A Cortex-inspired Associative Memory with *O*(1) Time Complexity Learning, Recall, and Recognition of Sequences

Gerard J. Rinkus
Volen Center for Complex Systems
Brandeis University

Abstract

A cortex-inspired associative memory model possessing *O*(1) time complexity for both storage (learning) and retrieval (recall, recognition) of sequences is described. It learns sequences, specifically, binary spatiotemporal patterns, with single trials. Results are given demonstrating *O*(1) retrieval of: a) a sequence's remaining items when prompted with its initial item, i.e., episodic recall; and b) the most similar stored sequence when presented with a novel sequence, i.e., recognition/categorization. The hidden (representation) layer, L2, is organized into winner-take-all competitive modules (CMs) hypothesized to be analogous to cortical minicolumns. Representations consist of one active unit per CM. The heart of the model is a matching algorithm that, given the current input *moment*, σ, i.e., the current input item in the context of the sequence thus far: a) finds the stored representation, Δ_{σ^*}, of that previously experienced moment, σ^*, out of *all* previously experienced moments, which is spatiotemporally most similar to σ; and b) returns a normalized measure, *G*, of that similarity. When in recall or recognition mode, the model simply reactivates Δ_{σ^*} since it is the most likely hypothesis given the model's history and the current input. When in learning mode, the model injects an amount of noise, inversely proportional G, into the process of choosing the cells to represent σ. This yields the property that the size of the intersection between representations is an increasing function of the spatiotemporal similarity of the moments that they represent. Thus, the higher-order statistics (spatiotemporal similarity structure) of the set of learned sequences is reflected directly in the patterns of intersections over the set of representations. This property, in conjunction with the use of the binary sparse representation, makes the *O*(1) recall and recognition (i.e., inference) possible.

1. Introduction

Human capability still far exceeds that of machines on a vast range of information processing tasks. Parsing speech in noisy environments, understanding the meaning of that speech at many levels from the literal up through recognizing sarcasm or irony, reading an opponent's intention, and thus being able to predict his or his team's next or ensuing movements, etc. These are all spatiotemporal (sequential) tasks that humans with sufficient experience routinely perform both accurately and in real time. An underlying commonality between tasks like these is that they all require rapid inference in exponentially large hypothesis spaces. For example, the space of possible movement patterns over the next five seconds of the set of opposing soccer team players, is exponentially large. Nevertheless, an experienced player can often correctly predict the pattern with sufficient accuracy to be successful in the moment. The relatively slow speed of neural operations implies that somehow the brain drastically pares down the huge hypothesis space so that very few hypotheses are explicitly considered at any given moment.

At base, the ability to achieve such a paring down is a function of how the hypotheses are represented in memory. For example, the $\log_2 N$ time complexity of binary search depends on the fact that the set of N items being searched over is arranged by numerical similarity, e.g.., 8 is more similar to 9 than to 2. More generally, one might say that search and retrieval is abetted to the extent that the similarity structure of the hypothesis space is reflected in the physical representation (storage) of the hypotheses. Of course, ensuring that such a property—i.e., similar inputs map to similar representations (SISR)—holds as successive inputs are presented to the system generally incurs increased storage-time complexity. However, the model described herein implements this property in such a way that it achieves the theoretically optimal time complexity, i.e., $O(1)$, for both storage and retrieval.

This model, originally developed in Rinkus (1995, 1996), employs sparse distributed representations to do unsupervised learning, recall and recognition of binary spatiotemporal patterns, or *sequences,* and possesses this immediate search/retrieval property. More formally, the model has $O(1)$ search time complexity, meaning that the number of computational steps necessary to find the most similar stored sequence (to some query sequence) remains constant no matter how many sequences have been stored, and no matter how long individual sequences are. This holds whether the model is asked to recall the remainder of a sequence given a prompt (episodic recall) or recognize a novel sequence (by activating the memory trace of the most similar previously learned sequence). Moreover, the model does *single-trial learning* of its input sequences and, in fact, has $O(1)$ storage time complexity as well.

The model, named TEMECOR (TEmporal MEmory using COmbinatorial Representations), achieves this performance primarily through two features: a) a particular, sparse distributed, k-WTA, representation, modeled after the minicolumn structure of neocortex; and b) an algorithm which computes, at each moment during learning, the maximal similarity, G, between the currently presenting sequence and *all* of its stored sequences (and *all* of their prefixes) and then adds an amount of noise, inversely proportional to G, into the choice of units that will become active at that moment. Together, these features yield the key property that the degree of overlap between two stored sequence representations (memory traces) is an increasing function of the spatiotemporal similarity of the sequences that they represent, i.e., the SISR property. Thus, both the memory traces of the individual sequences and the higher-order statistics (similarity structure) of the set of sequences reside in a single monolithic structure (set of weights).

The specific claim of this paper is that the proposed model has $O(1)$ computational *time* complexity for both storage (learning) and retrieval (recall, recognition) of sequences. This follows immediately from the fact that none of the model's equations (Eqs. 1-11) have any dependence on the number sequences stored.

While the computational *space* complexity, i.e., storage capacity, of an associative memory model is also of paramount importance, this paper does not present a definitive result for the model's space complexity. However, the appendix gives two indirect arguments suggesting that the model's space complexity will allow scaling to biologically relevant problem sizes. A comprehensive examination of the model's space complexity will be presented in a future paper.

2. Background

The use of distributed representations for categories (hidden variables, causes, factors) has been central to both the multilayer perceptron (MLP), e.g., Backpropagation, and associative memory approaches to computational intelligence. However, this is not true of the family of probabilistic models, generally referred to as *graphical models,* which has become the dominant paradigm in the field. This very large family includes Bayesian Belief Networks (Pearl, 1988), hidden Markov models (HMMs), mixture models, principal components analysis, amongst others (Roweis & Ghahramani, 1999). All of these models were originally described with the assumption of a localist (singleton) representation of categories. Many of them, notably HMMs, which is by far, the predominant approach for speech recognition (and perhaps for the domain of temporal/sequential pattern recognition in general), have retained this localist assumption over almost their entire developmental courses.

More recently, there has been a movement towards distributed representations for categories in graphical models. This movement is often attributed to the observation, by Williams & Hinton (1991), of a fundamental inefficiency of localist representations. They observed that, in order to represent only *N* bits of information about the history of a time series, the single multinomial state variable of a standard, localist HMM requires 2^N possible values, i.e., an HMM with 2^N nodes. In contrast, a distributed representation of state with only N binary variables (an HMM with *2N* nodes) can also store N bits of history information. A number of models using distributed representations of state have since been proposed (Ghahramani & Jordan, 1997; Brown & Hinton, 2001; Jacobs et al., 2002; Blitzer et al., 2005).

The benefit of distributed representations can be viewed as increased capacity for storing the higher-order statistics present in a sequence (or in any data set, more generally). This point can be made more concretely in terms of sparse distributed representations. In a sparse distributed representation, in which any given representee is represented by a small subset of representational units (e.g., 100 out of 1000) being active, the higher-order statistics, i.e., similarity structure, of the set of representees, can be represented directly by the patterns of overlaps (intersections) over their representations. The more similar two representees are, the more units they have in common.

Figure 1 illustrates this concept. At left is a hypothetical set of observations (A, B, D) in some feature space and a hypothetical underlying category structure (C1, C2) consistent with the distances between the observations in the feature space. At lower right is one possible sparse distributed representation: each observation is represented by 5 out of 12 of the units (which is sparse enough to make the point). The vertical dotted lines on the right show how the pattern of overlaps of the three representations can directly represent the higher-order statistics (a nested categorical structure) of the observations: the more similar observations, A and B, have three units in common, which constitutes a representation of subcategory, C1, while the full set of observations have only two units in common, which constitutes a representation of the super-category, C2.

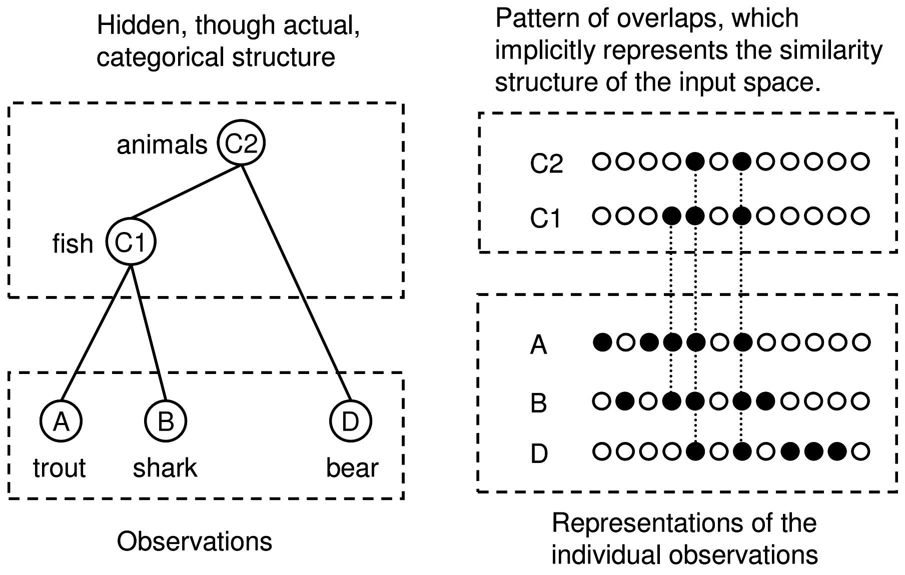

Hidden, though actual, categorical structure
animals
C2
fish
C1
A
B
D
trout
shark
bear
Observations
Pattern of overlaps, which implicitly represents the similarity structure of the input space.
C2
C1
A
B
D
Representations of the individual observations

Figure 1: (Left) A hypothetical set of three observed patterns (A, B, D) from some input space and a hypothetical underlying category structure. A and B are more similar than A and D or B and D. (Lower right) A sparse distributed representation of the individual observations, A, B, and D. (Upper right) Different subsets of units, C1 and C2, corresponding to the pair-wise and triple-wise overlaps respectively, constitute explicit representations of higher-order similarity (i.e., category) structure.

Figure 1 is intended to show only that it is *possible* to represent the similarity structure of a set of individual observation by the pattern of overlaps over their representations. The information present in the pattern of overlaps in the box at upper right is fully analogous to a dendrogram (hierarchical clustering analysis) typically used to demonstrate the categorization behavior of learning models that use fully distributed representations (Elman, 1990). In the case of sparse distributed representations, as in Figure 1, the appropriate measure of distance between representations is Hamming distance. The pattern of overlaps in the upper right box is isomorphic to a dendrogram in which A and B would be clustered as C1 and then C1 and D would be clustered as C2, which would look exactly like the tree structure on the left, as desired. The evidence, given in Section 4, that the proposed model actually uses this implicit category information is that it correctly classifies novel observation (sequences) even though there are no explicit and separate representations of category information in the model.

The representation of similarity structure by patterns of overlap constitutes nothing less than an additional representational dimension, one which is unavailable at either extreme of the representational continuum, i.e., in either localist or densely (fully) distributed representations. Under a localist scheme there are no code overlaps; thus, the only way to represent the higher-order statistical structure over the representees is in the weights between the representations. Under a dense scheme, where formally, *all* units participate in *all* representations, all code overlaps are complete, e.g., a hidden layer with M real-valued units in a Backpropagation model. Here, the only available representation of similarity is a scalar (e.g., Euclidean) distance between the representations.[1] The possibility of representing similarity by overlaps has been generally underappreciated thus far. However, it is at the core of the spatial model of Rachkovskij & Kussel (2000, 2001) and of the spatiotemporal model proposed herein.

TEMECOR has not been developed within the framework of graphical models. In particular, although it deals with spatiotemporal patterns, it has not been developed as an extension of any variant of HMMs, nor of the more general class of Dynamic Belief Nets (DBNs). Its learning algorithm will, thus, not be cast as a version of the expectation-maximization (EM) algorithm (Dempster & Laird, 1977). Instead, the learning algorithm, which uses only single trials, will be described as computing a spatiotemporal similarity metric and using it to guide the choice of representation so that the pattern of overlaps over the final set of representations reflects the higher-order spatiotemporal statistics of the input pattern set. As will be shown, this property leads directly to a simple retrieval (inference) algorithm, that finds the most similar stored sequence to a query sequence (i.e., the maximum likelihood hypothesis) in a single computation, i.e., a computation which does not depend on how many sequences have been stored..

TEMECOR has also not been extended from the various MLP variants that have been applied to temporal/sequential patterns, e.g., Waibel (1989), Jordan (1986), Elman (1990), Williams & Zipser (1989). The MLP-based approach has been successfully applied to many categorization problems.

[1] This issue is central to compositionality. Classical AI approaches are localist. Compositionality is implicitly present owing to the clear mapping between representees, both wholes and parts, and their representations. For connectionist approaches that employ dense distributions, there is no such clear mapping available, hence the classicist's (Fodor & Pylyshyn, 1988) claim that connectionist systems lack what Van Gelder (1990) terms *syntactic compositionality*. However, as discussed here, sparse connectionist models can possess syntactic compositionality.

However, such models have: a) not been demonstrated to *simultaneously* explain *both* episodic memory (recall of sequences) and semantic memory (categorization of novel sequences); b) require numerous training trials of each exemplar; and c) suffer from *catastrophic forgetting* in which nearly all information about a previously learned data set can be lost when the model learns a new data set (McCloskey & Cohen, 1989; French, 1991; Cleeremans, 1993). Moreover, as discussed by Cleeremans, such Backpropagation-based models have difficulties that are particularly exacerbated in the context of learning *complex sequences*, which are sequences in which items can recur multiple times and in varying contexts.

TEMECOR is closer to a number of sparse distributed associative memory models (e.g., Willshaw et al., 1969; Palm, 1980; Lynch, 1986; Kanerva, 1988; Moll & Miikkulainen, 1996), although it differs from these in many ways, including its focus on spatiotemporal patterns and its novel use of noise to effect the crucial SISR (similar inputs map to similar representations) property. It also has similarities to the Synfire Chains model (Abeles, 1991) in that it maps inputs, specifically sequences, into spatiotemporal memory traces, although the focus is not on the fine-scale timing of spikes.

Several authors, e.g., Dayan & Zemel (1995), have discussed the oppositional natures of the coding mechanisms of cooperation and competition. O'Reilly (1998) explains that any use of competition prevents the "cooperativity and combinatoriality of true distributed representations", and the "need to preserve independence among the units in a [distributed representation] prevents the introduction of any true activation-based competition." He adds that the development of sparse distributed representations, which necessarily integrate these two opposing mechanisms, has been challenging largely because they are difficult to analyze mathematically. Relatedly, a number of authors have discussed the benefits of and need for sparse distributed representations (Olshausen & Field, 1996; Hinton & Ghahramani, 1997). TEMECOR may provide a hopeful alternative in this regard because its particular implementation of the sparse distributed paradigm cleanly separates the cooperative and competitive aspects. As will be seen shortly, competition is isolated within the model's winner-take-all competitive modules (CMs), and the cooperation is present in that internal representations (codes) are sets of co-active units, one per CM.

3. Model Description

Figure 2 shows the types of sequences processed by the model. In this example, the input surface consists of 16 abstract binary features arranged in a vertical array; array position does not reflect any spatial relation in the physical input domain. Figure 2a shows an instance of a completely uncorrelated or ‘white noise’ sequence in which a randomly chosen subset of binary features is chosen to be active at each moment.. The intention of using input sets consisting of such uncorrelated sequences is to approximate the conditions of episodic memory wherein essentially arbitrary combinations of features are remembered essentially permanently despite occurring only once. The model also handles correlated, or more specifically, *complex,* sequence sets (Figure 2b) in which whole input items can recur multiple times and in different temporal contexts, as is necessary for modeling language-like domains.

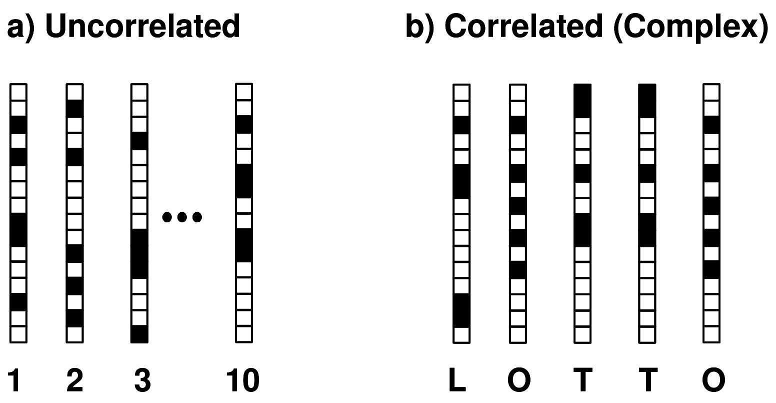


Figure 2: a) A random binary spatiotemporal pattern (sequence) having 10 time steps, or *moments*. The input surface has 16 abstract binary features, a randomly chosen subset of which is active at each moment. b) A complex sequence in which the same item (subset of features), e.g., an English letter, can recur in different contexts, as is appropriate for representing linguistic domains.

As shown in Figure 3, the two-layer model[2] uses sparse distributed binary representations, or *codes*, in its internal layer, L2. In this example, L2 is divided up into $Q = 4$ winner-take-all competitive modules (CMs) and every code consists of one active unit in each CM (black L2 units). Hence, the layer instantiates a particular type of *k*-WTA architecture. Here, L1 is arranged as a 2D grid of binary abstract features; the model is agnostic to modality and to any inherent dimensionality of the input space. Figure 3 shows the state of a small instance of the model on all four time steps

[2] This model has already been generalized to a hierarchical framework allowing an arbitrary number of layers (Rinkus & Lisman, 2005) and will be described in a separate manuscript.

during the learning of the sequence, [ABDB]. It illustrates several aspects of the model's basic operation, as well as some nomenclature. Note that only a small subset of the connections (weights) are shown in this figure. Throughout this paper, we will assume complete inter-level connectivity: all L1 units connect with all L2 units in both the bottom-up (BU) and top-down (TD) direction. We also assume nearly complete intra-level (horizontal, or H) connectivity (within L2 only): all L2 units contact all other L2 units except those in its own CM.

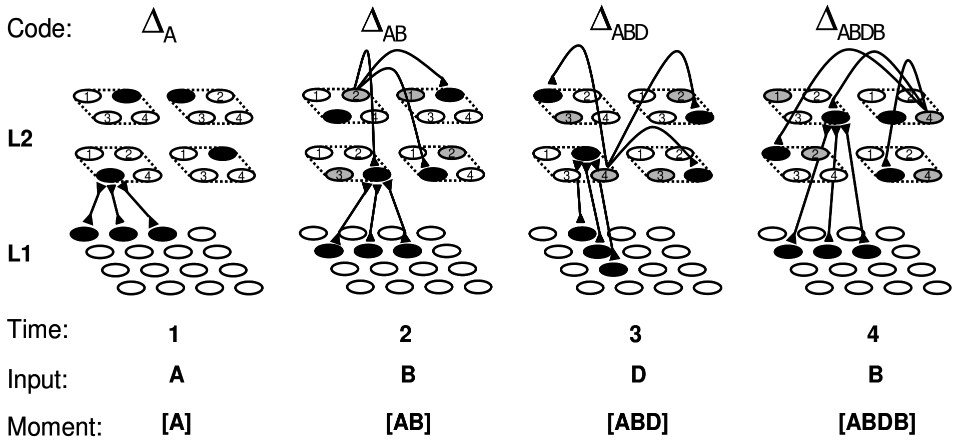


Figure 3: A depiction of a small instance of the model on all four time steps, or moments, while learning the sequence, [ABDB]. Only a representative subset of the weight increases on each time step are shown (black lines). See text for detailed explanation of the figure.

Figure 3 shows that during learning, a new L2 code generally becomes active for each successive time step, or *moment*, of a sequence. A moment is defined as a particular spatial input (item) in the context of the particular sequence of items leading up to it (i.e., a particular *prefix* or *history*). Thus, the sequence [ABDB] consists of four unique moments, [A], [AB], [ABD], and [ABDB], but only three unique input items. The symbol, σ, is used to denote a moment. The symbol, Υ, is used to denote an individual input item, i.e., a set of co-active L1 units (a purely spatial pattern). The symbol, Δ, is used to denote an L2 code, i.e., a set of active L2 units; e.g., Δ_{AB} (black units only) denotes the L2 code for the moment, [AB]. The figure also shows that the same input item, e.g., B, generally gives rise to different L2 codes, depending on prior context; hence, Δ_{AB} is not equal to Δ_{ABDB}, although they overlap at one CM (the one at lower right).

Figure 3 graphically depicts a representative subset of the learning that occurs during presentation of the sequence. All weights, BU, TD, and H, are initially zero. At *t=1*, an input item, A, presents at the input layer, L1.

Active L1 units send signals via their BU weights to L2. Assume for the moment that some random L2 code, Δ_A, is chosen. The details of how L2 codes are chosen will be elaborated shortly. The BU and TD weights between the three active L1 units comprising item A, and the four units of Δ_A are increased (only the increases involving the lower left CM's winning unit are shown). The arcs from the gray L2 unit in the upper left CM at *t=2* show a subset of the horizontal (temporal) learning that would occur within L2 from Δ_A (which was active at *t=1* and is shown as the gray units in L2 at *t=2*) onto Δ_{AB} (black units in L2 at *t=2*). Thus, successively active L2 codes are chained together by a temporal Hebbian learning law, Eq. 1a, in which an L2 unit active at *t-1* increases its weights onto all L2 units active at *t* (except for units in its own CM). The learning law for the BU and TD weights is the more familiar Hebbian rule, Eq. 1b, in which the weight is increased if L1 unit, *i*, and L2 unit, *j*, are co-active.

$$w_t(i,j) = \begin{cases} 1 & , i \in \Delta_{t-1}, j \in \Delta_t, CM(i) \neq CM(j) \\ w_{t-1}(i,j) & , \text{otherwise} \end{cases} \tag{1a}$$

$$w_t(i,j) = \begin{cases} 1 & , i \text{ and } j \text{ both active at } t \\ w_{t-1}(i,j) & , \text{otherwise} \end{cases} \tag{1b}$$

The model's operation during learning is as follows. In Eq. 2, each individual L2 unit, *i*, computes the weighted sum, $\phi(i)$, of its horizontal (H) inputs from the L2 units active on the prior time step, Δ_{t-1}. The analogous summation, $\psi(i)$, for the bottom-up (BU) inputs from the currently active L1 units, Υ_t, is computed in Eq. 3.[3] In Eqs. 4 and 5, these summations are normalized, yielding $\Phi(i)$ and $\Psi(i)$. The ϕ normalization is possible because L2 code size is invariant and equal to *Q*, the number of CMs. For instance, in the model of Figure 3, all L2 codes have exactly *Q* = 4 active units. Therefore, the maximum total horizontal input possible on any time step is *Q-1* = 3 (since units do not receive H synapses from other units in their own CM). Thus, we normalize by dividing a unit's summed H inputs by *Q-1*. Similarly, BU inputs can be normalized because the number of active input features on any given time step is assumed to be a constant, *M*.[4]

[3] This equation is changed slightly in Sec. 4 to accommodate a slightly more complex representation of time used in the computer model. The details will be explained in Sec. 4.

[4] The theory does not require the strict normalization of the number of active input features per time step, but this is not discussed in this paper.

$$\phi_t(i) = \sum_{k \in \Delta_{t-1}} w_{ki} \qquad (2)$$

$$\psi_t(i) = \sum_{k \in \Upsilon_t} w_{ki} \qquad (3)$$

$$\Phi_t(i) = \frac{\phi_t(i)}{Q-1} \qquad (4)$$

$$\Psi_t(i) = \frac{\psi_t(i)}{M} \qquad (5)$$

In Eq. 6, each L2 unit computes its own *local degree of match* (support), $\chi_t(i)$, to the current moment. Formally, this is implemented as multiplication between the *normalized* versions of a unit's horizontal (H) and bottom-up (BU) inputs, as indicated in Figure 4. Note however that on the first time step of any sequence ($t=1$), there is no horizontal input present. In this case, χ depends only on Ψ. This multiplication results in a sharpening of the posterior probability (or likelihood) over the hypothesis space, as in the Product of Experts model (Hinton, 1999) and consistent with the ideas exposited by Lee & Mumford (2002).

$$\chi_t(i) = \begin{cases} \Psi_t(i)^v, & t=1 \\ \Phi_t(i)^u \; \Psi_t(i)^v, & t>1 \end{cases} \qquad (6)$$

Let $\hat{\Phi}_{t,z}$ (depicted near the top right of Figure 4) denote the normalized H input vector over the units of the z^{th} CM. $\hat{\Phi}_{t,z}$ represents the temporal context of (history leading up to) the current moment, t, and is a function of the specific set of L2 units active at $t-1$. Similarly, $\hat{\Psi}_{t,z}$ (near the bottom right of Figure 4) denotes the normalized BU input vector over the units of the z^{th} CM. $\hat{\Phi}_{t,z}$ and $\hat{\Psi}_{t,z}$ are independently exponentiated, typically by some small integer (e.g., 2 or 3), to effect separate generalization gradients for each influence. This further sharpens the posterior.

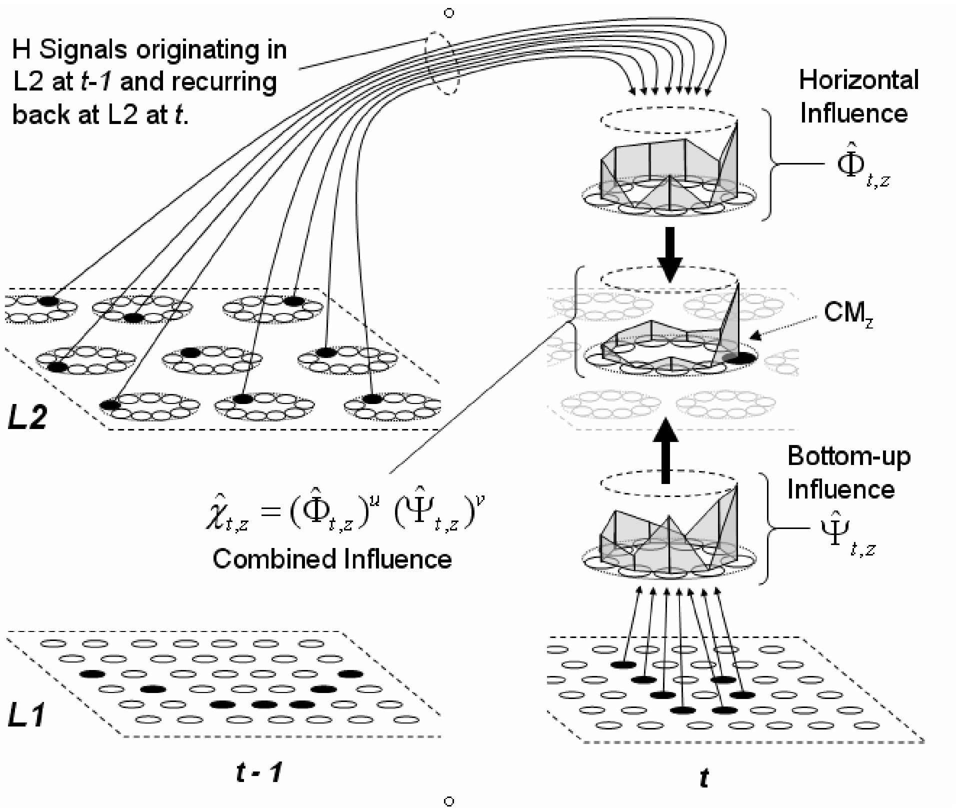

H Signals originating in L2 at *t-1* and recurring back at L2 at *t*.
Horizontal Influence
$\hat{\Phi}_{t,z}$
CM_z
L2
Bottom-up Influence
$\hat{\Psi}_{t,z}$
$\hat{\chi}_{t,z} = (\hat{\Phi}_{t,z})^u \, (\hat{\Psi}_{t,z})^v$
Combined Influence
L1
t - 1
t

Figure 4: $\hat{\Psi}_{t,z}$, depicts a hypothetical vector (distribution) of normalized BU inputs (Ψ values) to the units of CM_z at time *t*. The dashed ellipse represents a Ψ value of 1.0. Similarly, $\hat{\Phi}_{t,z}$ depicts a vector of normalized H inputs for CM_z. $\hat{\Psi}_{t,z}$ and $\hat{\Phi}_{t,z}$ are exponentiated (see text) and then multiplied together yielding a final combined local degree of support distribution, $\hat{\chi}_{t,z}$, over the units of CM_z.

The model's ability to easily represent very long-distance temporal dependencies derives from the huge space of representations available for representing moments. In the small toy example model of Figure 3, whose layer 2 has only four CMs each with four units, there are $4^4 = 256$ possible L2 codes, which may not seem like all that much. However, the CM is envisioned as analogous to the cortical minicolumn (Mountcastle, 1957; Peters & Sethares, 1996), or more specifically, to that subset of a minicolumn corresponding to its principal representing cells, e.g., to a minicolumn's approximately 30 layer 2/3 pyramidals. Further, the model's L2 is envisioned as a patch of cortex on the order of a hypercolumn, which subsumes approximately 70 minicolumns. Such a situation, depicted in Figure 5, yields an astronomical number, i.e., 30^{70}, of unique codes. This is not to say that all 30^{70} codes can be used. Because the model's learning rule increases all weights from one L2 code to the next, complete saturation of the H weight matrix, and thus total loss of information, would occur long before all 30^{70} codes were assigned. Just how large a fraction of the codes could be used while maintaining a given fraction of the total stored information is the issue of storage capacity. Although an analytic storage capacity result is not given here, empirical results for a simpler version of the model are given in the appendix. These results offer indirect evidence for substantial capacity, on the order of 0.135 bits/synapse, as the problem is scaled to arbitrarily large size.[5]

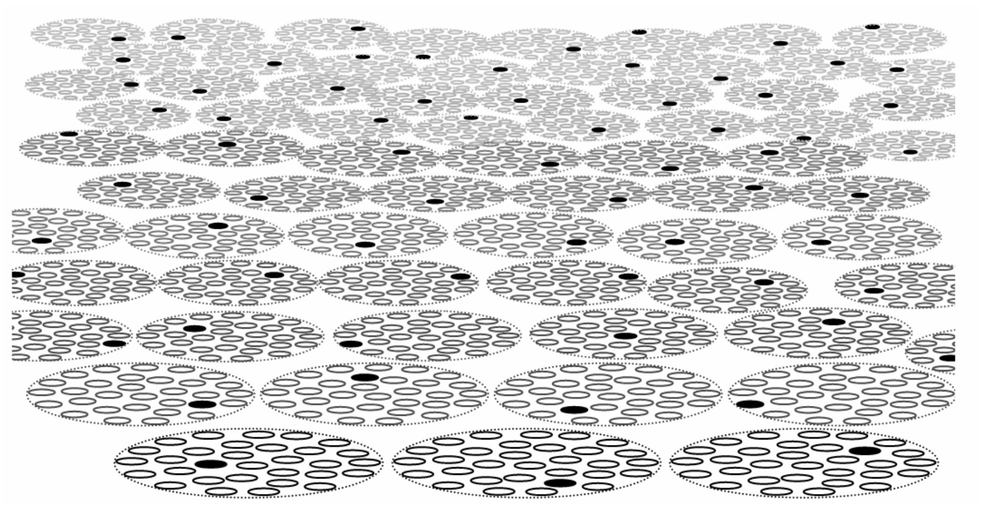


Figure 5: Diagram reflecting the approximate parameters of a hypercolumn-sized patch of cortex, e.g., about 70 minicolumns where each minicolumn contains about 30 representing cells, i.e., the layer 2/3 pyramidals. The code shown (set of black cells) is therefore one out of a possible 30^{70} codes.

In the next step of the learning algorithm, Eq. 7, the model simply finds the maximum χ value, $\overline{\chi}_{t,z}$, in each CM, z. Then, in Eq. 8, the model computes the maximum similarity, G, of the current moment, i.e., the current spatial input in the temporal context of the sequence leading up to the current input, to *all previously experienced moments*. That is, the model effectively evaluates the likelihoods *all* stored hypotheses in parallel. G is normalized between 0 and 1 and is defined as the average of the maximal

[5] The possibility of such a vast amount of codes (e.g., 30^{70}) provides a possible answer to the *binding* problem. It is simply that there are enough unique codes available to explicitly represent the combinatorially large number of possible input configurations for reasonably constrained input spaces. The same basic idea was proposed in O'Reilly et al., 2003.

local match values in each CM. It can be thought of as a *global match* (familiarity) signal. If G = 1, then the current moment is identical to some previously experienced moment. If G is close to 0, then the current moment is completely unfamiliar (novel).

$$\overline{\chi}_{t,z} = \max_{i \in CM_z} \chi_t(i) \tag{7}$$

$$G_t = \frac{\sum_{z=1}^{Q} \overline{\chi}_{t,z}}{Q} \tag{8}$$

In the last phase of the learning algorithm, the model uses the *global* familiarity measure, *G*, to nonlinearly transform the *local* degree of support values, χ, into final probabilities, ρ, of being chosen winner. Eqs. 9, 10 and 11 together define how *G* modulates, in a graded fashion, the character of that transform. The precise parameters of these equations are not particularly important: they provide various means of controlling generalization gradients and the average separation (i.e., Hamming distance) between codes. What is important is the overall effect achieved, i.e., that code overlap is an increasing function of spatiotemporal similarity.

$$a_t = \left(\left[\frac{G_t - \chi_\alpha}{1 - \chi_\alpha} \right]^{+} \right)^{d} \times C \times K \tag{9}$$

$$\xi_t(i) = \frac{a_t}{1 + e^{-b(\chi_t(i) - c)}} + 1 \tag{10}$$

$$\rho_t(i) = \frac{\xi_t(i)}{\sum_{k \in CM} \xi_t(k)} \tag{11}$$

Once, the final ρ-distribution is determined (Eq. 11), a choice of a single winner (in each CM) is made according to it. The behavior of the model in this last phase is best described in terms of three different regimes: $G \approx 1$, $G \approx 0$, and *G* somewhere in the middle. A *G* value close to 1 means that there is a particular previously experienced (stored) moment, σ^*, that is extremely similar to the current (query) moment and therefore is causing the high *G* value. In this case, the model should reinstate σ^*'s code, Δ_{σ^*}, with probability very close to 1.0. This is achieved by adjusting the transform's parameters so that it is a highly expansive nonlinearity, as shown in Figure

6a. This effectively causes units with the highest χ values in their respective CMs to win with probability close to one, and therefore Δ_{σ^*}, as a whole, to be reinstated with probability close to one. In contrast, when *G* is near 0, the transform becomes a completely flat constant function (Figure 6b), which assigns equal probability of winning to all units in a module. This minimizes the average overlap of the set of winners (which is a new code) and all previously stored (known) codes.

For middling *G* values ranging from 0 up to 1, the transform gradually morphs from the perfectly compressive constant function of Figure 6b into a highly expansive sigmoid as shown in Figure 6a. The higher the value of *G*, the larger the average overlap of the winning code and the closest matching stored code. Thus, this *G*-dependent gradual morphing of the transform directly yields the desired property that spatiotemporally similar moments map to similar codes.

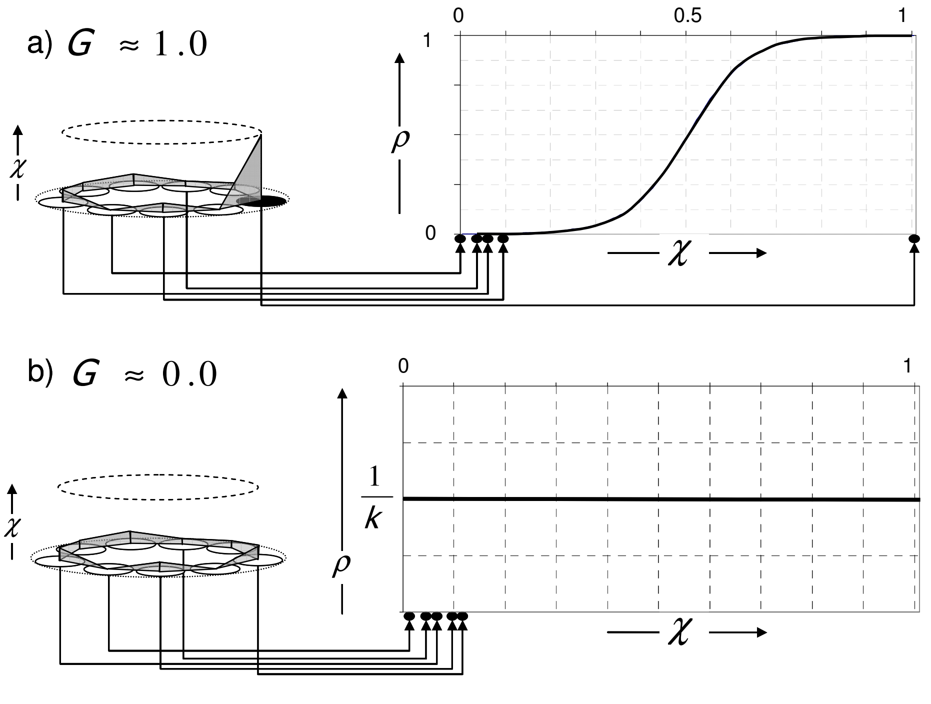


Figure 6: Mapping local similarity values (χ values) to final probabilities of being chosen winner (ρ values).

Note that in the $G \approx 1$ regime, the model is actually *not* in a learning mode. That is, because the choice of code depends on the product of H signals from the previously active code and BU signals from the input, exact reinstatement of a stored code implies that both the prior code and the current input are as they were on the past occasion that is so similar to the current moment (i.e., causing $G \approx 1$). That means that there are no novel pairings, *{i,j}*, of active units, such that unit *i* is active on the prior time step and unit *j* is active on the current time step, and thus that there are no new opportunities for synaptic increases. Thus, no learning can occur. The expected number of novel pairings and therefore the amount of learning that will occur gradually increases as *G* drops to zero. Thus, *G* effectively automatically controls the model's movement along the continuum from learning to recalling/recognizing.

Figure 7 depicts two hypothetical situations corresponding to $G \approx 1$ and $G \approx 0$. There is a clear maximal χ unit in each CM in Figure 7a. However, note that even in Figure 7b, there will generally still be a maximal χ unit in each CM (although it might not be distinguishable in this picture). In general, early in the model's learning period (life) the set of maximal χ units, on each successive moment, will be identical to some previously stored code. This is true regardless of the value of *G*. This is why the model cannot simply choose the maximal χ unit in each CM as the final winner. Doing so would lead to assigning almost all moments to a tiny number of codes (even a single code), essentially losing almost all information about the data set.

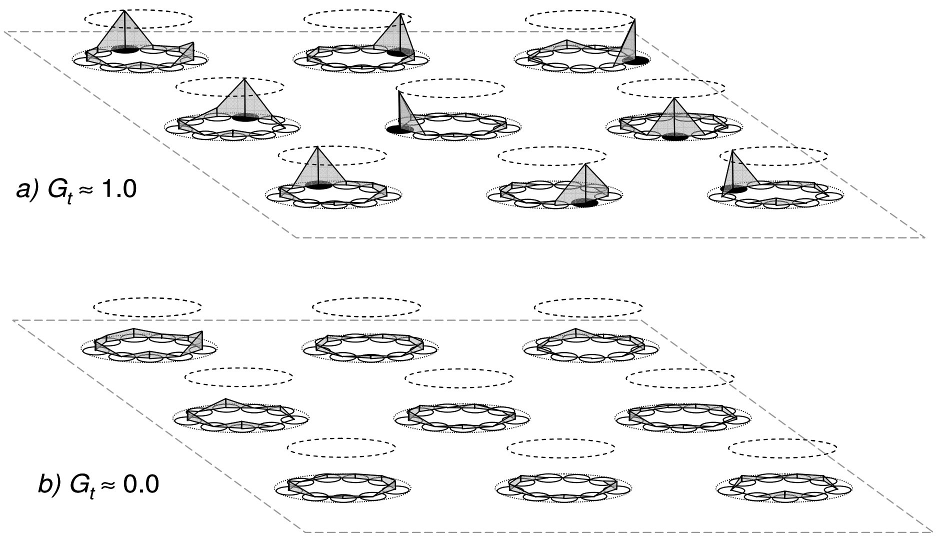


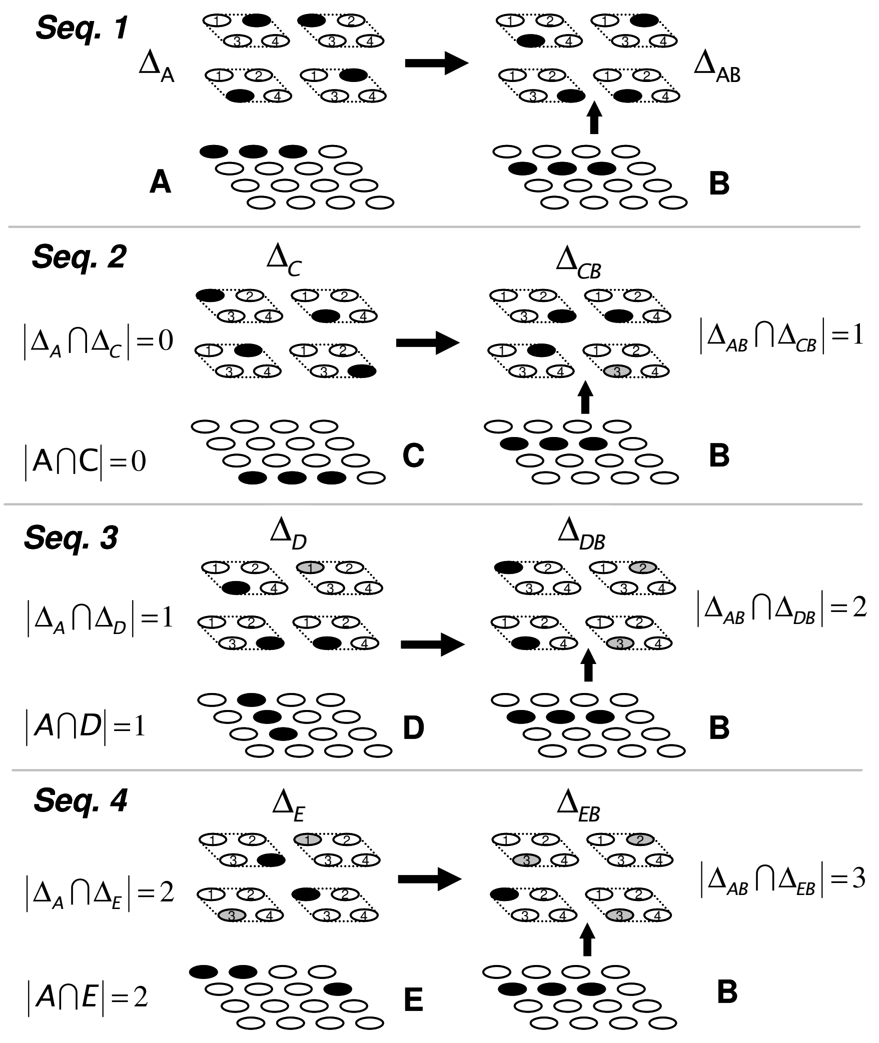
Figure 7: a) Hypothetical instance where $G \approx 1$. b) An instance where $G \approx 0$.

This last point in the discussion of the learning algorithm bridges naturally to the discussion of retrieval, i.e., recall and recognition. For when *G* is near 1, it *is* safe, in fact, optimal, to simply choose the max χ unit in each CM to become active: the winner selection process becomes one that can be purely local and still be optimal. By 'optimal', I mean that it retrieves the most similar stored moment to the current query moment, or in other words, retrieves the maximum likelihood hypothesis. Thus, during recall and recognition, at each moment, the model simply activates the maximal χ unit in each CM. The recognition algorithm consists of Eqs. 2-6, followed by activating the max χ unit in each CM. The retrieval algorithm is slightly simpler because, following presentation of the first item (i.e., the prompt), only H signals are used to determine which L2 units become active at each successive moment. Hence, the retrieval algorithm consists of Eqs. 2-5, followed by Eq. 6', and then by activating the maximal χ unit in each CM. Once the L2 code is activated, each L1 unit computes its total TD input. Those whose TD input exceeds a threshold, typically set slightly lower than the number of L2 CMs, Q, become active, thus replaying the original sequence at L1.

$$\chi_t(i) = \begin{cases} \Psi_t(i), & t = 1 \\ \Phi_t(i), & t > 1 \end{cases} \tag{6'}$$

An extended example follows, which illustrates two of the model's key properties: a) spatiotemporally similar moments map to similar codes, and b) the computation process for finding the closest matching stored moment does not depend on the number of moments (sequences) stored. This hypothetical example (Figure 8) shows the learning of the memory traces for four 2-item sequences by a small instance of the model.

When the first item, A, of Sequence 1 presents all BU weights are zero. All raw (ψ), and thus normalized (Ψ), BU input values are also zero. Thus, all local match (χ) values are zero, and thus, the global match (G) is zero. Accordingly, the χ values are mapped through a constant function, making all units in any given CM equally likely to win. The top panel shows a randomly chosen code, Δ_A, for item A. Learning would occur at this point: the BU weights from the units of A to the units of Δ_A would be increased to a weight of 1. The second item, B, presents on the next time slice. Since $|B \cap A| = 0$, none of the BU weights from the units of B have been increased yet. Furthermore, no H weights have been increased yet. Since we are beyond the first time slice of the sequence, χ is computed as the product of normalized H and BU (Φ and Ψ) inputs. However, since both ψ and ϕ are zero for all L2 units, so are all Φ, Ψ, and χ values, and so is G. Again, all units end up being equally likely to win. Another randomly chosen code, Δ_{AB}, is shown for item B (i.e., for moment [AB]). At this point, learning occurs in the H weight matrix from the previously active units of Δ_A to the units of Δ_{AB}, and in the BU matrix from the units of B to the units of Δ_{AB}.



Seq. 1
Δ_A
Δ_{AB}
A
B
Seq. 2
Δ_C
Δ_{CB}
$|\Delta_A \cap \Delta_C| = 0$
$|\Delta_{AB} \cap \Delta_{CB}| = 1$
$|A \cap C| = 0$
C
B
Seq. 3
Δ_D
Δ_{DB}
$|\Delta_A \cap \Delta_D| = 1$
$|\Delta_{AB} \cap \Delta_{DB}| = 2$
$|A \cap D| = 1$
D
B
Seq. 4
Δ_E
Δ_{EB}
$|\Delta_A \cap \Delta_E| = 2$
$|\Delta_{AB} \cap \Delta_{EB}| = 3$
$|A \cap E| = 2$
E
B

Figure 8: Hypothetical example of the first four sequences learned by a model, illustrating the property that similar spatiotemporal patterns (sequences) map to similar internal representations (codes). Gray units denote intersections with Δ_{AB}.

Next, Sequence 2 presents to the model. The first item, C, is unfamiliar. None of C's features have been present thus far. Again, $G = 0$; thus another random code, Δ_C, is chosen. Then the second item, B, presents. Item B has been experienced before but in the context of a different predecessor. It seems reasonable that the model should remember this novel moment, [CB], as distinct from [AB]. Thus, it should choose a novel L2 code, Δ_{CB}, having little (or at least, non-total) overlap with any previously chosen L2 code. While [CB] clearly has some similarity to the previously experienced moment, [AB], all χ values would in fact be zero in this case (because the H inputs to all L2 cells would be zero). Thus, Δ_{CB} is chosen randomly: let's assume that, by chance, it has one unit (gray cell in the lower right CM) in common with Δ_{AB}.

Now Sequence 3 presents. Item D has one feature in common with A (as well as one in common with B). Various L2 units will thus have non-zero ψ, and thus non-zero Ψ and χ values. However, none of the χ values will be close to1; thus, G will be non-zero but closer to zero than one. The χ-to-ρ transform will thus have some expansivity and therefore cause some bias favoring the units with higher χ values, though there will still be significant randomness (noise) in the final choice of winners. Consequently, we show the code, Δ_D, as having one unit in common with Δ_A. Then item B presents following D. This moment, [DB], has greater spatiotemporal similarity to [AB] than [CB] did. This increased similarity is manifest neurally in the unit that is common to Δ_A and Δ_D, which means that various L2 units will have non-zero ϕ, and therefore non-zero Φ, values at this moment. Item B is familiar: thus various L2 units will have non-zero ψ, and thus Ψ, values. Finally, some L2 units will have a non-zero product, χ, of Φ and Ψ values; thus G will also be non-zero. Again, the transform will have some expansivity and thus cause some bias favoring units with higher χ values. Thus, we depict a code, Δ_{DB}, having two units in common with Δ_{AB}.

Finally, Sequence 4 presents. It's first item, E, is even more similar to A than D was (2 out of 3 features in common). Without belaboring further, I will just state that this increased similarity leads to a code, Δ_E, having two units in common with Δ_A, and that this in turn leads to a code, Δ_{EB}, that has even more overlap with Δ_{AB} than did Δ_{DB}. Overall, this example shows how the model maps the spatiotemporal similarity of moments to the (spatial) similarity of codes. This is the crucial property of the model, which makes possible the $O(1)$ retrieval complexity.

Figure 9 demonstrates the correctness of the model's retrieval strategy—i.e., simply picking the max χ unit in each CM to win retrieves the most similar stored moment—for the first moment [A] of Sequence 1. To show this, I introduce a variant of the G measure, the *code-specific G*, $G(\Delta_x)$, which Eq. 12 defines as the average of the χ values *for the units comprising a specific code*, Δ_x, rather than the average of the maximal χ value in each CM. The term, $\Delta_{x,z}$, in Eq. 12, denotes the winning unit in CM z during moment x. Each panel of Figure 9 shows one of the eight L2 codes stored by the model during learning, along with item A at L1. The χ values, arising when item A presents, for each of the code's units is shown. The average of those four χ values, i.e., $G(\Delta_x)$, is shown at the top of each panel. Since this is the fist time step of the retrieval event (query), the χ values equal the Ψ values (Eq. 7, with exponent $v = 1$).

$$G(\Delta_x) = \frac{\sum_{z=1}^{Q} \chi(\Delta_{x,z})}{Q} \tag{12}$$

The main points here are: a) the code-specific G, $G(\Delta_A)$, for the correct code, Δ_A, is maximal; and b) the code-specific G's for the rest of the codes correlate roughly with the similarity of the moments that they represent and the query moment, [A]. For example, $G(\Delta_A) > G(\Delta_E) > G(\Delta_D) > G(\Delta_C)$, and by construction of the example, $|A \cap A| > |A \cap E| > |A \cap D| > |A \cap C|$. The code-specific G can be thought of as the relative likelihood of the hypothesis represented by the code.

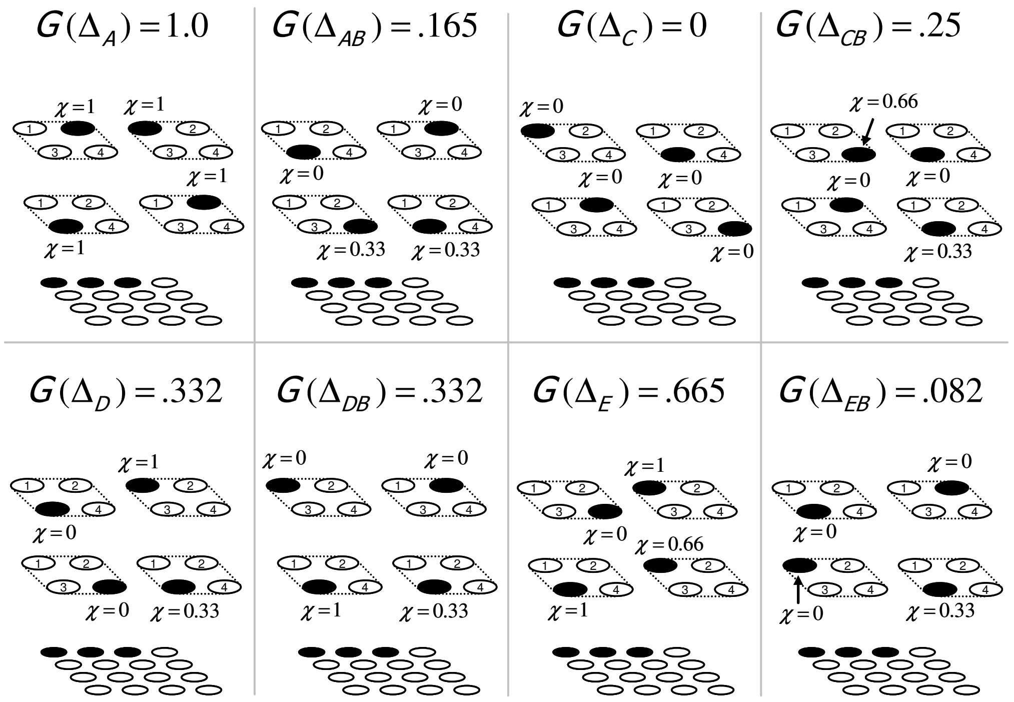


Figure 9: The code-specific G values (relative likelihoods), when the query moment is [A], of all eight moments (hypotheses) stored in the model. Notice that the stored representation of [A], Δ_A, has the highest code-specific G.

Figure 10 shows the code-specific G values for all eight codes (hypotheses) when the query moment is [AB]. Thus, this is a recognition test in which item A was presented on the prior time step and item B is now being presented. This figure reinforces the same two main points as Figure 9. The correct code, Δ_{AB}, has the highest code-specific *G*, and the other codes' code-specific *G*'s fall off roughly with spatiotemporal similarity to the query moment, [AB]. For example, $G(\Delta_{AB}) > G(\Delta_{EB}) > G(\Delta_{DB}) > G(\Delta_{CB})$. Again, by construction of the example, $sim([AB],[AB]) > sim([AB],[EB]) > sim([AB],[DB]) > sim([AB],[CB])$, where '$sim(\sigma_i, \sigma_j)$' is the simple spatiotemporal similarity metric, Eq. 13, that depends only on the total number of common features across all *T* time steps.

$$sim(\sigma_i,\sigma_j) = \sum_{t=1}^{T} \left| \sigma_i^t \cap \sigma_j^t \right| \tag{13}$$

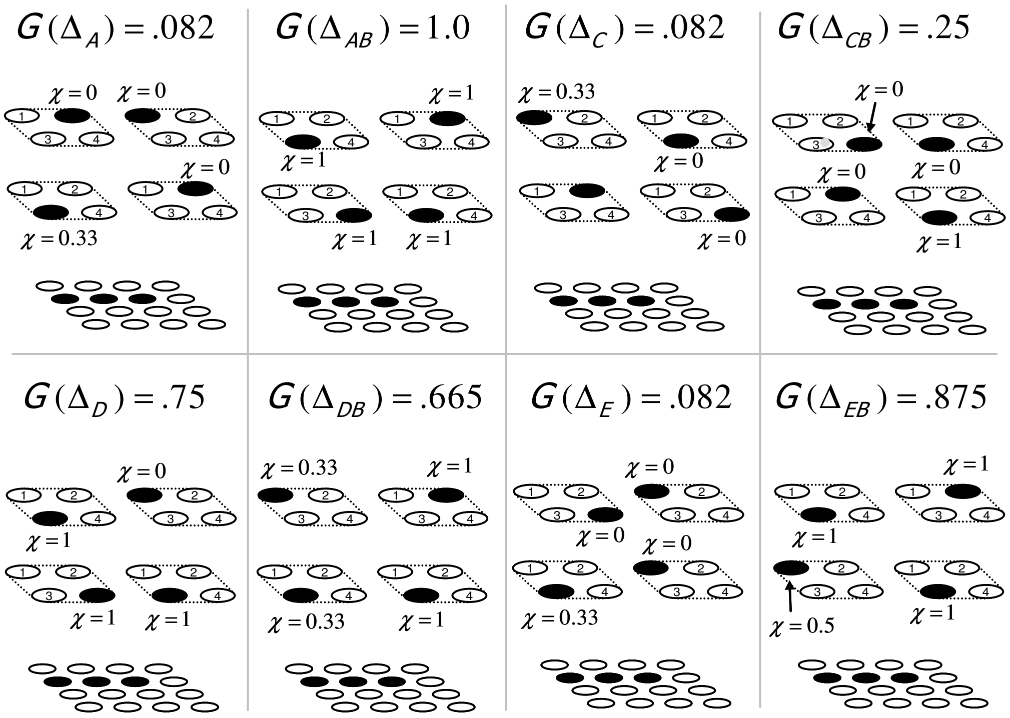


Figure 10: The code-specific *G*'s, when the query moment is [AB] (i.e., when B is presenting right after A presented), for all eight stored moments.

Table 1 provides the code-specific *G* (relative likelihood) value for each of the eight stored moments (hypotheses) given each of those same moments as queries. The data in the first two rows comes from Figures 9 and 10, respectively. The table demonstrates that the same properties shown in Figures 9 and 10 hold generally for the learning set as a whole. The most profound aspect of the model is that the model finds the most similar stored moment without explicitly computing these code-specific *G* values explicitly. In fact, the model does not even have to compute *G*, i.e., Eq. 8, during recall and recognition (i.e. during perceptual inference). Because the way in which codes were chosen during learning maps spatiotemporal similarity of moments into degrees of code overlap and thus automatically and implicitly embeds the higher-order statistics (spatiotemporal similarity structure) of the sequence set in the model's weight matrices, simply choosing the maximal χ unit in each CM is guaranteed to reactivate the code of the most similar stored moment, provided the query moment is close enough to that stored moment.

Table 1: Code-Specific G Values (Relative Likelihoods) of all Stored Moments (hypotheses) given those same Stored Moments as Queries

	Hypothesis							
Query	Δ_A	Δ_{AB}	Δ_C	Δ_{CB}	Δ_D	Δ_{DB}	Δ_E	Δ_{EB}
A	1.0	0.165	0	0.25	0.332	0.332	0.665	0.082
AB	0.082	1.0	0.082	0.25	0.75	0.665	0.082	0.875
C	0	0	1.0	0.25	0.75	0.665	0.082	0.875
CB	0	0.25	0.5	1.0	0.25	0.25	0.25	0.25
D	0.5	0.832	0.25	0.58	1.0	0.5	0.665	0.665
DB	0.25	0.665	0.332	0.25	0.415	1.0	0.25	0.25
E	0.83	0.25	0	0.332	0.5	0.332	1.0	0.165
EB	0.082	0.75	0.082	0.25	0.5	0.665	0.082	1.0

What Table 1 does not demonstrate directly is that novel sequences (moments), that are perturbations (noisy versions) of the learning set sequences, will be mapped to the spatiotemporally most similar stored moment. However, the demonstrated code-specific *G* gradients are, in some sense, the dual of that result. In any case, the simulation results in the following section do provide a direct demonstration of this property.

4. Simulation Results

The global familiarity, *G*, could be used as a signal that tells the model whether it is in storage (learning) or retrieval (recall, recognition) mode. However, the results reported here were produced using a simpler protocol in which the operational mode is mandated externally. Thus, these experiments have a learning phase in which the set of sequences is presented one time each and then a second test phase that is either a recall test or a recognition test. Weights are frozen during the test phase.

In recall tests, the model is *prompted* with just the first item from one of the sequences that it learned. This prompt item is also referred to as the *query*, or *query moment*. In this case, the model returns the remainder of the sequence. This can be thought of as a demonstration of *episodic* recall of a sequence. For recall tests, we report the recall accuracy at both layers. To compute the accuracy, $R_2(t)$, at L2, we compare the code at time *t* during the recall test to the code that occurred at time *t* during learning, as defined in Eq. 14. *E(t)* is the number of CMs in which the unit activated in recall differs from the winner during learning, i.e., errors. A sequence's overall L2 accuracy is the average L2 accuracy over all time steps (items) of the sequence, excluding the prompt time step. The formula for L1 recall accuracy, $R_1(t)$, is slightly more complicated since, at L1, the two kinds of errors, deleted features, *D(t)*, and intruding features *I(t)*, must be kept track of separately. The L1 code reinstated at time *t* during recall is compared to the L1 code that occurred at time *t* during learning, as defined in Eq. 15. *C(t)* is the number of correct features at L1 at time *t* during recall.

$$R_2(t) = \frac{Q - E(t)}{Q} \tag{14}$$

$$R_1(t) = \frac{C(t) - D(t)}{C(t) + I(t)} \tag{15}$$

In recognition tests, the model is presented with an entire sequence. This *query sequence* could be identical to one of the learning sequences or it could be a perturbed version of one of the learning sequences. In the former case, the model should reinstate the same sequence of L2 codes (the memory trace) that occurred during the learning trial. In the latter case, the model should activate, on each successive moment, the L2 code corresponding to the stored moment that is spatiotemporally most similar to the current query moment. This demonstrates recognition (classification) of a sequence. For recognition tests, we only report the L2 accuracy because the L1 code is supplied from the environment on all time steps of a recognition trial: thus, there is no notion of an error at L1 in a recognition test. The recognition accuracy at L2 is also computed using Eq. 14. In this case, the comparisons are made with the stored sequence from which the perturbed test sequence was generated.

Several parameters are common to all three simulation experiments. In all cases, the network's L1 consisted of 100 binary feature detectors; the L1 and L2 units were completely connected in both the BU and TD direction; and the L2 units were completely connected via the horizontal weight matrix. All weights are binary and initialized to zero.

Experiment 1

Experiment 1 provides a small proof-of-concept demonstration that a single network can learn a set of sequences with single trials and subsequently both recall all of the individual sequences essentially perfectly and also recognize substantially perturbed (very noisy), and thus, novel, instances of the learning set sequences. The learning set, shown in Figure 11, contained five sequences, each having five items. Each item consists of ten randomly selected features. Thus, there are 25 unique moments. L2 consisted of eight CMs, each with 10 units. The recall accuracy for this set was nearly perfect; 98.18% at L2 and 100% at L1. Of the 400 re-activations of L2 units that occur over the course of recalling all five sequences, only eight errors occurred.

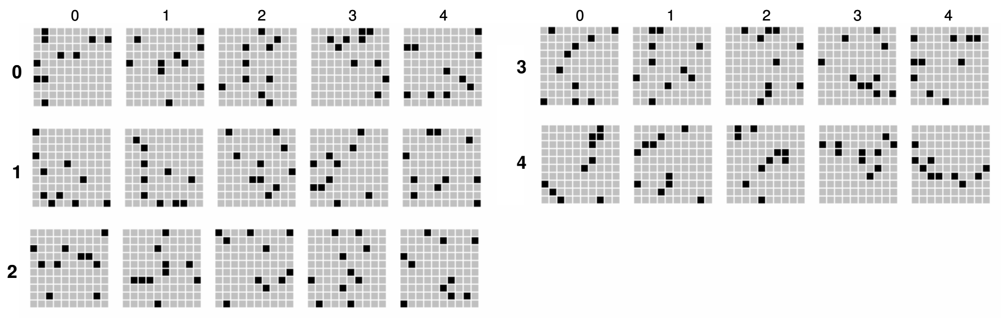


Figure 11: The five randomly created sequences comprising the learning set for Experiment 1. Item (time step) indices are across the top.

Figure 12 shows the complete learning and recognition traces, at both L1 and L2, for the first sequence. The top row of each trace shows the L2 codes. Each row of small gray squares in an L2 grid corresponds to a CM, and each individual square in the row, to a single unit within that CM. Black squares are the winners. Notice that the trace includes two sub-steps for each item of the sequence: e.g., item A is actually presented twice, at sub-steps 0 and 1, item B is presented at sub-steps 2 and 3, etc. This reflects the computer simulation's property that signals originating from a given input item are processed up through all layers (in this case, there are only two) before the next item is presented. Thus, on the first sub-step of each item, the input pattern itself becomes active at L1 and on the second sub-step, the BU signals from that L1 code reach L2 and activate an L2 code. Notice also that the L2 codes also stay active for two sub-steps, although they are delayed one sub-step from the L1 codes. This is the more complex representation of time alluded to in an earlier footnote and it entails modifying Eq. 3 to the following. The only difference is that the summation is over signals arriving from L1 units active on the prior sub-step, i.e., the set, Υ_{t-1}, rather than the current sub-step.

$$\psi_t(i) = \sum_{k \in \Upsilon_{t-1}} w_{ki} \tag{3'}$$

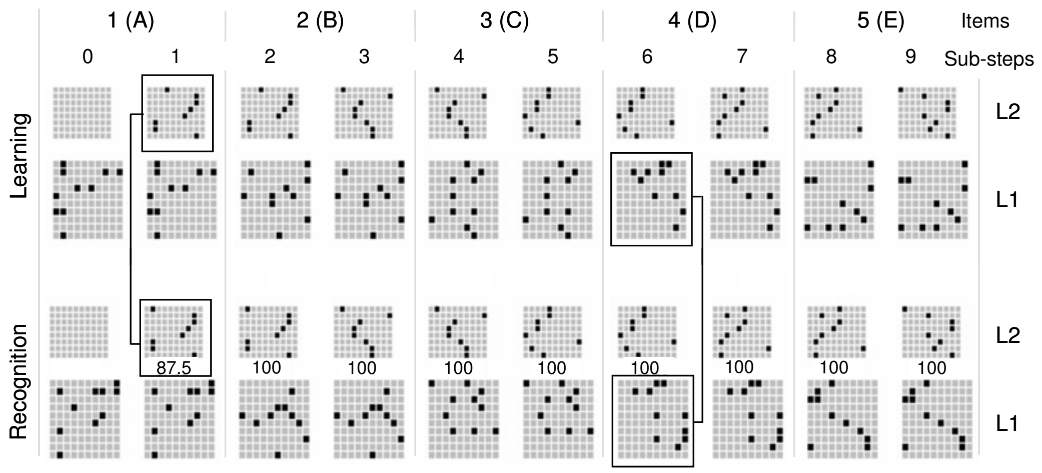


Figure 12: (top) The complete learning trace for Sequence 1 for both L1 and L2. (bottom) The complete recognition trace for a perturbed version of Sequence 1 which had 40% (4 out of 10) features changed on each item. The numbers below the L2 codes in the recognition trace are R_2 scores (percentages). The R_2 value of 87.5% on sub-step 1 means that the correct winner was reinstated in 7 out of the 8 CMs. See text for further discussion.

Recognition accuracy was also extremely good: 94.78% averaged over all five perturbed sequences. Note that only the final (in this case, 2nd) sub-step of each item is included in the computation of a sequence's recognition accuracy. The intuition here is that as long as the recognition L2 code tracks the original (learning) L2 code on the item timescale (as opposed to the finer sub-step timescale), the model should be seen as demonstrating recognition. These five sequences had a 40% perturbation rate: i.e., 4 out of the 10 features comprising each of the learning items were randomly changed. This very substantial degree of perturbation can be seen by visually comparing two L1 codes [the original learning item (top) and its perturbed variant (bottom)] as in the vertically connected boxes on the right of in Figure 12. This trace, typical of the whole set, shows that despite the constant extreme perturbation (noise) on every item of the recognition test sequence, the model reactivates the exact L2 code of the most similar stored moment on each sub-step throughout the whole recognition test sequence, except for one error on sub-step 1 (see the vertically connected boxes on the left; the error is in the CM represented by the top row of the L2 grid). That is, the model co-classifies the variant instance of Sequence 1 with the most similar stored sequence, which is Sequence 1.

Neither the learning algorithm (Eqs. 1-11) nor the retrieval algorithms (Eqs. 2-6 and Eqs. 2-6') depend on the number of sequences (or, moments) previously stored. Therefore, both learning and retrieval time are constant; i.e., $O(1)$ time complexity.

Experiment 2

The second experiment provides a larger example. In this case, there were 10 sequences, each with 10 items. However, this network's L2 is much larger: 9 CMs with 26 units each. Again, recall accuracy was virtually perfect: 99.05% at L2 and 100% at L1. The set of learning sequences, as a whole, contained 4616 bits of information and the network had 95,472 weights, yielding an information storage rate of 0.0483 bits/synapse. While this is about an order of magnitude below the theoretical maximum for sparse binary associative memories, these experiments are intended as a simple proof of concept. Numerous modifications and optimizations are possible and will be explored in the future (some are briefly described in the discussion section and appendix). Recognition accuracy was also extremely good: 94.78% averaged over all ten perturbed sequences. These sequences had a 30% perturbation rate: i.e., 3 out of the 10 features comprising each of the learning items were randomly changed. As in Experiment 1, learning and retrieval are $O(1)$. In this case, there are 10 x 10 = 100 stored moments whose representations compete at each query moment.

5. Future Work

As presented herein, the model has a significant problem learning a large set of sequences in which many sequences start with the same item or prefix of items, e.g., all the words of some natural language. The problem is *not* that the model has any intrinsic difficulty dealing with a set of complex sequences, i.e., a sequence in which items can recur many times and in many different contexts. As the example of Section 3 shows, the choice of representation for an item depends on the prior context. Thus, there are four different codes for item B in that example. In general, this allows the model to move through a common state to the correct successor state.

Rather, the problem is simply that far too small a set of L2 codes will ever be used to represent the initial moment of sequences in domains such as a natural language lexicon. For example, consider encoding the approximately 100,000 words of the English. Since there are only 26 letters, all 100,000 stored representations of the words will begin with one of 26 initial L2 codes. Moreover, the hundreds of words that begin with 'IN', for example, will begin with the same two L2 codes, etc. The result is that the proportion of horizontal (H) synapses both to and from the L2 cells in these 26 codes will be very much higher than average. In other words, the subset of the H weight matrix impinging on the relatively small fraction of L2 cells used to represent the beginnings of words will become saturated quickly. The integrity of this model, as of any sparse distributed associative model, depends on operating substantially below the saturation regime.

Thus, a central problem, which is currently being explored, is to find a way to use a much larger set of L2 codes for sequence-initial items (or prefixes). Essentially, what is needed is a way to make the selection of the L2 code for a sequence's initial item depend on one or more of the subsequent items. Thus, some way of maintaining a representation of the most recent several items, i.e., a short-term memory, is needed. One possible solution is briefly sketched here.

This solution requires three significant modifications to the model as presented herein: a) generalization to an arbitrary number of internal layers; b) a relaxation or generalization of the winner-take-all nature of the CM; and c) a learning rule for decreasing synaptic weights in certain circumstances. Regarding modification (a), cells at progressively higher internal levels would have progressively longer intrinsic activation durations (persistences); e.g., L3 cells would persist for twice as long as L2 cells, L4 cells would persist for twice as long as L3 cells, etc.[6] Regarding modification (b), on the first time step of a sequence, two winners would be allowed in each L3 CM instead of one. On the second time step of a sequence, the model would revert to the one-winner-per-CM rule.

These modifications allow the following solution. Suppose that the first item, 'I', of the word 'INTO' presents. An L2 code is chosen and an L3 code, with two winners in each CM, is also chosen. At the end of this first time step, learning takes place between L3 and L2 and between L2 and L1. Then 'N' presents. An L2 code is chosen. However, at L3 two considerations apply: a) the L3 code should remain since L3 cells have twice the persistence of L2 cells; and b) L3 reverts to having only one

[6] As noted earlier, TEMECOR has, in fact, already been generalized to having an arbitrary number of layers with this progressive persistence property. The generalized model demonstrates robust invariance to non-uniform time warping and retains its O(1) time complexity for learning, recall, and recognition.

winner per CM on this time step. This can be resolved by allowing one of the two active L3 cells in each CM to remain active on this second time step. And crucially, the second L2 code (for 'N') can, via its BU inputs to L3, have a causal influence on which subset of the L3 code remains active. At the end of the second time step, after the L3 code has been reduced to one winner per CM, learning takes place between this reduced L3 code and the new L2 code. Additionally, any *recently increased* weights to and from the L3 cells that were just turned off could be reduced. Thus, the learning between the subset of L3 cells *not* chosen to remain active and the L2 code for the *first* item ('I') could be reversed. The overall effect is that a one-winner-per-CM L3 code that was assigned on the *first* time step but that depends on the first *two* time steps of a sequence has been embedded. If we next presented the sequence, 'IT', then by the above reasoning, a different L3 code would be assigned. This is only a brief sketch of a possible solution to the problem of over-utilizing sequence-initial codes. It is promising because it utilizes only known or plausible neural properties (hierarchality, progressive persistence (Uusitalo, et al., 1996), some form of synaptic weight decrease) and only locally available information.

6. Discussion

The results show that TEMECOR can learn a set of sequences with single-trials and automatically embed the higher-order statistics of the input set in the process. The model's $O(1)$ computational time complexity for both storage and retrieval follows immediately from the fact that none of its equations have any dependence on the number of stored traces. That is, regardless of how many sequences are stored so far, both the time it takes to store another sequence and the time it takes to find the most similar stored sequence (or partial sequence) to a query (which is also a sequence or partial sequence) remain constant. The model thus possesses optimal computational time complexity, within a multiplicative constant. While I do not yet have a definitive analytic result on the model's computational spatial complexity, an indirect argument suggesting good scaling to large problem sizes is given in the appendix.

The 'similar inputs to similar representations' (SISR) property is achieved by TEMECOR's novel mechanism that injects an amount of noise that is inversely proportional to the G—which is the maximum spatiotemporal similarity of the current moment to all stored moments, into the winner selection process. The SISR property, in conjunction with its sparse distributed representation scheme, yields the robust generalization and categorization capability evidenced by the simulation results, i.e., co-categorizing novel sequences with up to 40% per-item degradation (noise) relative to the most similar known sequence. Note that this capability obtains even though all that the model explicitly does during learning is store individual sequences, each presented only once. This is consistent with the naturalistic way in which humans learn as opposed to the *concurrent* learning paradigm, employed in Backpropagation models, where all exemplars in the data set must be learned in parallel (i.e., every exemplar is present in every *epoch*) and each exemplar is presented numerous (e.g., 100's) times (French, 1991).

The network is a single monolithic substrate/mechanism in which are stored both the individual traces and the higher-order statistics. The simulation results show that that single substrate yields both episodic recall and generalization depending on how the model is prompted. If it is given an exact duplicate of a sequence's initial item as a prompt, then it can return an exact duplicate of the rest of the sequence. Here, the query and the original have identical statistics of all orders present. On the other hand if we give it a degraded version of a portion of the sequence (in our tests, we supplied degraded versions of all items of a sequence), then it returns the spatiotemporally most similar stored sequence. Here, the query and original share only some subset of their higher-order statistics.[7] Thus, the model simultaneously explains both episodic recall and generalization and categorization.

There have been few proposals of such monolithic explanations for these two kinds of memory (McClelland & Rumelhart, 1985). Rather, the prevailing view of the relevant research community is that episodic memory has more to do with the hippocampus while generalization and categorization (i.e., representing and responding on the basis of higher-order statistics) have more to do with neocortex (O'Reilly & McClelland, 1994). Of course, there is a great deal of evidence consistent with this viewpoint, going back at least as far as H.M. (Scoville & Milner, 1957). I do not argue that the hippocampus is not fundamental to episodic memory but rather present an existence proof that both types of information, episodic and semantic, can be stored in the same set of weights given an appropriate overall architecture and algorithm. The nature of the

[7] An item is defined herein simply as a set of co-active features chosen from a set of possible features. If two items, Υ_1 and Υ_2, are identical, then they share all higher-order statistics, meaning that all pairs of features contained in Υ_1 are also contained in Υ_2, all triples of features in Υ_1 are also in Υ_2, etc. As we consider a Υ_2 with progressively smaller overlap with Υ_1, they share progressively less higher-order statistics. This generalizes to whole sequences of items, wherein the number of possible higher-order features is exponentially higher because of the added temporal dimension.

relationship of hippocampus and neocortex is still an open question. Perhaps, TEMECOR's demonstration that episodic (0th order) and higher-order information can reside in a single set of binary weights will help clarify the relationship.

An important principle implicit in this model is that a physical hierarchy is not strictly necessary in order to represent a hierarchical category (or similarity) structure. Such structure can be represented directly in the patterns of overlaps over the set of codes defined over a single layer, e.g., TEMECOR's L2 (cf. Figure 1). Of course, an explicit hierarchy provides a further dimension for representing such structure. And, clearly the brain utilizes such an explicit hierarchy. I note however, that the former mechanism is simpler and must be phylogenetically older. Undoubtedly there was a 'single-layer' brain before there was a 'multi-layer' brain. In any case, the model proposed herein has already been generalized to a hierarchical framework with an arbitrary number of layers. This hierarchical version has demonstrated invariance to substantial non-uniform time warping (Rinkus & Lisman, 2005), while retaining its $O(1)$ time complexity, and will be explained in a subsequent paper.

Acknowledgment

This work has been supported by an individual grant from The Redwood Neuroscience Institute. Many thanks to my advisor, John Lisman, for numerous helpful comments and challenges that have greatly improved this paper.

References

Abeles, M. (1991) Corticonics: Neural Circuits of the Cerebral Cortex, Cambridge University Press, Cambridge, 1991.

Blitzer, J., Globerson, A. & Pereira, F. (2005) Distributed Latent Variable Models of Lexical Co-occurrences. Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics. (Eds.) Cowell, R. & Ghahramani, Z. Society for Artificial Intelligence and Statistics.

Brown, A. D. & Hinton, G. E. (2001) Training Many Small Hidden Markov Models. Proc. of the Workshop on Innovation in Speech Processing

Cleeremans, A. (1993). Mechanisms of Implicit Learning: Connectionist Models of Sequence Processing. A Bradford Book, The MIT Press, Cambridge, MA

Dayan, P. & Zemel, R. S. (1995) Competition and Multiple Cause Models. *Neural Computation* **7**(3).

Dempster, A. P. ,Laird, N. M., & Rubin, D. B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society, Series B*, **34**:1-38.

Elman, J. L. (1990) Finding structure in time. *Cognitive Science*, ***14***(2), 179–212.

Fodor, J. A. & Pylyshyn, Z. (1988) Connectionism and cognitive Architecture: A critical analysis. *Cognition*, **28:** 3-71

French, R. M. (1991) Using semi-distributed representations to overcome catastrophic interference in connectionist networks. *Proc. of the thirteenth annual conference of the cognitive science society*, 173-178. LEA, NJ.

Ghahramani, Z. & Jordan, M. I. (1997) Factorial Hidden Markov Models. *Machine Learning* **29**(2/3) 245-273.

Henderson, J. M., & Hollingworth, A. (1999) High-Level Scene Perception. *Annual Review of Psychology* **50**, 243-271.

Hinton, G. E. & Ghahramani, Z. (1997) Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London, B*, **352** 1177-1190.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart, J. L. McClelland, & the PDP research group (Eds.), Parallel distributed processing: Exploration in the microstructure of cognition 1: Foundations (pp. 77-109). Cambridge, MA: MIT Press.

Hinton, G. E. (1999) Products of experts. *Proc. of the Ninth International Conference on Artificial Neural Networks* (ICANN 99). V.1 1-6.

Jacobs, R. A., Jiang, W. & Tanner, M. A. (2002) Factorial Hidden Markov Models and the Generalized Backfitting Algorithm. *Neural Computation* **14**, 2415-2437.

Jordan, M. I. (1986) Serial order. Tech. rep. 8604, Institute for Cognitive Science, University of California, San Diego.

Kanerva, P. (1988). Sparse distributed memory. Cambridge, MA: MIT Press

Lee, T. S. & Mumford, D. (2002) Hierarchical bayesian inference in the visual cortex. J. *of the Optical Society of America, A*, **20**(7) 1434-1448.

Lynch, G. (1986) *Synapses, Circuits, and the Beginnings of Memory*. The MIT Press, Cambridge, MA.

McClelland, J. & Rumelhart, D. (1985) Distributed memory and the representation of general and specific information. *Journal of Experimental Psychology: General*, **114**(2), 159-188.

McCloskey, M. & Cohen, N. J. (1989) Catastrophic interference in connectionist networks: The sequential learning problem. In Bower, G. H. (Ed.), *The Psychology of Learning and Motivation*, Vol. **24**, 109–165. Academic Press.

Moll, M. & Miikkulainen, R. (1995) Convergence-Zone Episodic Memory: Analysis and Simulations. Tech. Report AI95-227. The University of Texas at Austin, Dept. of Computer Sciences.

Mountcastle, V. B. (1957) Modality and topographic properties of single neurons of cat's somatic sensory cortex. *J. Neurophysiology* **20**, 408–434.

Olshausen, B. A. & Field, D. J. (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* **381**, 607-609.

O'Reilly, R. C. (1998) Six principles for biologically based computational models of cortical cognition. *Trends in Cognitive Sciences*, **2**(11). 455-462.

O'Reilly, R.C., Busby, R.S. & Soto, R.(2003) Three Forms of Binding and their Neural Substrates: Alternatives to Temporal Synchrony. A. Cleeremans (Ed) *The Unity of Consciousness: Binding, Integration, and Dissociation,* 168-192, Oxford: Oxford University Press.

O'Reilly, R. C. & McClelland, J. L. (1994) Hippocampal conjunctive encoding, storage and recall: Avoiding a tradeoff. Tech. rep. PDP.CNS.94.4, Parallel Distributed Processing and Cognitive Neuroscience, Carnegie-Mellon University, Pittsburgh, PA.

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

Pearl, J. (1988) *Probabilistic Reasoning in Intelligent Systems*. Morgan-Kaufmann. San Mateo, CA.

Peters, A. & Sethares, C. (1996) Myelinated axons and the pyramidal cell modules in monkey primary visual cortex. *J. Comp. Neurol.* **365** 232-255.

Rachkovskij, D. A. & Kussel, E. M. (2000) Building large-scale hierarchical models of the world with binary sparse distributed representations.

Rachkovskij, D. A. & Kussel, E. M. (2001) Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning. *Neural Computation* **13**(2). 411-452.

Rinkus, G. J. (1995) TEMECOR: An Associative, Spatiotemporal Pattern Memory for Complex State Sequences. *Proceedings of the 1995 World Congress on Neural Networks*. LEA and INNS Press. 442-448.

Rinkus, G. J. (1996) A Combinatorial Neural Network Exhibiting both Episodic Memory and Generalization for Spatio-Temporal Patterns. Ph.D. Thesis, Graduate School of Arts and Sciences, Boston University.

Rinkus G. J. & Lisman J. (2005) Time-Invariant Recognition of Spatiotemporal Patterns in a Hierarchical Cortical Model with a Caudal-Rostral Persistence Gradient. *Society for Neuroscience Annual Meeting, 2005*. Washington, DC. Nov 12-16.

Roweis, S. & Ghahramani, Z. (1999) A Unifying Review of Linear Gaussian Models. *Neural Computation* **11** 305-345.

Scoville, W. B. & Milner, B. (1957) Loss of recent memory after bilateral hippocampal lesions. J. Neurology, Neurosurgery and Psychiatry **20** 11-21.

Uusitalo, M.A., Williamson, S.J. & Sepp, M.T. (1996) "Dynamical organization of the human visual system revealed by lifetimes of activation traces" Neuroscience Letters **213** 149-152.

Van Gelder, T. (1990) Compositionality: a connectionist variation on a classical theme. *Cognitive Science **14***, 355-384.

Waibel, A. (1989) Consonant recognition by modular construction of large phonemic time-delay neural networks. In Touretzky, D. S. (Ed.), *Advances in*

Neural Information Processing Systems ***I***, 215–223 San Mateo, CA. Morgan Kaufmann.

Williams, C. K. I. & Hinton, G. E. (1991) Mean field networks that learn to discriminate temporally distorted strings. In Touretzky, D., Elman, J., Sejnowski, T. & Hinton, G. (Eds.) *Connectionist models: Proceedings of the 1990 summer school.* San Francisco, CA. Morgan Kaufmann.

Williams, R. J. & Zipser, D. (1989) A learning algorithm for continually running fully recurrent neural networks. *Neural Computation* ***1***, 270-280.

Willshaw, D., Buneman, O., & Longuet-Higgins, H. (1969) Non-holographic associative memory. *Nature* **222**, 960-962.

Appendix: Remarks on Storage Capacity

Definitive storage capacity (i.e., space complexity) results for the version of TEMECOR described in this paper, referred to as T2 in this appendix, are not available yet. In lieu of this, I offer simulation results for a simpler version of the model (Rinkus, 1995, 1996), referred to as T1, which indirectly suggests that T2 will scale well to large problem sizes. T1 also has two layers and its L2 is also partitioned into Q WTA CMs. However, there are two important differences between T1 and T2:

a) In T1, only a small subset of the CMs were *active* on any given time step, i.e., 10 out of 100. Only active CMs could have an active L2 unit. In T2, all CMs are active on every time step. Because the *coding rate* (i.e., the fraction of units active at any given moment) is so much lower in T1, the rate of saturation of both L2's H-matrix and of the vertical weight matrices between L1 and L2 are also much lower. This sparseness is a major factor underlying T1's high storage capacity, evidenced in Figure A1.

b) T1 does not compute the degree of match between expected and actual inputs and thus does not titrate the amount of noise added into the winner selection process. Rather, on all time steps during learning, the winning unit in each active CM is chosen completely at random. Thus, the degree of overlap between two traces in T1 is not an increasing function of the spatiotemporal similarity of the input moments that they represent. Consequently, T1 does not explain similarity-based recognition/generalization.

The simulation results of Figure A1 show that T1's capacity increases approximately linearly in the number of synapses (i.e., network size). A large amount of information is stored in networks in these experiments. For the largest network size (16,240,000 binary synapses), the model was able to store 3084 10-item sequences, each item consisting of 20 out of 100 binary features, chosen at random. This set of sequences contains a total of 2,190,870 bits of information, yielding approximately 0.135 bits/synapse.[8] While this is lower than the theoretical maximum bits/synapse (~0.69

[8] In these T1 capacity experiments, a recall accuracy criterion of 97% was used. That is, each data point represents the maximum number of sequences that could be stored such that all sequences could be recalled with at least 97% accuracy. Due to the presence of some errors in the recalled sequences (i.e., an occasional deleted or intruding L2 unit), the actual bits/synapse value is slightly less than 0.135.

bits/synapse) for associative memory binary units, the more important question is whether or not the bits/synapse measure at least remains constant as the network grows and these empirical results suggest it does.

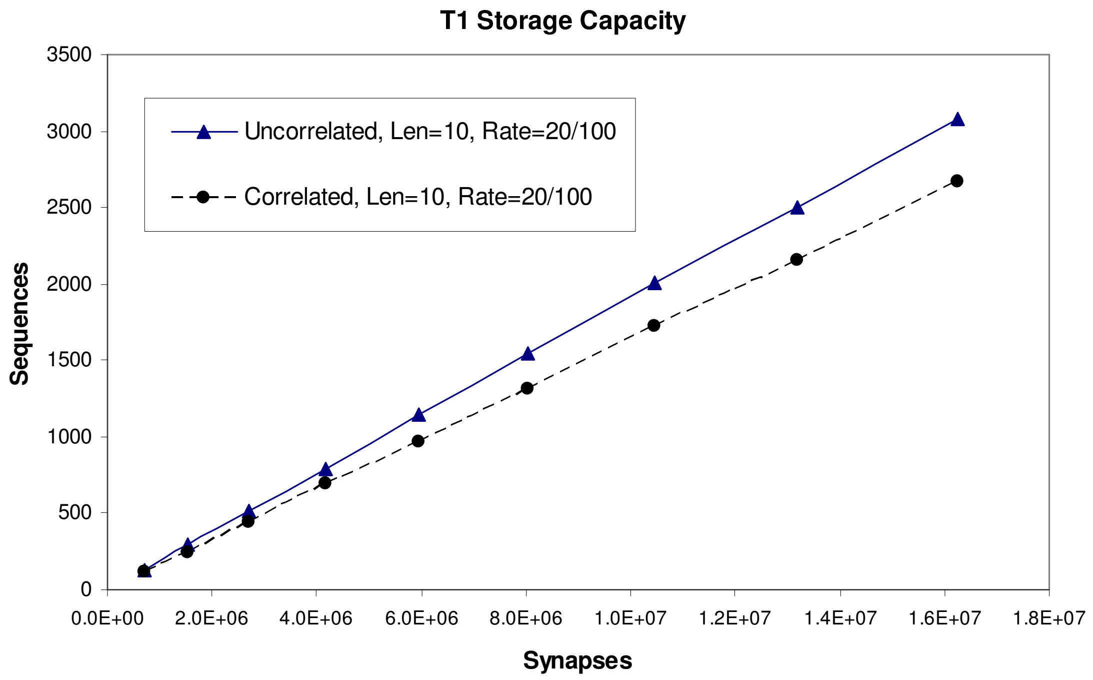


Figure A1: The storage capacity of T1 increases approximately linearly in the size of (i.e., the total number of synapses in) the network. This is true for uncorrelated and correlated sequences. Uncorrelated sequences had 10 time slices (items) and all items consisted of 20 features chosen at random from the 100 binary input features. For correlated sequences, we first generated an alphabet of 100 items; again, each item had 20 out of 100 randomly selected features. Then, each correlated sequence was generated by selecting each of its 10 items randomly with replacement from the alphabet. Thus, the correlated sequence sets are, more specifically, *complex* sequence sets, in that individual items occur numerous times and in numerous different contexts throughout the set.

As noted above, the sparseness of L2 codes in T1 is the major factor yielding T1's high capacity. This raises the question: can T2 be modified so that only a small fraction of its CMs are active at any moment? If so, then T2 is likely to have a capacity approaching that of T1. This is an active area of investigation.