Sequential Learning and Retrieval in a Sparse Distributed Memory: The K-winner Modern Hopfield Network

Shaunak Bhandarkar[*]
shaunakb@stanford.edu

James McClelland[*]
jlmcc@stanford.edu

Abstract

Many autoassociative memory models rely on a localist framework, using a neuron or slot for each memory. However, neuroscience research suggests that memories depend on sparse, distributed representations over neurons with sparse connectivity. Accordingly, we extend a canonical localist memory model—the modern Hopfield network (MHN)—to a distributed variant called the K-winner modern Hopfield network, equating the number of synaptic parameters (weights) in the localist and K-winner variants. We study both models' abilities to reconstruct once-presented patterns organized into long presentation sequences, updating the parameters of the best-matching memory neuron (or k best neurons) as each new pattern is presented. We find that K-winner MHN's exhibit superior retention of older memories.

1 Introduction

It is common to model memory using an individual neuron [1] or slot [2] for each memory. The modern Hopfield network (MHN) [3] exemplifies this, storing memories in the connections into and out of individual neurons and accessing them with computations equivalent to query-based retrieval of patterns stored in slots in modern artificial intelligence systems [4]. There is, however, an alternative tradition [5, 6, 7, 8], in which memories are distributed across the connections of a sparse ensemble of neurons, each connected only to a subset of the neurons representing the item to be stored, and each participating in many different memories. Inspired by the usefulness of slot-based systems in AI, we wondered whether certain aspects of their behavior can be captured as emergent properties of sparse, distributed memory systems. If so, this would help build bridges between computational abstractions and their possible biological implementations; and if sparse, distributed memories have advantages, they might be taken up to enhance AI systems.

Another tradition treats memory as unbounded, at least in principle [9, 10], but brains have limited capacity, so we studied learning in fixed-capacity systems. This requires a policy to allow rapid storage of new memories while minimizing interference with items previously learned. Here, we explore a simple policy from early biologically-inspired neural models [1, 11, 12], whereby each input updates the connections of neurons most closely aligned with itself. We apply this policy in a setting where the memory system encounters each pattern in a long sequence of independent (or structured) patterns exactly once, examining fidelity of retrieval of patterns of varying 'ages'.

2 Model Design

Extending the modern Hopfield network (MHN), we present the K-winner MHN, an autoassoicative memory with visible layer size n_v and hidden layer size n_h. We see the visible layer as a proxy for

[*]Department of Psychology, Stanford University, 450 Jane Stanford Way, Stanford, CA, 94305, USA

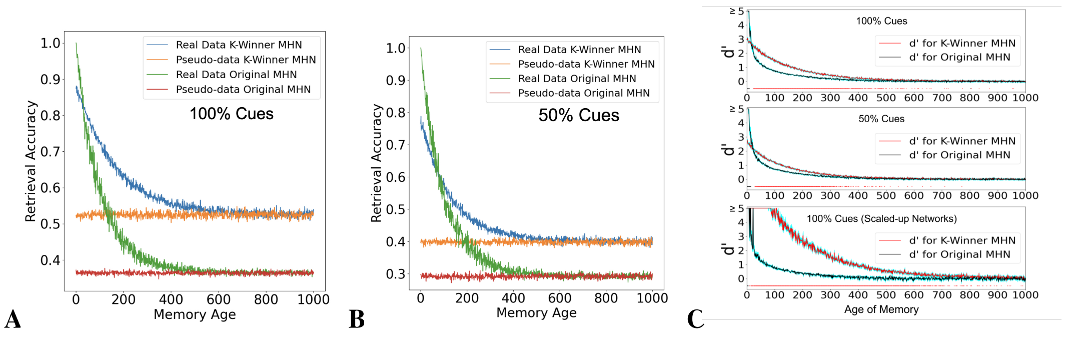


Figure 1: **A** Comparison of the small-scale K-winner MHN's and original MHN's retrieval ability (using 100% cues) for memories of different ages, relative to their an untrained (pseudo-memory) baselines. **B** Same as **A** but using 50% cues from real and pseudo-memories. Results were averaged over 200 independent runs of each model. **C** Retrieval sensitivity d' for the small-scale K-winner and original MHN using 100% cues (top) and 50% cues (middle); and d' with full cues for the large-scale networks described in the text (bottom). Cyan: d' standard error. Horizontal segments show ages where K-winner MHN d' is higher (red) and MHN d' is higher (black), with uncorrected $p < 0.0159$.

the input to the hippocampus (i.e. the entorhinal cortex) while the hidden layer is a simplified proxy for the hippocampus. The visible layer receives binary patterns with fixed sparsity s_v (the fraction of the n_v visible neurons set to 1). The hidden layer forms a binary representation of the input with sparsity s_h, less than s_v. We assess retrieval of previously seen patterns from complete or partial input cues. The original MHN is a special case of this approach, in which $s_h = 1/n_h$, so that a single hidden unit is chosen to represent each memory.

As in the MHN, the stored memories reside in two sets of weights: a matrix $M \in \mathbb{R}^{n_h \times n_v}$ from the visible layer to the hidden layer, and a return matrix $M' \in \mathbb{R}^{n_v \times n_h}$. Before training, each entry in each matrix is initialized uniformly to a number in $(0, 1)$. Additionally, we incorporate the concept of synaptic sparsity via a binary "fan-in" matrix $F \in \mathbb{R}^{n_h \times n_v}$, in which each row of F is randomly initialized to have a $f \cdot n_v$ of 1's, where f is the fraction of visible units with connections to each hidden unit. We thus obtain the effective weight matrices $W = M \odot F$ and $W' = M' \odot F^T$ (where $\odot$ denotes elementwise multiplication), enforcing symmetry as in the MHN. Then, for input pattern x, the retrieved output is given by

$$x^{out} = \sigma_v(W'\sigma_h(Wx)).$$

For simplicity, the functions σ_l ($l \in \{v, h\}$) are hard k-winner-take-all functions, such that the $k_l := s_l \cdot n_l$ units of layer l with the highest activations are set to 1 and the rest to 0. Using $z := \sigma_h(Wx)$ for the hidden representation, we generalize the modern Hopfield network via the biologically inspired weight update rule [1, 11]:

$$W_{ij} \leftarrow W_{ij} + \epsilon(x_j - W_{ij})z_i F_{ij}$$

where $\epsilon \in (0, 1)$ is the learning rate. We apply the same update to the jith entry in the return matrix W' [12], imposing symmetry as in the MHN. Note that when $s_h = 1/n_h$ (so that $k_h = 1$), $f = 1$, and $\epsilon = 1$, the model becomes a binary version of the original MHN, storing each new memory that enters the system in the incoming and outgoing connections of one of its n_h neurons, completely replacing one old memory. When we compare this localist model to the sparse distributed case ($k_h > 1, f < 1$), we equate the total number of weights (parameters) by setting n_h in the K-winner case to $1/f$ times the n_h of the localist case. We conducted exploratory modeling of the parameter space of small K-winner models, reporting below on a parameter setting that brings out interesting comparisons with the original (1-winner) MHN.

3 Experiments

3.1 Sequential Learning with Random Sparse Patterns

We present a small-scale K-winner MHN with $\epsilon = 0.3, f = 0.5, n_h = 200, s_h = 0.025$ ($k_h = 5$), and compare it to a small-scale MHN with $\epsilon = f = 1, n_h = 100, s_h = 0.01$ ($k_h = 1$), testing their

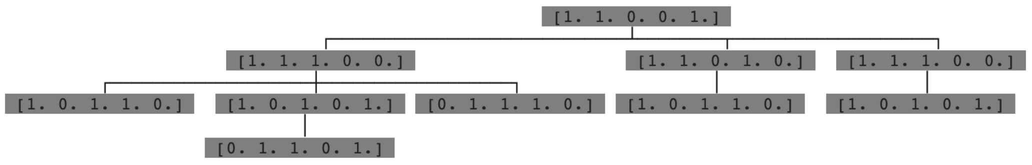


Figure 2: Sample tree generated by TGCRP consisting of 10 patterns with $n_v = 5$, $s_v = 0.6$, $b = 1$.

ability to retrieve previously learned patterns after a *single* pass through a 4000-pattern sequence of independent random binary patterns of dimension $n_v = 100$ and sparsity $s_v = 0.1$. Preliminary analyses showed that no trace of the first 3000 patterns remained in either model. We then examined each model's performance in retrieving the most recent 1000 patterns, with weights frozen, on the following measures.

Full pattern retrieval. For the pattern x_a of age a (the ath most recently learned pattern), we assess the proportion $\rho(x_a)$ of the units of x_a correctly retrieved at test time, when the full pattern x_a is passed as input (Fig. 1A). For each age a, we also consider an untrained "pseudo-memory" $\tilde{x}_a$ (with $s_v = 0.1$) and compute $\rho(\tilde{x}_a)$. The MHN perfectly retrieves the most recent pattern, but its performance rapidly degrades. The K-winner MHN's performance is slightly worse initially[2] but degrades more slowly; Fig. 1C (top) shows a d' measure[3] indicating how well $\rho(x_a)$ represents retrieval above baseline (see Appendix Section 6.1 for a discussion of the difference in baseline retrieval accuracies between the two models and an alternate metric of memory performance). The K-winner MHN's advantage is reliable from age 23.

Pattern completion. We also assessed the proportion of units correctly retrieved from 50% partial cues (half of the 1's in x_a are randomly chosen and set to 0) from real memories and pseudo-memories (Fig. 1B). Again, the K-winner MHN's performance is slightly worse than that of the original MHN at first, but degrades more slowly, as confirmed by d' in Fig. 1C (middle). The K-winner MHN's advantage is reliable from age 29.

Much sparser fan-in. The entorhinal input to the hippocampus contains far more than $n_v = 100$ units, and each hippocampal neuron connects with a far smaller fraction of entorhinal neurons than $f = 0.5$. In light of this, we scaled-up the visible layer, setting $n_v = 1000$, without changing input pattern sparsity $s_v = 0.1$. For the large-scale K-winner MHN, we set $n_h = 2000$ and $f = 0.05$ (with $\epsilon = 0.3$ and $s_h = 0.025$ as before, so that $k_h = 50$), and compared it to an MHN with $n_h = 100$ and $f = \epsilon = k_h = 1$. Fig. 1C (bottom) shows d' for full pattern probes, averaged over 10 independent d' samples with each computed using 20 independent runs. Here, the K-winner MHN shines. Both have $d' > 5$ at first. While the large-scale MHN is ahead initially, the large-scale K-winner model's d' advantage becomes reliable at age 9 and persists well past age 500.

3.2 Memory Performance with Structured Patterns

It is natural to wonder how our sparse distributed model fares against the 1-winner MHN when memories have nontrivial similarity structure. One can generate such data by means of a hierarchical generative process, as items reflective of natural experience often possess hierarchical structure [13, 14]. We devise a method called the Tree-Generating Chinese Restaurant Process (TGCRP) described in Appendix Section 6.2. The TGCRP algorithm generates a tree of binary patterns of size n_v and sparsity level s_v, in which each child node's pattern is the same as its parent node's pattern, except that b of its 1's are switched to 0's and b of its 0's are switched to 1's (at random), where b specifies the number of bit flips going from a parent to a child; the leaf nodes of the tree are used as the patterns with which to train our memory models (see Fig. 2).

[2]This occurs because the weight update is not large enough to ensure that the winning units' outgoing weights can reproduce the stored item perfectly.

[3]We averaged over 50 independent sample estimates of d' using 20 independent runs per sample. Within each sample, for the memory $x_{a,i}$ and pseudo-memory $\tilde{x}_{a,i}$ of age a in run i, we use $\rho(x_{a,i})$ and $\rho(\tilde{x}_{a,i})$ to compute $\delta_{a,i} = \rho(x_{a,i}) - \rho(\tilde{x}_{a,i})$. We then compute $d' = \mu_{\delta_a}/\sigma_{\delta_a}$ where $\mu_{\delta_a} = \frac{1}{20}\sum_{i=1}^{20}\delta_{a,i}$ and $\sigma_{\delta_a} = \sqrt{\sum_{i=1}^{20}(\delta_{a,i} - \mu_{\delta_a})^2/20}$.

We generated large pattern trees using the TGCRP with $n_v = 1000$, $s_v = 0.1$, and varying b, training the model on a total of 4000 randomly-ordered memory patterns sampled from the leaf nodes of the tree. As before, we tested the 1000 newest memories (of these 4000 patterns) for retrieval testing, with weights frozen; we also sampled 1000 untrained "pseudo-memory" patterns from the leaf nodes of the same tree. To compare the memory sensitivities (past baseline) of the K-winner and original MHN, we ran a d' experiment with our large-scale networks for varied b, testing retrieval for full (100%) cues. With $b = 10$ the performance of the two models is very similar; for $b = 5$, the original MHN performs better, but for $b = 15$ (out of 100 1-bits), the K-winner advantage for all but the most recent memories readily emerges (Fig. 3).

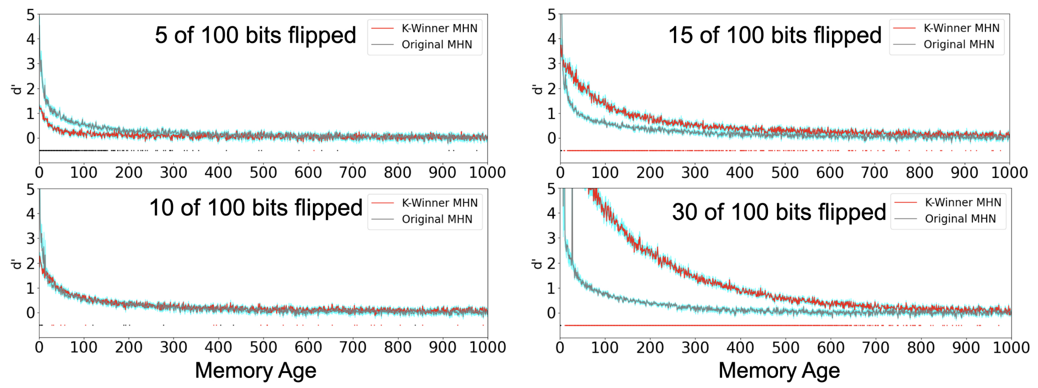


Figure 3: d' measure for our large-scale K-winner MHN and original MHN using 100% cues but varying the number of bit flips. Cyan: d' standard error. Horizontal segments show ages where K-winner MHN d' is higher (red) and MHN d' is higher (black), with uncorrected $p < 0.01$.

4 Discussion

Across our simulations, we find that the localist MHN maximizes retrieval accuracy for small-age memories, but rapidly loses memory information; our K-winner model yields slightly lower accuracy for small-age memories but degrades more gracefully for older memories. Although not shown, the K-winner models exhibit higher cumulative d' over all past memories, indicating a higher effective storage capacity. We believe this occurs in part because the K-winner MHN adjusts the weights for several units using partial weight updates; while this mechanism slightly compromises on perfect retrieval of recent memories, it enables a given memory trace to persist – superimposed on the weights associated with other memories – for many iterations of sequential learning before being fully erased. In contrast, in the MHN, a given memory is erased once its corresponding hidden neuron is assigned a new memory (see Appendix Section 6.1.3 for a quantitative characterization of this phenomenon).

The K-winner MHN's retention advantage for older memories appears to persist even when memories have nontrivial similarity structure (rather than being uniformly sampled at random); at the same time, training on TGCRP data with very high similarity structure causes learning ability to decrease across both networks, and the K-winner advantage can be reversed in very high-similarity training data. We aim to further probe this phenomenon in the context of one-shot visual recognition memory [15], in which the (structured) dataset consists of suitably generated latent embeddings of naturalistic images.

We have begun to explore the effects of lowering the learning rate (ϵ) in the original MHN. This allows traces of several memories to persist in this model as well, also at a cost to initial retrieval accuracy. Future work will explore the performance trade-offs of variants of the MHN with datasets of varying structure as the parameters k_h, f, and ϵ vary from their default values of 1 in the original MHN. Preliminary findings (see Appendix Section 6.1.4) suggest that a smaller learning rate increases d' for older memories in the MHN, but that a K-winner model may still retain an advantage.

In sum, further work is required to understand the full space of K-winner MHNs and their performance with varied data sets. The work presented here suggests that there may be data and parameter regimes in which K-winner MHNs (with $k_h > 1$) have clear advantages over MHNs. Future work will also explore refinements of both models that might further enhance their memory capabilities.

5 Acknowledgements

SB was supported by a summer internship award (2022) from the Symbolic Systems department at Stanford University. We thank members of the PDP lab at Stanford for discussion.

References

[1] Stephen Grossberg. Adaptive pattern classification and universal recoding: 1. parallel development and coding of neural feature detectors. *Biological cybernetics*, 23(3):121–134, 1976.

[2] Douglas L. Hintzman. "schema abstraction" in a multiple-trace memory model. *Psychological Review*, 93(4):411–428, 1986.

[3] Dmitry Krotov and John J. Hopfield. Large associative memory problem in neurobiology and machine learning. *ArXiv*, abs/2008.06996, 2021.

[4] Hubert Ramsauer, Bernhard Schafl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Milena Pavlovi'c, Geir Kjetil Sandve, Victor Greiff, David P. Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. *ArXiv*, abs/2008.02217, 2021.

[5] Pentti Kanerva. *Sparse distributed memory*. MIT press, 1988.

[6] D. Marr. Simple memory: a theory for archicortex. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 262(841):23–81, 1971.

[7] Bruce L McNaughton and Richard GM Morris. Hippocampal synaptic enhancement and information storage within a distributed memory system. *Trends in neurosciences*, 10(10):408–415, 1987.

[8] Randall C O'Reilly and James L McClelland. Hippocampal conjunctive encoding, storage, and recall: Avoiding a trade-off. *Hippocampus*, 4(6):661–682, 1994.

[9] A. Turing. Turing machine. *Proc London Math Soc*, pages 230–265, 1936.

[10] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, October 2016.

[11] Chr Von der Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14(2):85–100, 1973.

[12] Stephen Grossberg. Adaptive pattern classification and universal recoding: 2. feedback, expectation, olfaction, illusions. *Biological cybernetics*, 23(4):187–202, 1976.

[13] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019.

[14] J. McClelland and T. Rogers. The parallel distributed processing approach to semantic cognition. *Nat Rev Neurosci 4, (2003)*, 4:310–322, 2003.

[15] Lionel Standing. Learning 10000 pictures. *Quarterly Journal of Experimental Psychology*, 25(2):207–222, 1973.

[16] Laurens de Haan and Ana Ferreira. *Extreme Value Theory: An Introduction (Springer Series in Operations Research and Financial Engineering)*. Springer, 1st edition. edition, 2010.

6 Appendix

6.1 Baseline Retrieval Accuracy and a Raw Difference Metric for Memory Performance with Random Patterns

6.1.1 Baseline Accuracy

Here we briefly discuss the difference in baseline retrieval accuracy between the K-winner MHN and the original 1-winner MHN when trained on random untrained patterns (shown in Fig. 1A).

We first consider the original MHN's retrieval accuracy (where $n_v = n_h = 100$). Here, all of the real patterns and "pseudo-memory" patterns were sampled uniformly from the collection of length 100 binary patterns with 10 1-bits. When an untrained pattern is presented to an MHN with $n_h = 100$ other memories stored in it, it will replace the best-matching memory presently in the system, meaning that this pattern has is expected to have higher-than-average correlation with the new incoming pattern. This is why the retrieval accuracy baseline for the 1-winner MHN is greater than 0.1 – the average correlation between random patterns. More precisely, the accuracy baseline should roughly equal the maximum of $n_h = 100$ samples from the distribution of possible pattern overlaps; see Section 6.1.3 for more details.

We do not have a full analysis of the exact baseline retrieval accuracy of the K-winner MHN. However, we can offer an intuitive characterization of some factors contributing to its higher baseline accuracy (as compared to the original MHN). Recall that in the small-scale K-winner (where $n_v = 100$, $n_h = 200$, $k_h = 5$, $f = 0.5$, and $\epsilon = 0.3$), each of the hidden units 'sees' only a subset of the bits from any given input pattern. In this case, the distribution of best matches will be based on a smaller sample size ($fn_h = 50$ rather than $n_h = 100$ elements); correspondingly, the standard deviation of a sampled proportion is proportional to $\frac{1}{\sqrt{n}}$ and the maximum of such a distribution will be larger than the maximum of a distribution of samples each with a larger n. Additionally, it is possible that having $k_h > 1$ adds a measure of robustness to retrieval (under uncertainty), enabling the retrieved output to be averaged over k_h rows of the weight matrix W rather than a single row. Other factors likely influence the actual baseline retrieval accuracy as well.

6.1.2 Raw Difference Metric

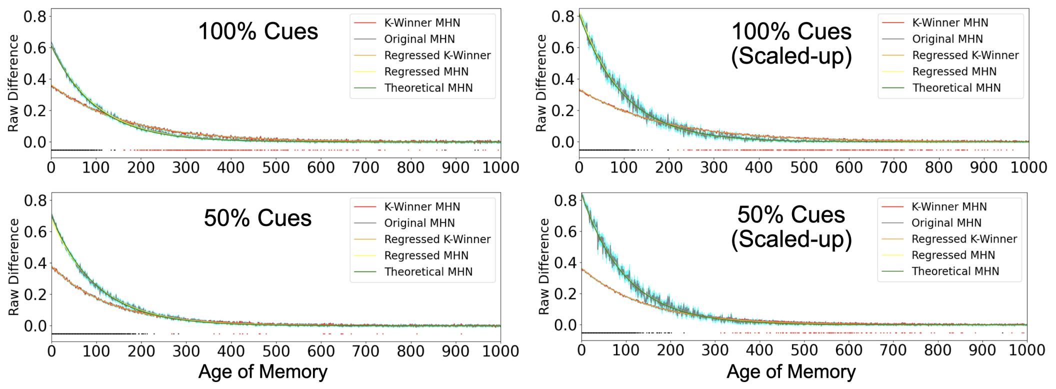


Figure 4: Raw difference (R.D.) across memory ages for the (small-scale and large-scale) K-winner and original MHNs, for 100% and 50% cues. Cyan: standard error. Horizontal segments show ages where the K-winner MHN has a higher R.D. (red) and original MHN has a higher R.D. (gray), with uncorrected $p < 0.01$. Exponential decay regression curves for K-winner (orange) and original MHN (yellow) nets (all r^2 values ≥ 0.96) as well as theoretical fit curve for original MHN shown (green).

In addition to the d' measure, we use a *raw difference* (R.D.) *metric*, averaged over the results 50 samples for our small-scale networks and 10 samples for our large-scale networks (see Fig. 4). Each R.D. sample itself is computed over 20 independent runs; defining $\delta_{a,i} := \rho(x_{a,i}) - \rho(\tilde{x}_{a,i})$ (as explained in Footnote 3), we compute the R.D. sample as $\mu_{\delta_a} = \frac{1}{20}\sum_{i=1}^{20} \delta_{a,i}$. Intuitively, this simplified metric quantifies a network's raw retrieval capability *past baseline*, owing to the learning process.

Next, we computed exponential regression curves for the R.D.'s of each network over memory age (Fig. 4). This was motivated by the fact that, for the original MHN, the retention for a given memory decays by a factor of $\left(1 - \frac{1}{n_h}\right)$ (proability of slot replacement) at each timestep; for a more detailed explanation, see Section 6.1.3 below. We perform exponential regression (R.D.$(a) \sim Ce^{-\beta(a-1)}$) over the first 200 memory ages; for example, for the large-scale networks with 50% cues (Fig. 4, bottom), we obtain $(C_{\text{k-win}}, \beta_{\text{k-win}}) = (0.366, 0.007)$ and $(C_{\text{MHN}}, \beta_{\text{MHN}}) = (0.847, 0.010)$, respectively. In Fig. 4, we also compute the theoretical decay curve for the original MHN (derived in Section 6.1.3 below): $(C_{\text{MHN-theory}}, \beta_{\text{MHN-theory}}) = \left(1 - s_v - \sqrt{\frac{2c}{n_v}(1-s_v)\log(n_h)}, -\log\left(1 - \frac{1}{n_h}\right)\right)$. Here, $c \in (0, 1]$ is the pattern cue level, expressed as a proportion. Each MHN theory curve matches its empirical counterpart, e.g., for the large-scale MHN with 50% cues, we obtain $(C_{\text{MHN-theory}}, \beta_{\text{MHN-theory}}) = (0.836, 0.010) \approx (C_{\text{MHN}}, \beta_{\text{MHN}})$. Comparing the empirical constants $C_{\text{k-win}}$ and C_{MHN} shows that the original MHN yields higher R.D.'s for small ages; comparing decay rates (β) shows that the K-winner MHN's retrieval performance degrades more slowly, supporting its superior retention of older memories.

6.1.3 Theoretical Analysis of MHN Retention Decay

The raw difference (R.D.) metric used in the previous section captures the retention decay in both our networks (past their respective baselines); understanding how an exponential decay curve specifically arises, as well as how the network hyperparameters (ϵ, f, k_h) contribute to the particular exponential decay curve obtained, warrants further theoretical analysis of the K-winner MHN. In this section, we present a detailed analysis of how the original MHN's (where $k_h = 1, \epsilon = 1, f = 1$) R.D. curve follows an exponential decay equation, given some light constraints on the cue level c and size of the network; in future work, we seek to theoretically characterize the decay equation for any K-winner MHN, using ideas from order statistics [16].

Theorem 1. *Suppose M is a slot-based (localist) MHN with input size n_v, input sparsity level s_v, and (sufficiently large) hidden size n_h, feedforward function f, and that M has been trained on N patterns where $N \to \infty$. Assume additionally that there exists a positive constant $c_0 << s_v \log\left(\frac{1}{s_v}\right)$ such that $n_h \le e^{c_0 n_v}$. Let x_a denote the pattern of age a (with $a \ge 1$, and letting $a = \infty$ mean that x_a is a pseudo-pattern), and moreoever let $x_{a,c}$ ($c \in (0, 1]$) be a partial cue for x_a, where a random proportion $1 - c$ of the 1 bits in x_a are made 0. Finally, let $\tilde{x} := f(x_{a,c})$. Then, we have that*

$$\theta := \frac{\log\left(1 - \left(1 - \frac{1}{n_h^{1+\epsilon}}\right)^{\frac{1}{n_h - 1}}\right)}{k_v \log(s_v)} << 1$$

for any $\epsilon \in (0, 1)$; moreover, if it holds that $c > \theta$, it follows that

$$\text{R.D.}(a) := \mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a - \frac{1}{k_v} f(x_{\infty,c})^T x_\infty\right]$$

$$= \left(1 - s_v - \sqrt{\frac{2c}{n_v}(1 - s_v)\log(n_h)}\right)\left(1 - \frac{1}{n_h}\right)^{a-1} + O\left(\frac{\log\log(n_h)}{\log(n_h)}\right).$$

Here, $k_v = s_v n_v$ and R.D.(a) denotes the expectation of the raw difference at age a, where the expectation is taken over all possible runs of the model M.

To prove this theorem, we need to first prove two propositions.

Proposition 2. *Suppose that, with the conditions of the above proposition, the slot that held the memory x_a in the MHN M has been replaced, or alternatively that $a = \infty$. Then, with this additional condition, we have*

$$\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] = s_v + \sqrt{\frac{2c}{n_v}(1 - s_v)\log(n_h)} + O\left(\frac{\log\log(n_h)}{\log(n_h)}\right).$$

Remark 3. *Observe that this proposition gives a formula for the baseline retrieval accuracy of the MHN M (for the cue level c), or equivalently, the expected retrieval accuracy for any given pseudo-pattern of cue level c.*

Proof. Suppose that $p_1, \ldots, p_{n_h}$ are the patterns stored in M. Then, for the input query $x_{a,c}$, and any random stored pattern p_i, we have

$$x_{a,c}^T p_i = \sum_{j:(x_{a,c})_j = 1} (p_i)_j$$

is a sum of ck_v random variables. Assuming these variables are approximately i.i.d. Bernoulli random variables with $p = \frac{k_v}{n_v} = s_v$, the Central Limit Theorem implies that $x_{a,c}^T p_i \sim \mathcal{N}\left(\frac{ck_v^2}{n_v}, \sqrt{ck_v s_v(1-s_v)}\right)$. At this point, the stored pattern p_i such that $x_{a,c}^T p_i$ will be retrieved. That is, retrieving p_i is equivalent to finding the largest value in a sample of n_h Gaussian random variables. To further this insight, we draw from the following theorem from Extreme Value Theory (see [16], Example 1.1.7):

Theorem 4. *Suppose that $s_1, s_2, \ldots, s_N$ are i.i.d. samples from the standard Gaussian distribution $\mathcal{N}(0,1)$. Let $S := \max_{1 \le j \le N} s_j$. Then, as $N \to \infty$, $\frac{S - a_N}{b_N}$ converges in distribution to the Gumbel distribution given by the pdf $p_G(z) = e^{-z - e^{-z}}$, where $a_N := \sqrt{2 \log n_h} - \frac{\log(4\pi \log(n_h))}{\sqrt{2\log(n_h)}}$ and $b_N := \frac{1}{\sqrt{2\log n_h}}$.*

With the terminology of the above theorem, we have that

$$\frac{\mathbb{E}[S] - a_N}{b_N} \to \mathbb{E}[G] = \gamma,$$

where G is a standard Gumbel variable (as specified by the theorem). It is well-known that the expectation of this variable is the Euler-Mascheroni constant, γ. Rearranging, we have that $\mathbb{E}[S] \to a_N + b_N\gamma = \sqrt{2\log(n_h)} + O\left(\frac{\log\log(n_h)}{\sqrt{\log(n_h)}}\right)$ as $N \to \infty$. Thus, for the random variables $x_{a,c}^T p_i$ ($1 \le i \le n_h$) given by $\mathcal{N}\left(\frac{ck_v^2}{n_v}, \sqrt{ck_v s_v(1-s_v)}\right)$, we have

$$\mathbb{E}\left[\arg\max_{1 \le i \le n_h} x_{a,c}^T p_i\right] = \frac{ck_v^2}{n_v} + \sqrt{ck_v s_v(1-s_v)}\mathbb{E}[S]$$
$$\approx \frac{ck_v^2}{n_v} + \sqrt{2ck_v s_v(1-s_v)\log(n_h)} + O\left(\frac{\sqrt{k_v}\log\log(n_h)}{\sqrt{\log(n_h)}}\right).$$

Now, suppose that the pattern p_l (for some $1 \le l \le n_h$ maximizes $x_{a,c}^T p_i$. Then, the MHN will simply retrieve the pattern $\tilde{x} := p_l$. Notice that

$$\mathbb{E}\left[\tilde{x}^T x_a\right] = \mathbb{E}\left[p_l^T x_{a_c}\right] + \mathbb{E}\left[p_l^T(x_a - x_{a,c})\right].$$

The first summand is exactly the quantity we have just approximated, and the latter summand is simply the expected amount of correlation between a random pattern with k_v 1-bits and a random pattern with $(1-c)k_v$ 1-bits. This is just $(1-c)k_v \cdot \frac{k_v}{n_v} = \frac{(1-c)k_v^2}{n_v}$ (again, simplifying by assuming that each bit of p_l is a Bernoulli variable). Therefore, (for n_h sufficiently large), we have

$$\mathbb{E}[\tilde{x}^T x_a] = \frac{ck_v^2}{n_v} + \sqrt{2ck_v s_v(1-s_v)\log(n_h)} + O\left(\frac{\sqrt{k_v}\log\log(n_h)}{\sqrt{\log(n_h)}}\right) + \frac{(1-c)k_v^2}{n_v}$$
$$= \frac{k_v^2}{n_v} + \sqrt{2ck_v s_v(1-s_v)\log(n_h)} + O\left(\frac{\sqrt{k_v}\log\log(n_h)}{\sqrt{\log(n_h)}}\right),$$

and therefore

$$\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] = s_v + \sqrt{\frac{2c}{n_v}(1-s_v)\log(n_h)} + O\left(\frac{\log\log(n_h)}{\sqrt{k_v}\sqrt{\log(n_h)}}\right).$$

Finally, from the conditions of Theorem 1, we know that $n_h \le e^{c_0 n_v}$, meaning that $n_v \ge \frac{1}{c_0}\log(n_h)$. Then, $\sqrt{k_v} \ge \sqrt{s_v n_v} \ge \sqrt{\frac{s_v}{c_0}\log(n_h)}$. Therefore, we in fact have

$$\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] = s_v + \sqrt{\frac{2c}{n_v}(1-s_v)\log(n_h)} + O\left(\frac{\log\log(n_h)}{\log(n_h)}\right).$$

□

Before proving the main theorem, we prove one further important proposition:

Proposition 5. *Fix $\delta \in (0,1)$. Suppose that when $x_{a,c}$ is queried to the MHN M, x_a is currently stored in M. Then, provided that*

$$c > \frac{\log(1-(1-\delta)^{\frac{1}{n_h-1}})}{k_v \log(s_v)},$$

it follows that

$$\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] > 1-\delta.$$

Remark 6. *In practice, the lower bound constraint on c can be further refined by more careful analysis. However, we should note that for our purposes, this bound will suffice. Indeed, for the large-scale MHN described in the main paper, we have $n_h = 100, k_v = 100, s_v = 0.1$; if we took $\delta = 10^{-3}$, then applying the inequality yields the constraint $c > 0.05$. Therefore, for the experiments detailed in our paper (which use $c = 0.5$ (50% cues) and $c = 1$ (100% cues)), we may effectively assume that $\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] = 1$ given that x_a is still stored by M.*

Proof. First, observe that if x_a is still stored by M, then the only way for $\tilde{x}$ to not be x_a is if there exists some other pattern p in M such that $x_{a,c}$ is a sub-pattern of p and the network retrieves this other pattern. That is,

$$\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] = \mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a \,\middle|\, \exists \text{ multiple super-patterns}\right] P(\exists \text{ multiple super-patterns}) + 1 \cdot P(\nexists \text{ other super-patterns}).$$

If we let $\eta := P(\exists \text{ multiple super-patterns})$, then $P(\nexists \text{ other super-patterns}) = 1-\eta$ and we have that $\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] \geq 1-\eta$. Now, we focus on bounding η above. Notice first that $P(\nexists \text{ other super-patterns})$ is the probability that each pattern p among the $n_h - 1$ patterns stored by M that are not x_a satisfies $p^T x_{a,c} < ck_v$; in this case, the hidden neuron corresponding to x_a will receive the highest activation and hence x_a will be retrieved. Moreover,

$$P(p^T x_{a,c} < ck_v) = 1 - P(p^T x_{a,c} = ck_v) = 1 - \frac{k_v}{n_v}\frac{k_v-1}{n_v-1}\cdots\frac{k_v-(ck_v-1)}{n_v-(ck_v-1)} \geq 1 - (\frac{k_v}{n_v})^{ck_v}.$$

Thus,

$$1-\eta = P(\nexists \text{ other super-patterns}) \geq \left(1 - s_v^{ck_v}\right)^{n_h-1},$$

so

$$\eta \leq 1 - \left(1 - s_v^{ck_v}\right)^{n_h-1}.$$

Now, given that

$$c > \frac{\log(1-(1-\delta)^{\frac{1}{n_h-1}})}{k_v \log(s_v)},$$

rearranging gives us

$$\begin{aligned} ck_v \log(s_v) &< \log\left(1-(1-\delta)^{\frac{1}{n_h-1}}\right) \\ \implies s_v^{ck_v} &< 1-(1-\delta)^{\frac{1}{n_h-1}} \\ \implies (1-\delta)^{\frac{1}{n_h-1}} &< 1 - s_v^{ck_v} \\ \implies \delta &> 1-(1-s_v^{ck_v})^{n_h-1}. \end{aligned}$$

Thus, given that c is bounded below as specified, we have that $\eta < \delta$ and thus

$$\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] \geq 1-\eta \geq 1-\delta.$$

□

Now, we are ready to prove our main result.

Proof of Theorem 1. Notice that, at test time, when $x_{a,c}$ is queried to the MHN M, the original pattern x_a is either still in the network or it has been replaced by another memory. Thus, we have

$$\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] = \mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a \,\middle|\, x_a \text{ in memory}\right] P(x_a \text{ in memory})$$
$$+\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a \,\middle|\, x_a \text{ not in memory}\right] P(x_a \text{ not in memory}).$$

By fixing suitable δ close to 0 (e.g., $\delta = 10^{-3}$, as in the case of our large-scale networks – see Remark 6), the first expectation term above is bounded below by $1-\delta$ and above by 1 (using Proposition 5). To obtain a precise error term (in the general case), suppose that $\delta = \frac{1}{n_h^{1+\epsilon}}$, for any $\epsilon \in (0,1)$. In this case, we have that $\frac{1}{n_h^2} \le \delta \le \frac{1}{n_h}$, so

$$\begin{aligned}
1-(1-\delta)^{\frac{1}{n_h-1}} &= 1-\left(1-\frac{1}{n_h-1}\delta+\frac{\frac{1}{n_h-1}\left(\frac{1}{n_h-1}-1\right)}{2!}\delta^2-\dots\right)\\
&\ge \frac{1}{n_h-1}\delta-\frac{\frac{1}{n_h-1}}{2!}\left(\delta^2+\delta^3+\dots\right)\\
&\ge \frac{1}{(n_h-1)n_h^2}-\frac{\frac{1}{n_h-1}}{2!}\left(\frac{1}{n_h(n_h-1)}\right)\\
&\ge \frac{1}{(n_h-1)n_h^2}-\frac{1}{2}\frac{1}{(n_h-1)^2 n_h}=\frac{n_h-2}{2n_h^2(n_h-1)^2}>\frac{1}{4n_h^3},
\end{aligned}$$

where we have assumed that $\frac{n_h-2}{n_h-1} > \frac{1}{2}$. Consequently, we have that

$$\begin{aligned}
\frac{\log(1-(1-\delta)^{\frac{1}{n_h-1}})}{k_v\log(s_v)} &\le \frac{\log(4n_h^3)}{-n_v s_v\log(s_v)} = \frac{1}{-s_v\log(s_v)}\frac{\log 4+3\log(n_h)}{n_v}\\
&\le \frac{1}{s_v\log\left(\frac{1}{s_v}\right)}\left(\frac{\log 4}{n_v}+3c_0\right)=o(1),
\end{aligned}$$

where we have used the assumption that n_v is sufficiently large and that $n_h \le e^{c_0 n_v}$ for some positive constant $c_0 << s_v\log\left(\frac{1}{s_v}\right)$. This indicates that for $\delta = \frac{1}{n_h^{1+\epsilon}}$, for any $\epsilon \in (0,1)$, the term

$$\theta := \frac{\log(1-(1-\delta)^{\frac{1}{n_h-1}})}{k_v\log(s_v)}$$

which constrains c from below is made to be sufficiently close to 0; moreover, for this choice of δ, we have

$$\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a \,\middle|\, x_a \text{ in memory}\right] = 1 + o\left(\frac{1}{n_h^{1+\epsilon}}\right)$$

for any $\epsilon \in (0,1)$.

Furthermore, by Proposition 2, both of the terms $\mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a \,\middle|\, x_a \text{ not in memory}\right]$ and $\mathbb{E}\left[\frac{1}{k_v}f(x_{\infty,c})^T x_\infty\right]$ are well-approximated by

$$s_v+\sqrt{\frac{2c}{n_v}(1-s_v)\log(n_h)}+O\left(\frac{\log\log(n_h)}{\log(n_h)}\right)$$

provided that n_h is sufficiently large. Finally, because M is a slot-based memory system, the probability that x_a is still in the memory is the probability that the hidden neuron that represents x_a has not since been replaced (for $a-1$ timesteps of learning). Thus,

$$P(x_a \text{ in memory}) = \left(1-\frac{1}{n_h}\right)^{a-1}$$

$$P(x_a \text{ not in memory}) = 1 - \left(1 - \frac{1}{n_h}\right)^{a-1}.$$

Putting all of our above facts together, we have that

$$\text{R.D.}(a) = \mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a - \frac{1}{k_v} f(x_{\infty,c})^T x_\infty\right] = \mathbb{E}\left[\frac{1}{k_v}\tilde{x}^T x_a\right] - \mathbb{E}\left[\frac{1}{k_v} f(x_{\infty,c})^T x_\infty\right]$$

$$= \left(1 + o\left(\frac{1}{n_h^{1+\epsilon}}\right)\right)\left(1 - \frac{1}{n_h}\right)^{a-1} + \left(s_v + \sqrt{\frac{2c}{n_v}(1 - s_v)\log(n_h)} + O\left(\frac{\log\log(n_h)}{\log(n_h)}\right)\right)\left(1 - \left(1 - \frac{1}{n_h}\right)^{a-1}\right)$$

$$- \left(s_v + \sqrt{\frac{2c}{n_v}(1 - s_v)\log(n_h)} + O\left(\frac{\log\log(n_h)}{\log(n_h)}\right)\right)$$

$$= \left(1 - s_v - \sqrt{\frac{2c}{n_v}(1 - s_v)\log(n_h)}\right)\left(1 - \frac{1}{n_h}\right)^{a-1} + O\left(\frac{\log\log(n_h)}{\log(n_h)}\right).$$

□

Theorem 1 indeed shows that the MHN raw difference decay curves presented in Fig. 4 are expected to closely follow exponential decay equations of the form $\text{R.D.}(a) = Ce^{\beta(a-1)}$, where

$$C = 1 - s_v - \sqrt{\frac{2c}{n_v}(1 - s_v)\log(n_h)}$$

$$\text{and } \beta = -\log\left(1 - \frac{1}{n_h}\right).$$

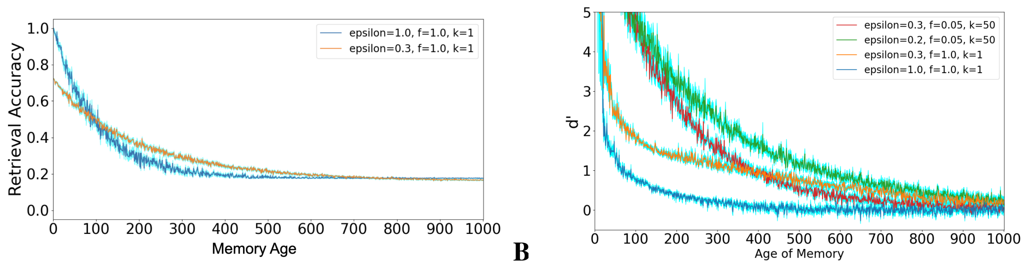
6.1.4 Performance of MHN with Graded Weight Updates

The theoretical analysis from the previous section fully characterizes retrieval accuracy (and raw difference) for a special case of the K-winner network where $\epsilon = 1, f = 1, k_h = 1$. Our endeavor to analyze retention decay in the case of general (ϵ, f, k_h) can be seen as an endeavor to understand the parameter space of all possible K-winner MHN's. As a first step in this direction, one might ask how the original one-winner MHN performs when the weight updates do not correspond to slot-based replacements but rather graded adjustments (i.e., $\epsilon < 1$). Does the distributed K-winner MHN (with $k_h > 1$) still retain its advantages over this modified instantiation of the MHN?

To empirically probe this question, we first ran a test of raw retrieval accuracy for our large-scale MHN ($\epsilon = 1, f = 1, k_h = 1, n_v = 1000, n_h = 100$) compared against a graded version of this MHN in which ϵ is reduced to 0.3. Each of their respective retrieval accuracies, shown in Fig. 5A, is averaged over 10 samples, where each sample itself was computed as an average over 20 independent runs of the respective model (similarly to Footnote 3). We also evaluated the d' sensitivities for these networks, in addition to those for two (large-scale) distributed K-winner MHN's ($f = 0.05, k_h = 50, n_v = 1000, n_h = 2000$) in which the learning rates were set to $\epsilon = 0.3$ and $\epsilon = 0.2$, respectively (Fig. 5B).

Somewhat like the K-winner MHN, the graded MHN ($\epsilon = 0.3$) appears to possess lower initial retrieval accuracy but more graceful retention decay compared to the slot-based MHN ($\epsilon = 1$); interestingly, the graded MHN also demonstrates robust d' sensitivity for older memories. In particular, it appears to maintain a higher d' than the corresponding large-scale K-winner MHN (with $\epsilon = 0.3$) from roughly age 400 onwards. However, when the distributed K-winner MHN's learning rate is reduced from $\epsilon = 0.3$ to $\epsilon = 0.2$, its resulting d' values appear to be higher than those of the graded MHN (Fig. 5B). This suggests that, for any given graded MHN with learning rate $\epsilon < 1$, we may be able to find a distributed K-winner MHN with learning rate $\epsilon' < 1$ that maintains a learning advantage over the graded MHN.

This preliminary exploration of the (ϵ, f, k_h)-parameter space additionally suggests that decreasing ϵ reduces initial retrieval accuracy while also slowing down retention decay for older memories. More broadly, it highlights the need to theoretically characterize the (ϵ, f, k_h) parameter space, so as to understand whether there exist classes of "optimal" K-winner MHN's that have high accuracy baselines *and* high learning ability past baseline, even for older memories.



A **B**

Figure 5: **A** Raw retrieval accuracies for large-scale MHN's with $\epsilon = 1$ and $\epsilon = 0.3$, respectively, after training on random patterns, using 100% cues during testing. **B** d' measure for various K-winner MHN's (with the same total number of weights). Reducing ϵ from 0.3 (red) to 0.2 (green) enables the distributed K-winner MHN to outperform the graded MHN with $\epsilon = 0.3$ (shown in orange).

6.2 More on Hierarchically Generated Patterns

6.2.1 Tree-Generating Chinese Restaurant Process Algorithm

In this section, we illustrate how the Tree-Generating Chinese Restaurant Process Algorithm (TGCRP) works. The 'Chinese Restaurant Process' refers to an iterative (discrete) stochastic process for clustering a group of objects, typically into clusters with uneven frequencies; such a process could, for example, be useful in modeling objects whose frequencies follow a power law. Because objects in natural experience tend to have hierarchical similarity structure (e.g., 'living things' might branch out into 'animals' and 'plants', which could further branch out into 'dogs', 'cats', 'trees', 'bushes', etc.), we devise a modified variant of the Chinese Restaurant Process that is capable of probabilistically generating a tree of patterns; lower patterns in the tree may be thought of as specific instantiations of their higher-up ancestors in the tree. Moreover, we only take the leaf nodes of such a tree and use these as the patterns for training and testing. Detailed in Algorithm 1 (below) is the TGCRP algorithm.

In considering the practical implementation of Algorithm 1, a few important points should be noted:

1. The slightly repetitive calculation of P_i and N_i over the course of generating the probabilistic tree appears to be onerous, but one can avoid such excessive computation by making them dynamically updating values that get updated after each iteration.
2. This algorithm is used to generate a tree that contains 'num_data' number of patterns. When running the TGCRP algorithm in practice, it should be noted that the number of leaf nodes of the tree is $\sim \frac{1}{2}$ of 'num_data'.
3. In each independent run of our retrieval d' (and raw difference) experiments with TGCRP-generated data, we generated one big tree with 14,000 total nodes, meaning that there were roughly 7,000 leaf nodes. We shuffled these leaf nodes, and sampled 3,000 of them for bringing our models' respective weight distributions to equilibrium (steady state), 1,000 of them to be used as patterns for learning and retrieval, and 1,000 of them to be used as pseudo-memories. (Note that the d' and raw difference measures themselves were calculated by averaging over many independent runs, each of which involved creating a new TGCRP tree.)

6.2.2 Similarity Structure of TGCRP-Generated Data

In practice, we observe that applying the TGCRP algorithm while varying b produces training data with varying similarity structure. This is reflected in the data similarity matrices for TGCRP-generated data (Fig. 6), where we have used a pattern size of $n_v = 1000$ and sparsity level of $s_v = 0.1$. Observe that decreasing b for a tree causes interspersed rectangular blocks of high similarity to appear, whereas increasing b sufficiently removes off-diagonal similarity structure (effectively bringing us back to the case of random i.i.d. patterns).

Algorithm 1 Hierarchical Data-Generating Algorithm (TGCRP). 'num_data' is the number of nodes in the entire tree. b is the number of bit flips done when going from a parent node to a child node. Note that the function bit_flipped(p, b) returns a slight 'corruption' of pattern p in which b random 1's are set to 0's and b random 0's are set to 1's. Additionally, let pattern(Q) denote the pattern stored within the tree node Q.

```
Require: num_data $\geq 1$, $1 \leq b \leq s_v n_v$
  Make a root node $R_0$ for the tree, and initialize it with a random binary pattern of length $n_v$ and sparsity level $s_v$.
  for $i$ in $1, \ldots,$ num_data - 1 do
      Initialize $X$ to be the tree's root node.
      Create node $R_i$ but do not yet add it to the tree.
      while node $R_i$ is not yet added to the tree do:
          Let $C_1, \ldots, C_m$ be the children nodes of $X$ $(m \geq 0)$.
          for $l$ in $1, \ldots, m+1$ do
              $N_r \leftarrow 1+ \#\{\text{descendants of } C_r\}$ for each $r = 1, \ldots, m$.
              if $l \leq m$ then
                  $P_l \leftarrow \frac{N_l}{1+\sum_{j=1}^{m} N_j}$
              else if $l = m+1$ then
                  $P_l \leftarrow \frac{1}{1+\sum_{j=1}^{m} N_j}$
              end if
          end for
          Sample $d \in \{1, \ldots, m+1\}$ using the categorical distribution Cat$([P_1, \ldots P_{m+1}])$.
          if $d = m+1$ then
              Make node $R_i$ a new child node of $X$.
              Make node $R_i$ store the pattern bit_flipped(pattern$(X), b)$.
          else
              $X \leftarrow C_d$
          end if
      end while
  end for
```

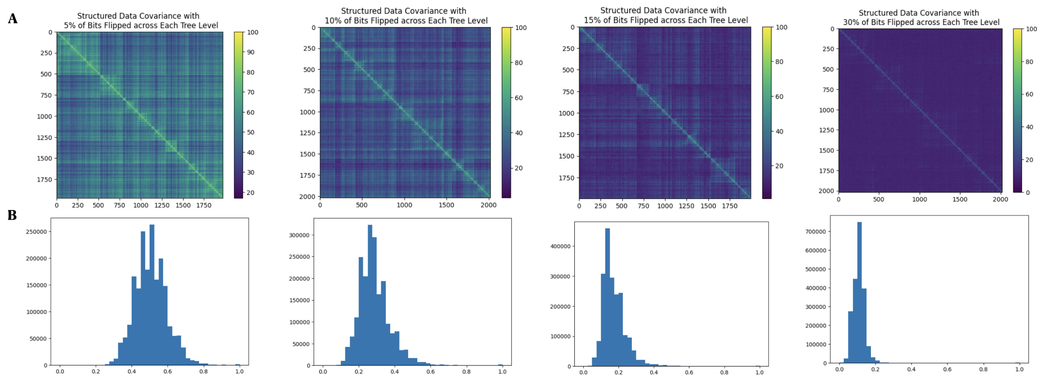


Figure 6: **A** Each matrix shown contains the pairwise dot product similarities for sample TGCRP-generated data. Matrices for TGCRP-generated data with $b = 5, 10, 15, 30$ (from left to right) are shown. **B** Histograms depicting the upper-triangular entries in each of the matrices in **A**; as b is increased (from left to right), the distribution of these entries tends to decrease to baseline. (For each similarity matrix-histogram pair, we generated a single tree with 4,000 nodes, using a pattern size of $n_v = 1000$ and $s_v = 0.1$.)