www.elsevier.com/locate/ipl

Neural associative memory for brain modeling and information retrieval ☆

Andreas Knoblauch [a,b,1]

[a] *Department of Neural Information Processing, University of Ulm, Oberer Eselsberg, D-89069 Ulm, Germany*
[b] *MRC Cognition and Brain Sciences Unit, Speech and Language Group, 15 Chaucer Road, Cambridge CV2 2EF, England*

Received 22 April 2004
Available online 23 June 2005
Communicated by J.L. Fiadeiro

Abstract

This work concisely reviews and unifies the analysis of different variants of neural associative networks consisting of binary neurons and synapses (Willshaw model). We compute storage capacity, fault tolerance, and retrieval efficiency and point out problems of the classical Willshaw model such as limited fault tolerance and restriction to logarithmically sparse random patterns. Then we suggest possible solutions employing spiking neurons, compression of the memory structures, and additional cell layers. Finally, we discuss from a technical perspective whether distributed neural associative memories have any practical advantage over localized storage, e.g., in compressed look-up tables.

Keywords: Spiking associative memory; Neural modeling; Algorithms; Information retrieval; Fault tolerance

1. Introduction

Associative memories are systems that contain information about a finite set of associations between pattern vector pairs $\{(\mathbf{u}^\mu \mapsto \mathbf{v}^\mu)\colon\ \mu = 1,\ldots,M\}$, where $\mathbf{u}^\mu$ and $\mathbf{v}^\mu$ are called *address* and *target* patterns, respectively [16]. Given a possibly noisy address pattern $\tilde{\mathbf{u}}$ the problem is to find a target pattern $\mathbf{v}^\mu$ for which the corresponding address pattern $\mathbf{u}^\mu$ is most similar to $\tilde{\mathbf{u}}$. This is a variant of the *Best Match Problem* in [18] and efficient solutions have widespread applications, e.g., for cluster analysis, speech and object recognition, or information retrieval in large databases [16,23,6]. Interesting performance measures are

(i) *storage capacity* defined as the maximal amount of stored information,

☆ The author is grateful to Günther Palm, Fritz Sommer, Thomas Wennekers, and Friedemann Pulvermüller for valuable comments, discussions, and support. Sponsored by the MirrorBot project of the European Union IST-2001-35282.
E-mail address: andreas.knoblauch@honda-ri.de (A. Knoblauch).
[1] Current address: Honda Research Institute Europe, 63073 Offenbach/Main, Germany.

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2005.05.021

(ii) *retrieval efficiency* defined as the time required for retrieving a pattern, and
(iii) *fault tolerance* defined as the maximal allowed distance between address patterns $\tilde{\mathbf{u}}$ and $\mathbf{u}^\mu$ for retrieving $\mathbf{v}^\mu$.

In *neural implementations* the information about the associations is stored in the synaptic connectivity of one or more neuron populations [28,8,22]. Neural associative memories play an important role in many *brain theories* (e.g., [7,2,21,4]), where the patterns correspond to attractors in the brain's neuronal state space. From the *technical perspective*, neural associative memories can be advantageous over hash-tables or simple look-up-tables if the number of patterns is large, if parallel implementation is possible, or if fault-tolerance is required [20,11,12].

In the following we will review results of a simple binary model, the so-called Willshaw model, which has been used for brain modeling and which appears to be most efficient for technical applications [11,13,28, 20]. We review and slightly extend the classical analysis of storage capacity, fault tolerance, and retrieval efficiency and point out problems such as limited fault tolerance and restriction to logarithmically sparse patterns without any correlations. We will see how the problems can be solved by model variants employing spiking neurons, compression of the memory structures, and additional ("grandmother"-like) cell layers. Finally, we critically discuss from the technical perspective whether distributed neural associative memories have any advantage over localized storage, for example, in compressed look-up-tables.

2. Binary Willshaw associative memory

2.1. Learning patterns

An attractive model of neural associative memory both for biological modeling and applications is the so-called Willshaw or Steinbuch model with binary neurons and synapses [28,26,19,22]. Each pattern is a sparse binary vector of length n containing $k \ll n$ one-entries and $n-k$ zero-entries. The M pattern pairs are stored *hetero-associatively* in a binary *memory matrix* $\mathbf{H} \in \{0,1\}^{n\times n}$, where

$$H_{ij} = \min\left(1, \sum_{\mu=1}^{M} u_i^\mu \cdot v_j^\mu\right) \in \{0,1\}. \tag{1}$$

The *neural interpretation* is that of two neuron populations, an address population u and a target population v, each consisting of n neurons. The patterns $\mathbf{u}^\mu$ and $\mathbf{v}^\mu$ describe the activity states of the two populations at time μ, and H_{ij} is the strength of the Hebbian learned synaptic connection from neuron u_i to neuron v_j. For the auto-associative case $u = v$ (i.e., if address and target populations are identical), the network can be interpreted as an undirected graph with n nodes and edge matrix $\mathbf{H}$ where patterns correspond to cliques of k nodes.

2.2. Recalling patterns: One-step retrieval algorithm

After learning, the stored information can be retrieved applying an address pattern $\tilde{\mathbf{u}}$ containing $z := \sum_{i=1}^{n} \tilde{u}_i$ one-entries. The retrieval result will crucially depend on the retrieval algorithm. The simplest algorithm is *one-step retrieval* where all neurons are updated synchronously (in one step) [28,19,25]: A vector–matrix-multiplication yields the neural potentials $\mathbf{x} = \tilde{\mathbf{u}} \cdot \mathbf{H}$ of the target population, and imposing a threshold Θ gives the retrieval result $\hat{\mathbf{v}}$,

$$\hat{v}_j = \begin{cases} 1 & x_j = (\sum_{i=1}^{n} \tilde{u}_i H_{ij}) \geqslant \Theta, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Choosing $\Theta = z$ will be referred to as the *Willshaw threshold* and plays an important role for spiking associative memory (see Section 4). Note that if the set of one-entries in the address pattern $\tilde{\mathbf{u}}$ is a subset of the one-entries in $\mathbf{u}^\mu$, then the associated pattern $\mathbf{v}^\mu$ will be a subset of the retrieval result $\hat{\mathbf{v}}$.

In the following we briefly review the analysis of one-step retrieval in the Willshaw model for random patterns [28,19,12]. First we compute the matrix load p_1 which is the fraction of one-entries in the memory matrix $\mathbf{H}$. The probability that a given synapse is *not* set by the learning of one pattern pair is $1 - k^2/n^2$, therefore for $k \ll n$,

$$p_1 = 1 - (1 - k^2/n^2)^M, \tag{3}$$

$$M = \frac{\ln(1-p_1)}{\ln(1-k^2/n^2)} \approx -\frac{n^2}{k^2}\ln(1-p_1). \tag{4}$$

For retrieval we address with a noisy version of pattern u^μ containing $\lambda \cdot k$ "correct" and $\kappa \cdot k$ "false" one-entries ($0 < \lambda \leqslant 1$; $0 \leqslant \kappa$; $z := \lambda \cdot k + \kappa \cdot k$). In the following we assume $\kappa = 0$, i.e., *no "false" one-entries*. Then we can apply the Willshaw threshold $\Theta = z =$

$\lambda \cdot k$ which will cause no "missing" ones, but possibly "false" ones in the retrieval result $\hat{v}$. A zero-entry in v^μ will become a "false" one in $\hat{v}$ with probability

$$p_{01} \approx p_1^{\lambda \cdot k}. \tag{5}$$

To obtain good retrieval results we demand a *high-fidelity-requirement* $p_{01}/(k/n) \approx 0$ and $p_{01} \to 0$ for $n \to \infty$ which states that the relative number of false ones is near zero. *High-fidelity* can be obtained by requiring $p_{01} \leqslant \varepsilon \cdot k/n$ for a small positive ε and $k/n \to 0$ which is true for (sublinearly) sparse patterns. From Eq. (5) we have to require

$$\lambda \geqslant \lambda_{\text{HiFi}} := \frac{\ln(\varepsilon \cdot k/n)}{k \cdot \ln p_1}. \tag{6}$$

The storage and retrieval of the M pattern pairs as described can be thought of as the transmission of $M \cdot n$ binary digits (of the target patterns $\mathbf{v}^\mu$) over a binary channel. For $I(p) := -p \cdot \operatorname{ld} p - (1-p) \cdot \operatorname{ld}(1-p)$ and small k/n the information per digit is $I(k/n) \approx -(k/n) \cdot \log(k/n)$. For small ε the totally stored (trans-)information is

$$\begin{aligned} T(n,k;M) &\approx M \cdot n \cdot I(k/n) \\ &\approx M \cdot k \cdot \operatorname{ld}(n/k). \end{aligned} \tag{7}$$

To *maximize* T for given n, k, and λ, we store as many patterns as possible such that the *hifi-requirement* is still fulfilled. This means we can increase the memory load p_1 by storing patterns until $\lambda_{\text{HiFi}}(p_1) = \lambda$. From the hifi-requirement and Eq. (5) we get the *maximal* matrix load $p_{1,\max}$ and with Eqs. (4), (6) the maximal number of stored patterns $M_{\max}$,

$$p_{1,\max} = (\varepsilon k/n)^{1/\lambda k}, \tag{8}$$

$$\begin{aligned} M_{\max} &\approx -\lambda^2 \cdot (\ln p_{1,\max})^2 \cdot \ln(1-p_{1,\max}) \\ &\quad \cdot \frac{n^2}{(\ln \frac{n}{\varepsilon \cdot k})^2}. \end{aligned} \tag{9}$$

Thus we obtain for the (normalized) *storage capacity* $C(n,k) := T(n,k;M_{\max})/n^2$ as the *maximal* stored information per synapse, or equivalently, per physical memory unit (bit for $\operatorname{ld} = \log_2$),

$$\begin{aligned} C(n,k) &\approx \lambda \cdot \operatorname{ld} p_{1,\max} \cdot \ln(1-p_{1,\max}) \\ &\quad \cdot \frac{1}{1 + \frac{\ln \varepsilon}{\ln(k/n)}} \leqslant \lambda \cdot \ln 2, \end{aligned} \tag{10}$$

where the upper bound is reached in the limit $n \to \infty$ for $p_{1,\max} = 0.5$ and $k = (\operatorname{ld} n)/\lambda$.

Thus in a network of n neurons a large number of $M \sim n^2/\log^2 n$ sparse patterns[2] with $k \sim \log n$ can be stored and safely retrieved using noisy address patterns where only a fraction λ of the original one-entries are used. If the address pattern contains $z \sim k$ one-entries then a stored pattern can be retrieved sequentially in time $t_{\text{seq}} = z \cdot n + n \sim n \cdot \log n$, which is much faster than $\sim M$ required by a "brute force" algorithm looking at all patterns stored in a look-up table. In a parallel implementation with n processors a retrieval requires only time $t_{\text{prl}} = z + 1 \sim \log n$.

2.3. Problems of one-step retrieval in the Willshaw model

Although the analysis of the Willshaw model may seem encouraging for technical applications, the storage capacity C, the number of storable patterns M, and thus also the advantage over simple look-up-tables may be severely limited in the following three cases.

(i) The first problem occurs if the patterns have *non-logarithmic sparseness*, i.e., $k \not\sim \log n$. From Eq. (10) we can see that $C \to 0$ for $p_{1,\max} \to 0$ or $p_{1,\max} \to 1$. On the other hand, by taking logarithms we can see from Eq. (8) that $\lim_{n\to\infty} p_{1,\max} \notin \{0,1\}$ if and only if $k \sim \log n$. For sublogarithmic $k(n)$ we have a *sparse* memory matrix with $p_{1,\max} \to 0$, and for superlogarithmic $k(n)$ we have a *dense* memory matrix with $p_{1,\max} \to 1$, both cases implying $C \to 0$.

(ii) The second problem occurs for $\kappa > 0$, i.e., if we require *fault tolerance against "false" one-entries or superpositions* in the address pattern. This can be understood by considering the two potential distributions of the "correct" and the "false" neurons in the target population. To obtain a correct retrieval result, the two distributions must not have any overlap, i.e., the largest potential of the "false" neurons should be smaller than the smallest potential of the "correct" neurons. Increasing κ will increase the overlap between the two distributions and may therefore lead to retrieval errors. The problem is most serious if the noisy one-entries in the address pattern correlate with one or several learned patterns. For example, when addressing with

[2] We write $f(n) \sim g(n)$ for $\lim_{n\to\infty} f(n)/g(n) = c$ where $0 < c < \infty$.

a superposition of two previously learned address patterns the result of one-step retrieval will always be a superposition of the corresponding target patterns.

(iii) The third problem occurs for *non-random patterns*. Our analysis is valid only for uncorrelated (e.g., random) patterns. For non-random patterns the Hebbian learning rule can lead to superposition effects that effectively extinct the information about learned patterns. For example, storing auto-associatively the three binary pattern vectors $1100\ldots0$, $0110\ldots0$, and $1010\ldots0$ is sufficient to activate all synapses connecting the first three neurons and thereby "merging" the individual patterns.

In Sections 3, 4, and 5 we will see how these problems can be solved, respectively.

3. Non-logarithmic sparseness: Compressing the memory matrix

In technical applications, the first problem of low storage capacity C for non-logarithmically sparse patterns can be solved by compressing the memory matrix applying Huffman or Golomb coding [5,9,11,12]. Optimal compression of the $n \times n$ memory matrix containing a fraction of p_1 one-entries will reduce the required physical memory by a factor of $I(p_1)$. Thus, writing p_1 for $p_{1,\max}$ for brevity, the storage capacity Eq. (10) improves to

$$C^{\mathrm{cmpr}} \approx \lambda \cdot \frac{\ln p_1 \cdot \ln(1-p_1)}{-p_1 \cdot \ln p_1 - (1-p_1)\cdot \ln(1-p_1)} \cdot \frac{1}{1+\frac{\ln \varepsilon}{\ln(k/n)}} \leqslant \lambda. \tag{11}$$

The first fraction on the right side of Eq. (11) becomes 1 for $p_1 \to 0$ or $p_1 \to 1$ and is always larger than $\ln 2$. With Eq. (8) it is easy to see that $p_1 \to 0$ for all sublogarithmic functions $k(n)$, and $p_1 \to 1$ for all superlogarithmic $k(n)$. Thus for *all* sublinear $k(n)$ we can obtain a high storage capacity $C > 0.69\lambda$. In particular, for $\lambda = 1$ we can fully exploit the physical memory, which has previously been thought to be impossible for distributed storage [22].

Compressing the memory matrix not only improves storage capacity, but can also accelerate retrieval if the *sparse* matrix entries (either 1s for $p_1 \to 0$ or 0s for $p_1 \to 1$) are efficiently represented by Golomb coding [5]. Then the sequential retrieval time is $t_{\mathrm{seq}} \sim z \cdot n \cdot \min(p_1, 1-p_1)$ which is superior to the uncompressed model even when normalizing to the information per pattern, or the totally stored information [12]. However, for parallel implementations the uncompressed model remains superior since it seems impossible to fully parallelize the access to the compressed memory matrix.

4. Improving fault tolerance by using spiking neurons

4.1. Spike counter algorithm

The second problem is how to improve retrieval quality for $\kappa > 0$ when the address pattern contains many "false" one-entries or a superposition of several learned patterns. One possibility is to iterate the retrieval several times auto-associatively and/or bidirectionally [17,24,25]. A complementary approach is to improve the one-step algorithm by retrieval with *spiking* neurons in *continuous* time [27,10,13,12].

The *basic idea* is to interpret the potentials $\tilde{\mathbf{u}} \cdot \mathbf{H}$ of the classical model as initial values for the *temporal* change $\dot{\mathbf{x}} := \mathrm{d}\mathbf{x}/\mathrm{d}t$ of the potentials. The most strongly excited neuron (with largest $\dot{\mathbf{x}}$) will be the first to emit a binary spike event after reaching an arbitrary but fixed threshold Θ. Each spike is auto-associatively fed back to the other neurons thereby updating $\dot{\mathbf{x}}$ and biasing the next spike. For hetero-association (with $u \neq v$), this requires an additional auto-associative memory matrix $\mathbf{A}$ which can be computed similar to Eq. (1).

We can embed the Willshaw threshold strategy of one-step retrieval in the continuous model such that for any time t neuron v_i has a positive potential change $\dot{x}_i(t) > 0$ if and only if it is connected to *all* neurons v_j with $\hat{v}_j(t) = 1$ that have already emitted a spike. For this we can define *spike counter* vectors $\mathbf{c^H} := \tilde{\mathbf{u}} \cdot \mathbf{H}$, $\mathbf{c^A}(t) := \hat{\mathbf{v}}(t) \cdot \mathbf{A}$, and $\mathbf{c^\Sigma}(t) := \sum_{i=1}^{n} \hat{v}_i(t)$ for the auto-associatively, hetero-associatively, and totally transmitted spikes, respectively. A simple linear example of $\dot{x}_i$ having the desired properties is

$$\dot{x}_i(t) = a \cdot c_i^H(t) + b \cdot \left(c_i^A(t) - \alpha \cdot c_i^\Sigma(t)\right) \tag{12}$$

for $a \ll b$ and $\alpha \approx 1$. In our example $c_i^H(t)$ is independent of t, and $c_i^\Sigma(t)$ independent of i. a and b are strengths of feedforward and feedback spike inputs, respectively, and the inhibition parameter α

determines the fraction of auto-associatively received spikes necessary for an *excitatory* effect of feedback. The model can be implemented efficiently using the following *spike counter algorithm* [12],

(1) $\mathbf{c^H} := \tilde{u} \cdot \mathbf{H}$; $\mathbf{c^A} := \mathbf{0}$; $\mathbf{c^\Sigma} := \mathbf{0}$; $\mathbf{x} := \mathbf{0}$; $\hat{\mathbf{v}} := \mathbf{0}$;
(2) WHILE $0 < 1$ DO
(3) $\quad \dot{\mathbf{x}} := a \cdot \mathbf{c^H} + b \cdot (\mathbf{c^A} - \alpha \cdot \mathbf{c^\Sigma})$;
(4) $\quad (t_s, j) :=$ next spike in $\{(x_i - \Theta)/\dot{x}_i, i)\}$
$\qquad$ (or $t_s = -1$ if none)
(5) $\quad$ IF $t_s < 0$ THEN RETURN $\hat{\mathbf{v}}$
(6) $\quad \mathbf{x} := \mathbf{x} + \dot{\mathbf{x}} \cdot t_s$;
(7) $\quad \hat{v}_j := 1$; $c_j^A := -\infty$;
(8) $\quad \mathbf{c^A} := \mathbf{c^A} + \mathbf{A}_j$; $\mathbf{c^\Sigma} := \mathbf{c^\Sigma} + \mathbf{1}$;
(9) ENDWHILE

In line 7, $c_j^A := -\infty$ prevents neuron v_j from emitting a second spike or "bursting". In line 8, $\mathbf{A}_j$ denotes the jth line of matrix $\mathbf{A}$. A *sequential implementation* requires $t_{\text{seq}} \sim (z + k) \cdot n$ which is $\sim n \cdot \log n$ for $k \sim \log n$, not worse than one-step retrieval. However, a *parallel implementation* with n processors requires $t_{\text{prl}} \sim z + k \cdot \log n \sim \log^2 n$ since computing the minimal predicted spike time t_s in line 4 requires time $\log n$ for each of the k loop runs. This is still quite efficient, but slower than one-step retrieval. Even more efficient could be a direct VLSI *analog implementation* where "computing" the next spike would be an intrinsic effect of the electrical dynamics and a retrieval would be possible in a single "step" [3]. For this the regime with a sparse memory matrix, $p_1 \to 0$, would be most preferable since the number of $p_1 \cdot n^2$ synaptic connections is the limiting factor for on-chip integration.

Note that for $\kappa = 0$ (no "false" one-entries in the address pattern) the spike counter algorithm is equivalent to one-step retrieval since then all target neurons will fire at the same time. Thus the analysis for storable patterns and storage capacity in Section 2 applies also to the spike counter algorithm.

To understand the advantage over one-step retrieval we note that the spike counter algorithm works as a *clique-detector* (at least in the regime $a \ll b$, $\alpha \approx 1$): A neuron can emit a spike only if it is connected to all other neurons that have already spiked. Thus the retrieval result will be a maximal clique (of the graph defined by the auto-associative connections). In contrast to one-step retrieval this allows the recall of patterns with different numbers of one-entries. It also means that there is a "built-in" threshold control for noisy address patterns. Finally, low quality retrieval (due to low quality address patterns) can be detected and rejected at an early time when only few incompletely connected neurons have spiked.

This also implies that the spike counter algorithm is much more robust for $\kappa > 0$, which can be understood by considering the neurons' potential distribution separately for the "correct" and "false" neurons (cf. the remarks in Section 2.3) *before* the first spike occurs (which is equivalent to the distribution of the potentials $\mathbf{x} = \tilde{\mathbf{u}}H$ for one-step retrieval). While one-step retrieval requires that *all* "correct" neurons have a larger potential than any "false" neuron, the spike counter algorithm requires this only for a small number of *highly excited* "correct" neurons. This ensures that the first few spikes are "correct", which is sufficient for suppressing the "false" neurons by subsequent feedback. An asymptotic analysis in [12] shows that for maximal p_1 and M as in Eqs. (8), (9), $n \to \infty$ and *sublogarithmic* $k(n)$ the spike counter algorithm is robust against a very large number of $\kappa_{\max} \cdot k$ "false" one-entries in the address pattern, where

$$\kappa_{\max} \approx \frac{\lambda^2}{2} \cdot (\varepsilon k)^{1/(\lambda k)} \cdot \frac{k \cdot n^{1/(\lambda k)}}{\ln n}. \tag{13}$$

This means that κ is allowed to grow faster than any polynomial in $\log n$. Of course, requiring fault tolerance against missing one-entries with sufficiently small $\lambda < 1$ and $\kappa = 0$ will generally (not only for sublogarithmic $k(n)$) imply fault tolerance against "false" one-entries with $\lambda = 1$ and $0 < \kappa \leqslant \kappa_{\max}$, where $\kappa_{\max}$ depends on n, k, M, and ε. One interesting consequence of this is that iterating (auto-associative) retrieval can further improve the result. For example, consider addressing in the first step with $\lambda < 1, \kappa = 0$. The result will be a pattern containing all correct one-entries but possibly additional false one-entries, i.e., $\lambda = 1$ and $\kappa \geqslant 0$. Addressing with this can finally result in a perfect retrieval result. Thus, iterative retrieval can in principle improve the storage capacity equation (10) of one-step retrieval (or at least the convergence towards $\lambda \cdot \ln 2$), but only when addressing with $\lambda < 1$ (e.g., see [24] for auto-associative iteration of one-step retrieval with $\lambda = 0.5$).

4.2. Biological variant of the spike counter model

The spike counter algorithm has also a very nice biological interpretation. (Actually, the spike counter algorithm has been derived from biological models [10,13].) Consider the membrane potential equation of the Hodgkin–Huxley type (e.g., see [15]),

$$\tau \frac{\mathrm{d}}{\mathrm{d}t} x(t) = -x(t) + g_{\mathrm{ex}}(t) \cdot \big(E_{\mathrm{ex}} - x(t)\big) + g_{\mathrm{in}}(t) \cdot \big(E_{\mathrm{in}} - x(t)\big), \tag{14}$$

where x is the membrane potential, τ is the decay time constant of membrane potentials, E_{ex} and E_{in} are excitatory and inhibitory equilibrium (reversal) potentials, and g_{ex} and g_{in} are excitatory and inhibitory conductances. Each time the neuron receives an excitatory spike input this will cause a temporary increase of g_{ex} which will drive x towards E_{ex}. Different inputs from different synapses sum up in g_{ex} if they occur within a sufficiently small time window. For inhibitory inputs similar arguments hold.

Thus, within a sufficiently small time window, the excitatory conductance g_{ex} approximates the weighted sum of the spike counters c^H and c^A, and the inhibitory conductance g_{in} can approximate c^Σ if one additionally assumes an unspecific all-to-all inhibitory recurrent connection within population v, i.e., each neuron makes synapses with weight -1 to *all* other neurons. Since real neurons can make only either excitatory or inhibitory synapses (Dale's law) a more realistic equivalent model variant assumes an extra inhibitory neuron population which receives heteroassociative feedforward and auto-associative feedback inputs similar to the excitatory population (see [10,13] for more details).

Simulations with the realistic neuron model show the following points [13,12]:

(i) Biologically realistic networks can exhibit a performance similar to the technically optimized networks even in a realistic "noisy" regime with balanced excitation and inhibition.
(ii) The precise input spike times on a millisecond scale play a decisive role for the activation of neurons, in particular if the input addresses superpositions of several patterns.
(iii) Plausible durations of retrieval are 5–10 ms alternating with silent inhibitory phases, resulting in oscillations or irregular fluctuations.
(iv) A rank order code relative to the oscillations or fluctuations is most plausible: Early spikes carry much more reliable information than late spikes.
(v) Reciprocal connections between neuron populations can improve performance similar to iterative retrieval.

5. Non-random patterns, look-up tables, and "grandmother cells"

A serious limitation of distributed neural associative memory is that capacity C and number M of storable pattern associations $\mathbf{u}^\mu \mapsto \mathbf{v}^\mu$ can be severely limited if the patterns are not uncorrelated as it is the case for most real-world data. The reason for this problem is the distributed way information is stored in the memory matrix. Each neuron and each synapse codes a large number of different entities causing unwanted superposition effects. We can avoid this superposition problem by adding an intermediary *"grandmother-cell"* layer w mediating between populations u and v, where neuron w_μ codes the μth entity or pattern pair ($\mu = 1, \ldots, M$). In the following we will refer to this model variant as the *grandmother model* (cf. [1]).

The grandmother model corresponds simply to a look-up table and can be implemented efficiently as follows: Let $\mathbf{U}$, $\mathbf{V}$ be $M \times n$ matrices such that $\mathbf{U}_\mu = \mathbf{u}^\mu$ and $\mathbf{V}_\mu = \mathbf{v}^\mu$ for $\mu = 1, \ldots, M$, i.e., each matrix row corresponds to a pattern vector. The matrices can be compressed analogously to Section 3 and will require only space $M \cdot n \cdot I(k/n)$ for patterns with k one-entries. For retrieval with an address pattern $\tilde{\mathbf{u}}$ containing z one-entries we first compute the overlaps $\mathbf{x} := \mathbf{U} \cdot \tilde{\mathbf{u}}^{\mathrm{T}}$ and then the retrieval result $\hat{\mathbf{v}} = \mathbf{v}^{\max \arg x_\mu}$. (For arbitrary patterns we would use the Hamming distance instead of overlaps.) The overlaps can be interpreted as the potentials of the grandmother cells w, where $\hat{\mathbf{v}}$ results from a winner-takes-all competition between the grandmother cells.

A sequential implementation in analogy to Section 3 requires only $t^{\mathrm{G}}_{\mathrm{seq}} \approx z \cdot M \cdot k/n$ steps for the grandmother model. (Essentially, we have to add z sparse *columns* of $\mathbf{U}$.) This compares to $t^{\mathrm{W}}_{\mathrm{seq}} \approx z \cdot n \cdot \min(p_1, 1 - p_1)$ for the Willshaw model. With M, p_1, and $p_{1,\max}$ as in Eqs. (4), (3), (8) we obtain

$$\nu := \frac{t^{\mathrm{G}}_{\mathrm{seq}}}{t^{\mathrm{W}}_{\mathrm{seq}}} \approx \frac{-\ln(1-p_1)}{k \cdot \min(p_1, 1-p_1)}$$

$$\leqslant \frac{-\ln(1-p_{1,\max})}{k \cdot \min(p_{1,\max}, 1-p_{1,\max})} \tag{15}$$

$$\approx \begin{cases} 1/k, & p_{1,\max} \to 0, \\ \lambda \cdot \frac{\ln(\lambda k) - \ln\ln\frac{n}{\varepsilon k}}{\ln\frac{n}{\varepsilon k}}, & p_{1,\max} \to 1, \end{cases} \tag{16}$$

where we used $1 - p_{1,\max} \approx -\ln p_{1,\max}$ for $p_{1,\max} \to 1$. Remember from Section 2.3 that the memory matrix is sparse, balanced, or dense for sublogarithmic, logarithmic, or superlogarithmic $k(n)$, respectively. Thus, even for uncorrelated patterns, the Willshaw model performs worse than the grandmother model ($\nu < 1$) for almost all parameters. The Willshaw model remains superior only for dense memory matrix with $p_{1,\max} \to 1$ when k is almost linear, for example, $k = n^d$ with $1/(1+\lambda) < d < 1$.

The Willshaw model also remains superior for a *parallel implementation* if the patterns are uncorrelated and a limited fault tolerance is sufficient. Then a retrieval requires $t^{\mathrm{W}}_{\mathrm{prl}} = z + 1$ for the Willshaw model with n processors, compared to $t^{\mathrm{G}}_{\mathrm{prl}} = z + \log M$ for the grandmother model with $M \gg n$ processors. Note that both the Willshaw and the grandmother model are efficient ($t_{\mathrm{seq}}/M, t_{\mathrm{prl}}/n \to 0$) only for sparse patterns with $k/n \to 0$. Non-sparse patterns require additionally a sparse recoding (or indexing) as is done in multi-index hashing [6]. In summary, although there are quite efficient implementations, it appears that distributed neural associative memories have no practical advantage over compressed look-up tables or multi-index hashing, at least not for solving the Best Match problem on sequential computers.

6. Conclusions

In this work we have reviewed the analysis of different variants of neural associative memory [28, 19,24,13,11,12] in the context of the Best-Match-Problem [18]. First we have discussed the classical one-step retrieval algorithm of the distributed Willshaw model [28,19] and its use for modeling neural networks of the brain and for applications in information retrieval. On the one hand side the Willshaw model allows a fast and fault-tolerant access to stored pattern information, much faster than naive implementations of look-up tables [20]. On the other hand, the classical Willshaw model has three serious problems: First, a high storage capacity can be achieved only for $k \sim \log n$, i.e., if the number k of one-entries in a pattern grows logarithmically in the neuron number n. Second, one-step retrieval has only a limited tolerance against noisy one-entries in the address patterns, in particular if the noise correlates with distracting patterns, e.g., if the address pattern is a superposition of several learned patterns. Third, the number M of storable patterns may be severely limited for non-random patterns.

The first problem can be solved by compressing the memory matrix [11] leading to high storage capacities for *any* sublinear $k(n)$. To solve the second problem we have proposed the spike counter algorithm which significantly improves fault tolerance against noisy one-entries in the address patterns and allows a fast separation of superpositions by making use of precise spike timing [12]. Besides this, the spike counter model has a very natural implementation in biologically realistic conductance based neuron models and in plausible cortical network models [13,14,12,15]. The third problem of storing correlated patterns appears to be inherent in distributed storage by Hebbian learning. One possible solution is to have extremely sparse representations, in the most extreme case a "grandmother cell" model, where each cell codes exclusively a single entity [1]. We have worked out the relation between the "grandmother cell" model, look-up tables, and the distributed Willshaw model. Surprisingly, it turns out that the "grandmother cell" model, if implemented as a compressed look-up table on a sequential computer, is even more efficient than the distributed Willshaw model in most cases. For information retrieval in technical applications the distributed Willshaw model remains superior only for uncorrelated patterns, if a limited fault tolerance is sufficient, and if the patterns are almost non-sparse ($k \sim n^d$, $d \approx 1$) or if a parallel implementation is possible.

References

[1] H.B. Barlow, Single units and sensation: a neuron doctrine for perceptual psychology, Perception 1 (1972) 371–394.

[2] V. Braitenberg, Cell assemblies in the cerebral cortex, in: R. Heim, G. Palm (Eds.), Theoretical Approaches to Com-

plex Systems, in: Lecture Notes in Biomathematics, vol. 21, Springer-Verlag, Berlin, 1978, pp. 171–188.
[3] E. Chicca, D. Badoni, V. Dante, M. D'Andreagiovanni, G. Salina, L. Carota, S. Fusi, P. Del Giudice, A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory, IEEE Trans. Neural Networks 14 (2003) 1297–1307.
[4] R. Fay, U. Kaufmann, A. Knoblauch, H. Markert, G. Palm, Integrating object recognition, visual attention, language and action processing on a robot using a neurobiologically plausible associative architecture, in: Proc. KI2004 Workshop on NeuroBotics, Ulm, Germany 2004.
[5] S.W. Golomb, Run-length encodings, IEEE Trans. Inform. Theory 12 (1966) 399–401.
[6] D. Greene, M. Parnas, F. Yao, Multi-index hashing for information retrieval, in: Proc. 35th Annual Symp. on Foundations of Computer Science, 1994, pp. 722–731.
[7] D.O. Hebb, The Organization of Behavior. A Neuropsychological Theory, Wiley, New York, 1949.
[8] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. USA 79 (1982) 2554–2558.
[9] D.A. Huffman, A method for the construction of minimum redundancy codes, Proc. Inst. Radio Engineers 40 (1952) 1098–1101.
[10] A. Knoblauch, Assoziativspeicher aus spikenden Neuronen und Synchronisation im visuellen Kortex, Diploma thesis, Department of Neural Information Processing, University of Ulm, Germany, 1999 (in German).
[11] A. Knoblauch, Optimal matrix compression yields storage capacity 1 for binary Willshaw associative memory, in: O. Kaynak, E. Alpaydin, E. Oja, L. Xu (Eds.), Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003, in: Lecture Notes in Comput. Sci., vol. 2714, Springer-Verlag, Berlin, 2003, pp. 325–332.
[12] A. Knoblauch, Synchronization and pattern separation in spiking associative memory and visual cortical areas, PhD thesis, Department of Neural Information Processing, University of Ulm, Germany, 2003.
[13] A. Knoblauch, G. Palm, Pattern separation and synchronization in spiking associative memories and visual areas, Neural Networks 14 (2001) 763–780.
[14] A. Knoblauch, G. Palm, Scene segmentation by spike synchronization in reciprocally connected visual areas. I. Local effects of cortical feedback, Biol. Cybernet. 87 (3) (2002) 151–167.
[15] C. Koch, I. Segev (Eds.), Methods in Neuronal Modeling, MIT Press, Cambridge, MA, 1998.
[16] T. Kohonen, Associative Memory: A System Theoretic Approach, Springer, Berlin, 1977.
[17] B. Kosko, Bidirectional associative memories, IEEE Trans. Systems Man Cybernet. 18 (1988) 49–60.
[18] M.L. Minsky, S. Papert, Perceptrons: An Introduction to Computational Geometry, MIT Press, Cambridge, MA, 1969.
[19] G. Palm, On associative memories, Biol. Cybernet. 36 (1980) 19–31.
[20] G. Palm, Computing with neural networks, Science 235 (1987) 1227–1228.
[21] G. Palm, Cell assemblies as a guideline for brain research, Concepts in Neurosci. 1 (1990) 133–148.
[22] G. Palm, Memory capacities of local rules for synaptic modification. A comparative review, Concepts in Neurosci. 2 (1991) 97–128.
[23] R.W. Prager, F. Fallside, The modified Kanerva model for automatic speech recognition, Comput. Speech Language 3 (1989) 61–81.
[24] F. Schwenker, F.T. Sommer, G. Palm, Iterative retrieval of sparsely coded associative memory patterns, Neural Networks 9 (1996) 445–455.
[25] F.T. Sommer, G. Palm, Improved bidirectional retrieval of sparse patterns stored by Hebbian learning, Neural Networks 12 (1999) 281–297.
[26] K. Steinbuch, Die Lernmatrix, Kybernetik 1 (1961) 36–45.
[27] T. Wennekers, G. Palm, On the relation between neural modelling and experimental neuroscience, Theory Biosci. 116 (1997) 273–289.
[28] D.J. Willshaw, O.P. Buneman, H.C. Longuet-Higgins, Nonholographic associative memory, Nature 222 (1969) 960–962.