# ENCORE: A BIOLOGICALLY PLAUSIBLE ENCODING OPTIMIZATION FOR ONLINE ASSOCIATIVE MEMORY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Episodic memory requires rapid, one-shot encoding of experiences as they arrive, yet associative memory models such as Hopfield network and their modern variants (Modern Hopfield Network) are typically formulated and tested in the batch setting, where all patterns are stored simultaneously. We study the *online* associative memory problem, in which patterns arrive sequentially and must be encoded via one-shot exposure. Prior works have improved capacity along two axes: retrieval dynamics (e.g., Modern Hopfield Network) and learning rules (e.g., Storkey rule). Inspired by the hippocampus architecture and the neuronal memory allocation mechanism in the brain, we explore a third, orthogonal axis: optimizing the encoded patterns themselves. We propose an associative memory system in which the encoding dynamics are explicitly designed to minimized interference with previously stored encoded patterns. On online associative memory benchmarks, our system outperforms both Modern Hopfield networks and Hopfield network with Storkey learning rule by more than 5x in MNIST and 2x in ImageNet 128x128. These results suggest that pattern encoding, long overlooked relative to retrieval dynamics and learning rules, is a significant lever for designing associative memory in realistic online settings.

## 1 INTRODUCTION

A central challenge of artificial intelligence is how to build intelligent machines that can efficiently learn from experience and adapt to new environments. The last two decades have seen significant progress on one half of this problem. Great advances in deep learning (Krizhevsky et al., 2012; Silver et al., 2016; Brown et al., 2020), have produced systems that learn extraordinarily well on a fixed data distribution. However, the second half of the challenge, how to build machines that can learn in an online manner and adapt in a changing environment, still remains an open question.

The brain offers a potential solution to this problem via the Complementary Learning Systems (CLS) theory (McClelland et al., 1995). CLS theory proposes that the brain relies on two complementary systems: a fast learning hippocampal system that rapidly encode episodic experience (what, when, where), and a slow learning neocortex system that gradually extracts and learn an internal model of the world over time. While most current deep learning models are analogous to the role of the neocortex, CLS theory emphasizes the need to build an associative memory system that can quickly encode experience as they arrive, as a critical component for continual learning.

Associative memory models, including Hopfield networks and Modern Hopfields Network (Hopfield, 1982; Krotov & Hopfield, 2016; Ramsauer et al., 2021) are usually served as the model for the hippocampal memory storage process. These networks store episodic experiences (or memories) as attractors of a dynamical system, and given a cue (noisy or incomplete pattern), the memory can be retrieved via the network dynamic. However, these models have two major limitations. First, the models usually assume a batch setting, in which all patterns are presented to the model simultaneously, and the model's weight is computed over the entire dataset. This assumption bypass the central difficulty of online learning: the model only has one-shot exposure to the patterns, and as patterns arrive once at a time, naively updating the entire network can overwrite previous information, leading to catastrophic forgetting. Second, the models only focus on the retrieval dynamics and treat the memory patterns as either random or given. However, in the hippocampus, which neurons are recruited into a memory engram is not fixed in advanced. At encoding time, neurons have to



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

compete to be included in the engram (Han et al., 2007), and neurons that are more excitable (for example, because they were not used extensively in other recent memories), are more likely to be recruited. Therefore, the encoded memory representations depends not only on the current sensory inputs, but also on which neurons were used by recently formed memories, naturally preventing newly formed memory to interfere with existing ones (Josselyn & Tonegawa, 2020).

In this work, we propose a novel associative memory system with a distinct memory encoding component that, instead of storing patterns as given, optimizes the representation that store the patterns. When a pattern arrives, our method searches for an encoding representation that balance between three competing objectives (Eq. (5) and Table 1): staying faithful to the sensory input, spreading activity across underused neurons, and moving away from regions in the state-space that are already occupied by stored memories. This search is implemented via iterative k-winner-take-all procedure (Algorithm 1), which update the effective drive according to the gradient of the combined objective and re-select the $k$ active units until the code reaches a fixed point. In online associative memory tasks in which the inputs come in as an online stream and the model only has one-shot exposure to each pattern, our system with the encoding optimization component offers a significant gain in memory capacity, with 5x increase in MNIST and 2x increase in ImageNet 128x128 over both Modern Hopfield networks trained via backpropagation and Hopfield networks trained via Storkey learning rule. Our results indicate that memory encoding, an aspect that is largely overlooked compared to retrieval dynamics and learning rules, is in fact a significant component for associative memory in the online setting, and point to a principled and biologically grounded route to model online memory formation. Our primary contributions are:

1. **Memory Encoding as a New Axis to Improve Online Memory Capacity:** We introduce a memory encoding component that, instead of treating memory patterns as random or given, optimize each encoded pattern to minimize interference with those already stored. While previous works focus on retrieval dynamics and learning rules, we show that memory encoding is an orthogonal axis to improve online memory capacity.
2. **A Biologically Plausible Hippocampal Architecture:** Our model implements key computational operations in the hippocampal pathway (Fig. 1), using local Hebbian and delta-rule updates. Entorhinal cortex sensory inputs is mapped to a high-dimensional sparse representation through dentate gyrus–inspired expansion and inhibition, followed by CA3 pattern completion. Expanding inputs into a high-dimensional space yields many candidate codes, allowing encoding to select memories that reduce interference with existing ones.
3. **Large Capacity Gains in Online Associative Memory Tasks:** On online associative memory tasks, our system with encoding optimization improve memory capacity by more than 5x in MNIST and 2x in ImageNet 128x128 over basedline models (including Modern Hopfield networks and Hopfields network with Storkey learning rules).

## 2 ENCORE: ENCODING AS OPTIMIZATION FOR RECALL

### 2.1 PROBLEM SETUP

We consider a set of $N$ binary sparse patterns $\Xi = \{\xi^{(i)}\}_{i=1}^{N}$. Each pattern has dimension $D_{EC}$ (EC as in entorhinal cortex), activity (or sparsity) level $a_{EC}$, and number of active units $k_{EC} = \lfloor a_{EC} D_{EC} \rfloor$. The patterns are presented as a stream $(\xi^{(t)})_{t=1}^{N}$ one sample per time step.

The goal of the online associative memory model is to store previously seen patterns, and given a partial cue from a previously seen pattern, complete the cue to that pattern. Concretely, at time step $t$, given the partial cue $\tilde{\xi^{(i)}}$ from pattern $\xi^{(i)}$ $(i < t)$, the network needs to reconstruct $\xi^{(i)}$ from $\tilde{\xi^{(i)}}$.

### 2.2 OUR APPROACH

While most associative memory models performs pattern completion directly on the pattern space (Ramsauer et al., 2021) or as a dynamical recurrent between the visible and hidden units (Krotov & Hopfield, 2021; Kafraj et al., 2026), the hippocampus splits the memory encoding to two stages: (1) pattern separation in the dentate gyrus (DG), and (2) pattern completion via recurrent connection in the CA3 region. We propose an approach that captures this multi-stage processing (Fig. 1):

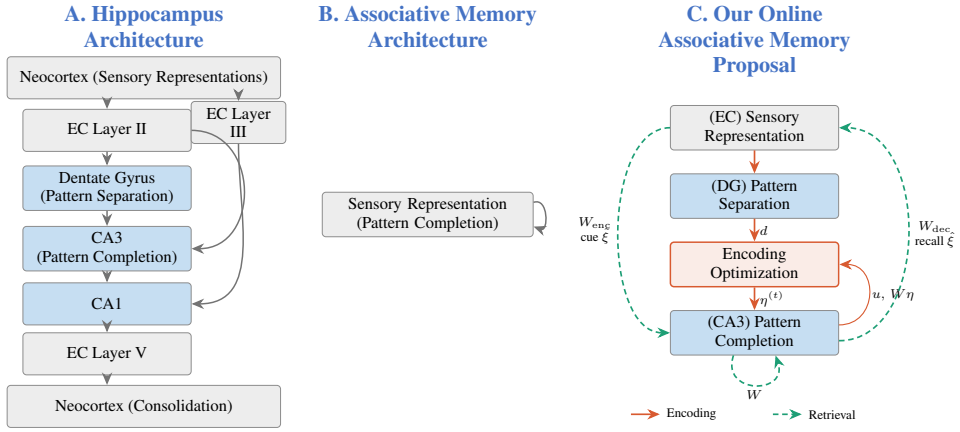



Figure 1: **Illustration of the memory encoding and retrieval pathway. (A)** Simplified hippocampal pathway for memory encoding and retrieval. **(B)** Standard associative memory architecture, which mainly focuses on retrieval dynamics. **(C)** ENCORE, with separate encoding and retrieval pathways. *Encoding* (solid): DG pattern-separates the input, and iterative $k$-WTA selects the CA3 code $\eta^{(t)}$ that balances fidelity ($d$), neuron usage ($u$), and interference ($W\eta$) (Eq. (5), Alg. 1); *Retrieval* (dashed): a cue $\tilde{\xi}$ enters CA3 via $W_{\text{enc}}$, is completed by attractor dynamics on $W$, and is read out as $\hat{\xi}$ via $W_{\text{dec}}$ with skip connection (see Appendix A.2.1).

**Step 1 (Dimensionality Expansion and Sparsification, Section 2.3).** Project the pattern $\xi$ to encoded pattern $\eta$ via random projection into a higher dimensional space $D_{CA3} = eD_{EC}$ ($e$ is the expansion factor) and sparsification $a_{CA3} = a_{EC}/e$ via k-Winner-Take-All operations (k-WTA) [1]. High dimensional projection and k-WTA operations have been applied as model of the cerebellum (Kawato et al., 2021), dentate gyrus (Rolls, 2016), and fruit fly's olfactory (Dasgupta et al., 2017) and have been shown both theoretically (Papadimitriou, 2019; Babadi & Sompolinsky, 2014) and empirically (Dasgupta et al., 2017) to have pattern separation property.

**Step 2 (Encoding Optimization, Section 2.4).** At time step $t$, the pattern separation in Step 1 is independent of both the history (previous patterns $\xi^{(t')}, t' < t$ ) and current pattern $\xi^{(t)}$. In the online memory setting, we would like to choose the encoded pattern $\eta$ to minimize the interference with previously stored encoded patterns $\eta^{(t')}$. To do this, we introduce an iterative encoding optimization procedure (Algorithm 1) to select the encoded patterns with lowest interference to existing patterns.

**Step 3 (Learn projection via local learning rules, Section 2.5).** After selecting the encoded pattern $\eta^{(t)}$, we need to learn the mapping between the pattern $\xi^{(t)}$ to $\eta^{(t)}$ (the encoding mapping), and $\eta^{(t)}$ to $\xi^{(t)}$ (the decoding mapping). We can parameterize this mapping via the weight matrix $W_{enc} \in \mathbb{R}^{D_{CA3} \times D_{EC}}$ and $W_{dec} \in \mathbb{R}^{D_{EC} \times D_{CA3}}$ and run local update via Hebbian learning or delta-rule.

### 2.3 Dimensionality Expansion

Our dimensionality expansion component mirrors the sparsity level and connectivity pattern of the dentate gyrus (DG). The encoding maps a pattern $\xi \in \{0,1\}^{D_{\text{EC}}}$ from entorhinal cortex (EC) space to an encoded pattern $\eta \in \{0,1\}^{D_{\text{CA3}}}$ in CA3 space via two successive random projections.

**Stage 1 (EC $\to$ DG):** A Gaussian random matrix $M^{\text{DG}} \in \mathbb{R}^{D_{\text{DG}} \times D_{\text{EC}}}, (M^{\text{DG}}_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(0, \frac{1}{D_{\text{EC}}}\right))$ projects $\xi$ into an intermediate DG space with expansion ratio $e_{\text{DG}} = D_{\text{DG}}/D_{\text{EC}} = 10$ (Andersen et al., 2006). A k-WTA operation then enforces sparsity $a_{\text{DG}} = a_{\text{EC}}/e_{\text{DG}}$, $\xi^{\text{DG}} = \text{kWTA}_{k_{\text{DG}}}\left(M^{\text{DG}}\,\xi\right), k_{\text{DG}} = \lfloor a_{\text{DG}} D_{\text{DG}} \rfloor$.

**Stage 2 (DG $\to$ CA3):** A sparse binary random matrix $M^{\text{CA3}} \in \{0,1\}^{D_{\text{CA3}} \times D_{\text{DG}}}, \sum_{i=1}^{D_{\text{CA3}}} M^{\text{CA3}}_{ij} = c, \forall j$. Each column contains $c = 15$ ones placed uniformly at random, models the sparse mossy-fiber projection from DG granule cells to CA3 pyramidal neurons via detonator synapses (Kumaran et al., 2016). Let the output of this step be $d = M^{CA3}\xi^{DG}$.

[1] A k-Winner-Take-All (k-WTA) operation keeps the $k$ largest entries of a vector (setting them to 1) and set the other entries to 0. It models the competitive inhibition that allows only the most activated neurons fire.

**Candidate set:** For a given pattern $\xi$, the *candidate set* $\mathcal{C}(\xi) \subseteq \{1, \ldots, D_{\text{CA3}}\}$ is the set of CA3 units that receive at least one active DG input: $\mathcal{C}(\xi) = \left\{ i \in \{1, \ldots, D_{\text{CA3}}\} : \left(M^{\text{CA3}}\, \xi^{\text{DG}}\right)_i \geq 1 \right\}$.

### 2.4 Encoding Optimization

Instead of obtaining the encoded pattern $\eta$ from a single k-WTA pass over the expansion output $d$, we cast encoding as an optimization problem: given the candidate set $\mathcal{C}(\xi)$, find the encoded pattern $\eta$ that (1) prefers underused neurons and (2) minimizes interference with stored patterns.

**Preference for Underused Neurons.** To make full use of the available coding dimensions, we must avoid the *dead neuron* problem, common in sparsity-enforcing models such as Sparse Autoencoders (Gao et al., 2024) and Sparse Distributed Memory (Bricken et al., 2023), where k-WTA repeatedly selects the same winners. We address this via *homeostatic plasticity* (Wu et al., 2021), raising the firing threshold of frequent winners and lowering it for rarely active units. We define the cumulative usage deviation of neuron $j$ in CA3 at time step $t$ as its activation count over all previously encoded patterns $\eta$, minus the expected firing rate $a_{CA3}$:

$$u_j^{(t)} = \sum_{i=1}^{t} \left[ \mathbb{1}(\eta_j^{(i)} = 1) - a_{CA3} \right] \tag{1}$$

Consequently, we have the objective to choose encoded pattern $\eta$ to favor the underused neurons as

$$\min_{\eta \in \{0,1\}^{D_{\text{CA3}}}} u^\top \eta \qquad \text{s.t.} \quad \text{supp}(\eta) \subseteq \mathcal{C}(\xi), \quad \|\eta\|_0 = k_{\text{CA3}} \tag{2}$$

**Minimize interference with existing patterns.** At time step $t$, the stored patterns $\eta^{(i<t)}$ induce a retrieval energy landscape $E_{t-1}(\cdot)$ in which each pattern sits at a local minimum (attractor). Therefore, encoding a new memory with minimal interference means selecting $\eta^{(t)}$ at a point of *high energy*, maximally far from existing attractors, so that subsequent Hebbian learning can carve a new attractor without colliding with stored ones:

$$\eta^{(t)} = \arg\max_{\eta} E^{(t-1)}(\eta) \quad \text{s.t} \quad \text{supp}(\eta) \subseteq C(\xi), \quad ||\eta||_0 = k_{CA3} \tag{3}$$

We focus on the sparse Hopfield network (Tsodyks & Feigel'man, 1988), whose energy function (with neural usage $u$ from Eq. (1) as the threshold) is

$$E^{(t)}(\eta) = -\frac{1}{2} \eta^\top W^{(t)} \eta + \eta^\top u^{(t)}, \tag{4}$$

where $W^{(t)} = W^{(t-1)} + (\eta^{(t)} - a_{CA3})(\eta^{(t)} - a_{CA3})^\top$ is updated incrementally via Hebbian learning of centered patterns. See Appendix E for other energy functions.

**Combined Optimization Objective.** Based on the output of the dimensionality expansion step, the neural allocation objective in Eq. (2) and the interference minimization objective in Eq. (3), we define a combined optimization objective that governs the encoding process:

$$\mathcal{L}^{(t)}(\eta) = \underbrace{-d^\top \eta}_{\text{fidelity}} + \underbrace{\alpha\, \eta^\top u^{(t-1)}}_{\text{allocation}} - \underbrace{\beta\, E^{(t-1)}(\eta)}_{\text{interference}} + \underbrace{\theta\, \|\eta\|_1}_{\text{sparsity}}, \tag{5}$$

where each term has a distinct computational role and biological correspondence (Table 1).

**Gradient-Guided Iterative $k$-WTA Search.** We solve the combined objective in Eq. (5) via the iterative procedure in Algorithm 1. At each iteration $l$, we compute a per-unit effective drive $h_j$ (Eq. (6)), which is the negative gradient of objective with respect to $\eta_j$, and re-select the top $k$ active units. Each step solves a linear minimization of the locally linearized objective over $k$-sparse binary set (Frank & Wolfe, 1956). Note that the drive $h_j$ is computed via network dynamics without explicit gradient computation. Biologically, we hypothesize that gamma oscillations, which is observed in the hippocampus to meditate engram selection during memory encoding (de Almeida et al., 2009), can provide the neural substrate for this iterative procedure.

$$h_j^{(t,l)} = (-\nabla_\eta L^{(t,l)}(\eta))_j = \underbrace{d_j^{(t)}}_{\text{external input}} - \underbrace{(\alpha - \beta) u_j^{(t-1)}}_{\text{threshold}} - \underbrace{\beta \left[ W^{(t-1)} \eta^{(t,l-1)} \right]_j}_{\text{recurrent inhibition}} \tag{6}$$

Note that in Eq. (5) we have hyperparamters $\alpha$ and $\beta$, and in Eq. (6), both of these terms are folded into $(\alpha - \beta)$ for the threshold term. Therefore, we need to set $\alpha > \beta$ for the allocation term to take effect. In our setting, we set $\alpha = 1.0$ and $\beta = 0.15$.

**Algorithm 1** Gradient-guided iterative $k$-WTA for interference-aware encoding

**Require:** DG drive $\mathbf{d}$, neuron usage $\mathbf{u}$, energy $E$, hyperparameters $\alpha, \beta, k$, max iterations $T_\gamma$

$\eta^{(t,0)} \leftarrow \text{top-}k(\mathbf{d})$ ▷ Initialize from the dimensionality expansion output

**for** $l = 1, \ldots, T_\gamma$ **do**

  $h_j^{(t,l)} = d_j^{(t)} - \alpha u_j^{(t-1)} + \beta \left[\nabla_\eta E^{(t-1)}\left(\eta^{(t,l-1)}\right)\right]_j \quad \forall j$ ▷ Per-unit gradient drive

  $\eta^{(t,l)} \leftarrow \text{top-}k(h_j^{(t,l)})$ ▷ $k$-WTA selection

  **if** $\eta^{(t,l)} = \eta^{(t,l-1)}$ **then**

    **break** ▷ Converged

  **end if**

**end for**

**return** $\eta^{(t,l)}$

| Term | Role | Potential biological relevance |
|---|---|---|
| $-d^\top \eta$ | Input fidelity (random projection) | DG mossy-fiber detonator synapse (Kumaran et al., 2016) |
| $u^\top \eta$ | Penalize overused neurons | Intrinsic excitability adaptation (Turrigiano, 2012) |
| $-E(\eta)$ | Minimize interference | SST interneuron inhibition (Honoré et al., 2021) |
| $\|\eta\|_1$ | Sparsity enforcement | PV$^+$ basket-cell global inhibition (Freund & Katona, 2007) |

Table 1: Role and suggested biological correspondence of each term in Eq. (5). Mappings motivate the design and are not mechanistic claims.

### 2.5 Learning the projection mapping via local learning rules

Given the selected code $\eta^{(t)}$, we learn the encoding map $\xi \to \eta$ and decoding map $\eta \to \xi$, parameterized by $W_{enc} \in \mathbb{R}^{D_{CA3} \times D_{EC}}$ and $W_{dec} \in \mathbb{R}^{D_{EC} \times D_{CA3}}$. For one-shot encoding, both matrices are zero-initialized and each unit $j$ uses a usage-dependent learning rate $\omega$ that decays exponentially in its raw usage $c_j^{(t)} = \sum_{i=1}^{t} \mathbb{1}(\eta_j^{(i)} = 1)$, with time constant $\tau$: $\omega_j^{(t)} = e^{-c_j^{(t)}/\tau}$.

The encoding map uses a Hebbian update and the decoding map uses delta-rule update:

$$W_{enc}^{(t)} = W_{enc}^{(t-1)} + \eta^{(t)} \xi^{(t)\top} \tag{7}$$

$$W_{dec}^{(t)} = W_{dec}^{(t-1)} + \omega^{(t)} \left(\xi^{(t)} - W_{dec}^{(t-1)} \eta^{(t)}\right) \eta^{(t)\top} \tag{8}$$

## 3 Experiment Setup

We test our proposal (ENCORE) in the online associative memory tasks, in which patterns arrive as a stream and later must be retrieved from a partial or corrupted cue. In the section below, we outline the baseline models, evaluation metrics, and evaluation dataset, with full details in the Appendix.

### 3.1 Baselines

We compare against (1) *Sparse Hopfield networks* (Tsodyks & Feigel'man, 1988), updated incrementally via $W^{(t)} = W^{(t-1)} + (\xi^{(t)} - a)(\xi^{(t)} - a)^\top$ with sparsity level $a$, (2) *Modern Hopfield networks* (Ramsauer et al., 2021), which in the online setting are trained to minimize the reconstruction error of the current pattern via backpropagation, following Krotov & Hopfield (2016), (3) *Hopfield networks with Storkey rule* (Storkey, 1997), a local incremental rule that subtracts an estimate of cross-talk noise via a local-field correction, achieving higher capacity than Hebbian learning while preserving locality, and (4) *Hopfield networks with the expansion layer*, which apply the same dimensionality expansion and number of encoded units as ENCORE (details in Appendix A).

### 3.2 Evaluation Metrics

We use two metrics mean retrieval accuracy and online retrieval capacity (details in Appendix B).

**Mean Retrieval Accuracy:** At each evaluation step $T$, we report the accuracy $\mathrm{Acc}(T)$ as the fraction of the $T$ stored patterns that are successfully retrieved from a cue (Eq. 12). A retrieved pattern

$\hat{\xi}^{(t)}$ counts as a success (Eq. 11) if cosine similarity exceeds $\tau_{\cos}$ (binary patterns) or mean-squared error (MSE) falls below $\tau_{\text{mse}}$ (continuous patterns). We use $\tau_{\cos} = 0.95$ and $\tau_{\text{mse}} = 0.01$, and report $\text{Acc}(T)$ as mean $\pm$ standard error over 5 seeds.

**Online Retrieval Capacity:** To summarize storage capacity as a single scalar, we report the online retrieval capacity $m^\star$, which is the number of stored patterns at which $\text{Acc}(T)$ first drops below $0.5$. We sweep $T$ in steps of 100 and estimate the crossing point by linear interpolation as in Eq. (13). We report $m^\star$ as mean $\pm$ standard error over 5 seeds.

### 3.3 Evaluation Dataset

We evaluate ENCORE on three different dataset with increasing difficulty: binary synthethic data, MNIST dataset, and ImageNet dataset (full details in Appendix C).

**Synthetic Patterns.** We generate random binary synthetic data with dimension $D = 2000$ and sparsity level $a = 0.05$ based on the sparsity level of layer EC II in the hippocampus (O'Reilly & McClelland, 1994). The cue is partial cue with 0.75 fraction cue.

**MNIST Dataset.** MNIST images are dense and not binary, with the mean pixel intensity is $0.1307$. To match the sparse binary format of our synthetic patterns, we pass each image through a fixed pre-processor, which applies a random ReLU projection to $D = 2000$ dimensions, then a $K$-WTA step that keeps the top $K = \lfloor aD \rfloor$ units with $a = 0.05$. All storage and retrieval happen in this binary space. To show reconstructions in pixel space, we map retrieved codes back to pixels with a linear decoder fitted by ridge regression.

**ImageNet Dataset.** We run our models on the latent space of an autoencoder (stable diffusion VAE (Rombach et al., 2022)). Image is resized to `128x128`, resulted in the latent vector of size `4x16x16`. For each random seed, we randomly sample 1000 images from 20 classes equally.

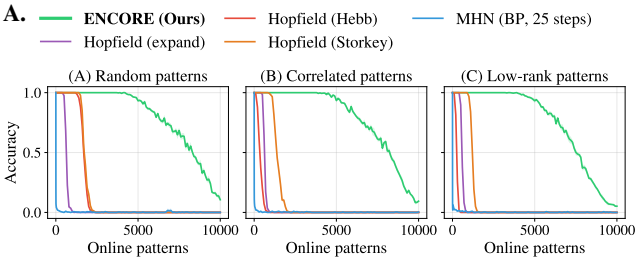
**Corruption types.** For MNIST dataset, we test three types of corruptions on the pixel space: white noise, pixel drop, right mask with corruption level 0.25. For ImageNet dataset, due to the high non-linearity of the autoencoder, we test the above corruptions in the latent space. Examples of corrupted samples can be found in Fig. 7.

## 4 Results

We evaluate ENCORE on online associative memory tasks against baseline models in Section 3.1. In Section 4.1, we show that ENCORE outperformed the baseline models by **4x** in all synthetic pattern family. In Section 4.2, we show results in real image dataset, in which ENCORE outperformed the baseline models by **5x** for MNIST and **2x** for ImageNet. In Section 4.3, we perform ablation analysis for each component in the objective function and show that each component has a distinct role in increasing the memory capacity.

### 4.1 Online Retrieval Capacity with synthetic patterns

**A.**




**B.**

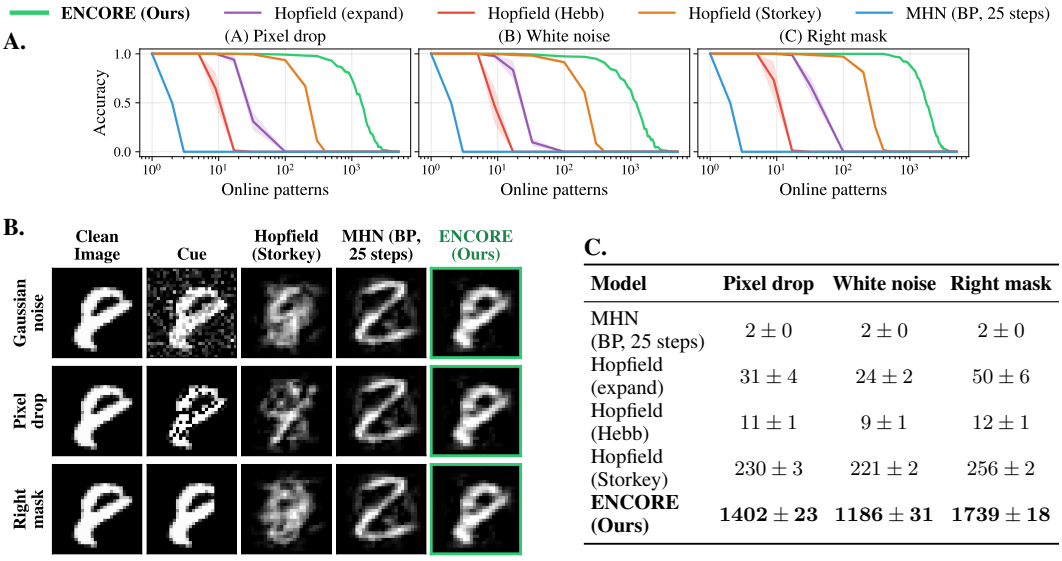
| Model | Random | Correlated | Low-rank |
|---|---|---|---|
| MHN (BP, 25 steps) | $3 \pm 1$ | $3 \pm 1$ | $3 \pm 1$ |
| Hopfield (expand) | $630 \pm 15$ | $631 \pm 10$ | $601 \pm 13$ |
| Hopfield (Hebb) | $1698 \pm 9$ | $345 \pm 8$ | $256 \pm 4$ |
| Hopfield (Storkey) | $1721 \pm 12$ | $1373 \pm 7$ | $1209 \pm 9$ |
| **ENCORE (Ours)** | $\mathbf{8836 \pm 64}$ | $\mathbf{8891 \pm 45}$ | $\mathbf{8744 \pm 35}$ |

Figure 2: **Online retrieval on synthetic patterns.** **(A)** Mean retrieval accuracy, reported every 100 time steps, as a function of the number of online patterns, across 3 synthetic pattern families: random binary sparse patterns, correlated patterns ($n_{\text{proto}} = 100$, intra-prototype cosine similarity 0.5), and low-rank patterns (generated via random projection from $\text{rank} = 32$). **(B)** Online retrieval capacity $m^*$ (mean $\pm$ s.e.m., 5 seeds).

**Online Retrieval Capacity on Synthetic Patterns.** We first report the retrieval accuracy of our model (with $D_{EC} = 2000$) on 10000 synthetic sparse binary patterns on different pattern statistics (random, correlated, low-rank) in Fig. 2. Our method improves the retrieval accuracy versus the second-best baseline (Hopfield Storkey model) by a factor of **4.8x**, **5.8x**, and **6.2x** in the setting of random, correlated, and low-rank patterns respectively. Interestingly, the Modern Hopfield Network with 2000 units with exponential storage capacity in the batch settings only have retrieval capacity around 2 to 3 patterns in the online setting since the memory matrix is trained to minimize the reconstruction error of the current pattern via backpropagation, leading to catastrophic forgetting and only able to retrieve the most recent 2-3 patterns.

### 4.2 Online Retrieval Capacity with Real Image Datasets

Section 4.1 has showed ENCORE's advantage on controlled synthetic patterns, we now test our model on real images under an online stream. We evaluate on two datasets: MNIST stored as binary patterns (Section C.2), and ImageNet $128 \times 128$ stored as continuous latents (Section C.3). In both cases patterns arrive one at a time and must be retrieved from a corrupted cue, using the same baselines and online retrieval capacity metric $m^\star$ as the synthetic experiments.




C.

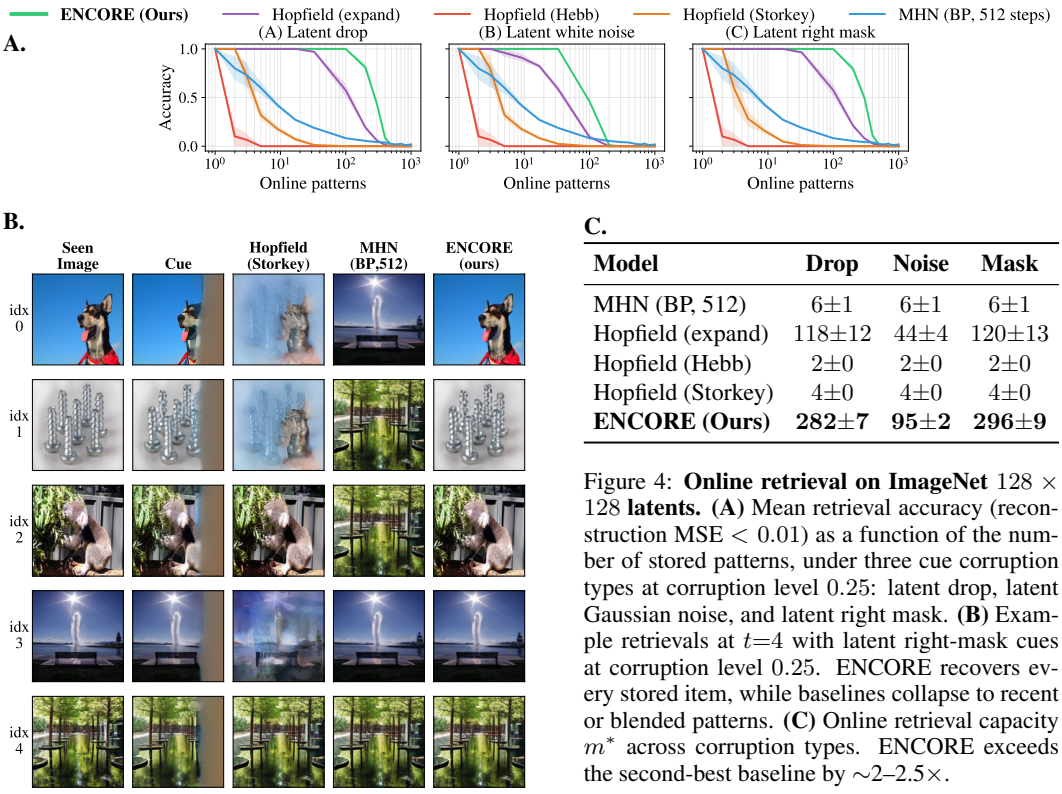
| Model | Pixel drop | White noise | Right mask |
|---|---|---|---|
| MHN (BP, 25 steps) | $2 \pm 0$ | $2 \pm 0$ | $2 \pm 0$ |
| Hopfield (expand) | $31 \pm 4$ | $24 \pm 2$ | $50 \pm 6$ |
| Hopfield (Hebb) | $11 \pm 1$ | $9 \pm 1$ | $12 \pm 1$ |
| Hopfield (Storkey) | $230 \pm 3$ | $221 \pm 2$ | $256 \pm 2$ |
| **ENCORE (Ours)** | $\mathbf{1402 \pm 23}$ | $\mathbf{1186 \pm 31}$ | $\mathbf{1739 \pm 18}$ |

Figure 3: **Online retrieval on MNIST. (A)** Mean retrieval accuracy, reported every 100 time steps, as a function of the number of stored patterns, under three cue corruption types at corruption level 0.25: pixel drop, Gaussian noise, and right mask. **(B)** Example retrievals for one stored digit under each corruption type, after storing $m = 300$ patterns. **(C)** Online retrieval capacity $m^*$ (mean $\pm$ s.e.m., 5 seeds).

**MNIST Dataset.** In Fig. 3, we evaluate on MNIST images stored as binary patterns, under three cue corruption types at level 0.25 (white noise, pixel drop, right mask). ENCORE again outperforms substantially over the second-best baseline (Hopfield Storkey), improving online retrieval capacity by a factor of **6.17x**, **5.37x**, and **6.79x** for pixel drop, Gaussian noise, and right mask respectively. The advantage over the standard Hopfield baselines is larger (over $100\times$), and the backpropagation-trained Modern Hopfield Network again collapses to retrieving only the 2 most recent patterns.

**ImageNet Dataset.** Finally, in Fig. 4, we evaluate on ImageNet $128 \times 128$ images encoded as continuous latents (from stable diffusion variational autoencoder (Rombach et al., 2022)), retrieving each pattern from a corrupted cue at level 0.25 (latent white noise, latent drop, latent right mask). Here retrieval is scored by reconstruction MSE ($< 0.01$). Fig. 4A shows ENCORE attains the highest capacity across all three corruption types, exceeding the second-best baseline (Hopfield expand) by roughly **2.39x**, **2.16x**, and **2.46x** for latent drop, Gaussian noise, and right mask respectively. In Fig. 4B, we show the example retrieved images from three different class of models: Hopfield (Storkey), MHN trained with backpropagation, and ENCORE. Interestingly, we can observe different failure modes: Hopfield (Storkey) produces spurious attractors, in which two distinct patterns are

blended (see sample `idx0` and `idx1`); MHN trained with backpropagation leads to catastrophic forgetting, with recent patterns overwritting earlier ones (sample `idx4` overwritten `idx1`, `idx2` and sample `idx3` overwritten `idx0`). In contrast, ENCORE reconstructs every stored item faithfully.

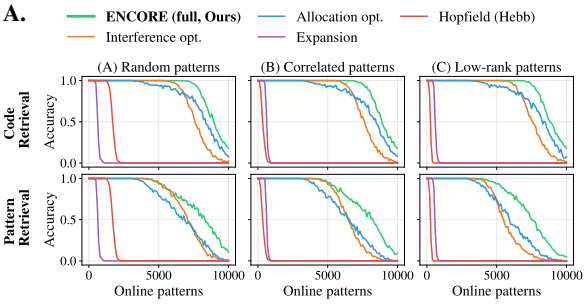



**C.**

| Model | Drop | Noise | Mask |
|---|---|---|---|
| MHN (BP, 512) | 6±1 | 6±1 | 6±1 |
| Hopfield (expand) | 118±12 | 44±4 | 120±13 |
| Hopfield (Hebb) | 2±0 | 2±0 | 2±0 |
| Hopfield (Storkey) | 4±0 | 4±0 | 4±0 |
| **ENCORE (Ours)** | **282±7** | **95±2** | **296±9** |

Figure 4: **Online retrieval on ImageNet** $128 \times 128$ **latents. (A)** Mean retrieval accuracy (reconstruction MSE $< 0.01$) as a function of the number of stored patterns, under three cue corruption types at corruption level 0.25: latent drop, latent Gaussian noise, and latent right mask. **(B)** Example retrievals at $t$=4 with latent right-mask cues at corruption level 0.25. ENCORE recovers every stored item, while baselines collapse to recent or blended patterns. **(C)** Online retrieval capacity $m^*$ across corruption types. ENCORE exceeds the second-best baseline by ∼2–2.5×.

### 4.3 Ablation Analysis




**B.**

| Model | Random | Correlated | Low-rank |
|---|---|---|---|
| Expansion | $630 \pm 15$ | $631 \pm 10$ | $601 \pm 13$ |
| Hopfield (Hebb) | $1698 \pm 9$ | $345 \pm 8$ | $256 \pm 4$ |
| Interference opt. | $7616 \pm 34$ | $7628 \pm 39$ | $7481 \pm 43$ |
| Allocation opt. | $8282 \pm 39$ | $8431 \pm 75$ | $8307 \pm 54$ |
| **ENCORE (full)** | $\mathbf{8836 \pm 64}$ | $\mathbf{8891 \pm 45}$ | $\mathbf{8744 \pm 35}$ |

Figure 5: **Component ablation of ENCORE encoding objective. (A)** Online retrieval accuracy as a function of the number of stored patterns, across three synthetic pattern families: random, correlated, and low-rank. We compare the full model against the following ablations: interference optimization only ($\alpha = 0$), allocation optimization only ($\beta = 0$), and expansion only ($\alpha = \beta = 0$), with the Hopfield (Hebb) baseline for reference. **(B)** Online retrieval capacity $m^*$ (mean ± s.e.m., 5 seeds).

We ablate each component in Eq. (5) and measure the resulting retrieval capacity (Fig. 5). We compare: **Expansion only** ($\alpha = \beta = 0$); **Allocation only** ($\beta = 0$); **Interference only** ($\alpha = 0$); the **Full model**; and the **Sparse Hopfield (Hebbian)** baseline. First, we showed in Fig. 5B that the full model outperforms other ablated models in all three pattern family (random, correlated, and low-rank), confirming that both optimization terms in Section 2.4 are necessary for good retrieval performance. Second, each ablated model show distinct failure mode specific to its optimization objective. Removing the interference term causes accuracy to drop early. In the encoded-pattern

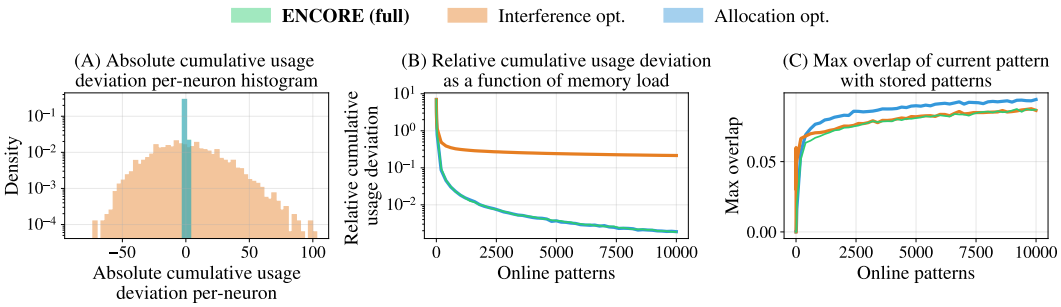



Figure 6: **Memory allocation and interference analysis**. **(A)** Distribution of per-neuron cumulative usage deviation at a memory load of $t = 5000$ patterns. **(B)** Relative cumulative usage deviation as a function of the number of online patterns. **(C)** Maximum overlap between the currently encoded code and all previously stored codes, as a function of the number of online patterns.

retrieval row of Fig. 5A, the allocation-only model drops below $0.95$ accuracy at $t = 4400$, $4600$ and $4500$ patterns (random, correlated, low-rank), well before the interference-only model at $t = 6400$, $6400$ and $6200$, and far before the full model at $t \approx 7600$. Fig. 6C identifies the cause: without the interference term, the code selected for the current pattern overlaps its nearest stored neighbour more ($0.094$ vs. $0.087$ at $t = 10^4$), so collisions with existing memories appear at a lower load.

Removing the allocation term instead causes accuracy to decrease faster once degradation begins. Although the allocation-only model fails first, its accuracy declines more gradually and overtakes the interference-only model at $t \approx 6800$. Fig. 6A and B explain why. In Fig. 6A, in models with the allocation term, per-neuron cumulative usage stays tightly around its expected value: at $t = 5000$ the usage deviation is confined to $[-1, 2]$ (s.d. $0.37$), whereas the interference-only model spreads it over $[-71, 107]$ (s.d. $24.1$). This trend is also observed at all time steps (see Fig. 8). In Fig. 6B, the relative usage deviation (Eq. 22) differs by two orders of magnitude at $t = 10^4$ ($0.0019$ vs. $0.21$). The allocation term therefore keeps the population load-balanced, so no subset of neurons is overused and capacity is spent evenly across the network. Together, these findings suggests the distinct role of each optimization term in maximizing the online memory retrieval capacity.

## 5 Discussion

We introduce ENCORE, an online associative memory system built on a shift in perspective from the memory engram allocation literature (Josselyn & Frankland, 2018): rather than treating memory patterns as random or given, we cast the selection of the encoded memory as an optimization procedure. Prior work focuses on either retrieval dynamics (Krotov & Hopfield, 2016; Ramsauer et al., 2021) or learning rules (Storkey, 1997), leaving the choice of which patterns to store as given. By formulating encoding as maximization of a combined fidelity, allocation, and interference objective, we improve sequential capacity by **5×** over the second-best baseline on MNIST and **2×** on ImageNet 128x128, with ablations confirming that both the allocation and interference terms contribute.

This view has a natural biological correspondence: neurons compete for recruitment into an engram, and more excitable, underused neurons are preferentially allocated, naturally preventing new memories from colliding with recent ones (Josselyn & Tonegawa, 2020). The expansion and sparsification step maps to hippocampal anatomy (Andersen et al., 2006), and each objective term admits a suggestive correspondence to known inhibitory and homeostatic mechanisms (Table 1).

Several limitations remain. First, we use a fixed sparsity level taken from hippocampal measurements, rather than adapting it to dataset statistics (e.g., information content per pixel). Second, our tasks consist of discrete items, so the objective need to only minimize interference between separate patterns. Perception, however, is continuous in time and space. A more realistic procedure would minimize interference while preserving the coherence of temporally and spatially contiguous experience (Cai et al., 2016). Extending the encoding optimization to structured, trajectory-like inputs, allocating overlapping but distinguishable codes for continuous experience, is therefore a promising direction, connecting ENCORE to spatiotemporal memory in navigation and robotics.

### AI USE STATEMENT

In this work, we used generative AI tools for design or provide feedback on research methodology or experiments, clean and reformat dataset, implement methods. We have not used generative AI tools for help develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, support qualitative and thematic data analysis, interpret results, and the rest of the required disclosure tasks, including generate synthetic data sets, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, assist with translation, are not applicable to this work. Additionally, we used generative AI tools for create or modify scientific figures or images, create or edit software code, summarize or analyse existing literature, edit a research paper to improve readability. We have reviewed all AI-assisted work. We validated LLM literature survey manually. LLM-generated code was verified and tested by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Per Andersen, Richard Morris, David Amaral, Tim Bliss, and John O'Keefe (eds.). *The Hippocampus Book*. Oxford University Press, 12 2006. ISBN 9780195100273. doi: 10.1093/acprof:oso/9780195100273.001.0001. URL https://doi.org/10.1093/acprof:oso/9780195100273.001.0001.

Baktash Babadi and Haim Sompolinsky. Sparseness and expansion in sensory representations. *Neuron*, 83(5):1213–1226, 2014.

Trenton Bricken, Xander Davies, Deepak Singh, Dmitry Krotov, and Gabriel Kreiman. Sparse distributed memory is a continual learner. In *The Eleventh International Conference on Learning Representations*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Denise J Cai, Daniel Aharoni, Tristan Shuman, Justin Shobe, Jeremy Biane, Weilin Song, Brandon Wei, Michael Veshkini, Mimi La-Vu, Jerry Lou, et al. A shared neural ensemble links distinct contextual memories encoded close in time. *Nature*, 534(7605):115–118, 2016.

Sanjoy Dasgupta, Charles F Stevens, and Saket Navlakha. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796, 2017.

Licurgo de Almeida, Marco Idiart, and John E Lisman. A second function of gamma frequency oscillations: an e%-max winner-take-all mechanism selects which cells fire. *The Journal of neuroscience*, 29(23):7497–7503, 2009.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Tamás F Freund and István Katona. Perisomatic inhibition. *Neuron*, 56(1):33–42, 2007.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2024.

Jin-Hee Han, Steven A Kushner, Adelaide P Yiu, Christy J Cole, Anna Matynia, Robert A Brown, Rachael L Neve, John F Guzowski, Alcino J Silva, and Sheena A Josselyn. Neuronal competition and selection during memory formation. *science*, 316(5823):457–460, 2007.

Eve Honoré, Abdessattar Khlaifia, Anthony Bosson, and Jean-Claude Lacaille. Hippocampal somatostatin interneurons, long-term synaptic plasticity and memory. *Frontiers in neural circuits*, 15:687558, 2021.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Sheena A Josselyn and Paul W Frankland. Memory allocation: mechanisms and function. *Annual review of neuroscience*, 41(1):389–413, 2018.

Sheena A. Josselyn and Susumu Tonegawa. Memory engrams: Recalling the past and imagining the future. *Science*, 367(6473):eaaw4325, 2020. doi: 10.1126/science.aaw4325. URL https://www.science.org/doi/abs/10.1126/science.aaw4325.

Mohadeseh Shafiei Kafraj, Dmitry Krotov, and Peter E. Latham. A biologically plausible dense associative memory with exponential capacity. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mRZOayQL1i.

Mitsuo Kawato, Shogo Ohmae, Huu Hoang, and Terry Sanger. 50 years since the marr, ito, and albus models of the cerebellum. *Neuroscience*, 462:151–174, 2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Dmitry Krotov and John J Hopfield. Large associative memory problem in neurobiology and machine learning. In *International Conference on Learning Representations*, 2021.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20 (7):512–534, 2016.

Carlo Lucibello and Marc Mézard. Exponential capacity of dense associative memories. *Physical Review Letters*, 132(7):077301, 2024.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

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

Christos H Papadimitriou. Random projection in the brain and computation with assemblies of neurons. In *10th Innovations in Theoretical Computer Science Conference*, 2019.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, et al. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021.

Edmund T Rolls. Pattern separation, completion, and categorisation in the hippocampus and neocortex. *Neurobiology of learning and memory*, 129:4–28, 2016.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Amos Storkey. Increasing the capacity of a hopfield network without sacrificing functionality. In *International Conference on Artificial Neural Networks*, pp. 451–456. Springer, 1997.

Mikhail V Tsodyks and Mikhail V Feigel'man. The enhanced storage capacity in neural networks with low activity level. *EPL (Europhysics Letters)*, 6(2):101–105, 1988.

Gina Turrigiano. Homeostatic synaptic plasticity: local and global mechanisms for stabilizing neuronal function. *Cold Spring Harbor perspectives in biology*, 4(1):a005736, 2012.

Chi-Hong Wu, Raul Ramos, Donald B Katz, and Gina G Turrigiano. Homeostatic synaptic scaling establishes the specificity of an associative memory. *Current biology*, 31(11):2274–2285, 2021.

# A MODELS

## A.1 BASELINE MODELS

1. **Traditional Sparse Hopfield networks** (Hopfield, 1982; Tsodyks & Feigel'man, 1988). While the original model updates the recurrent weight matrix $W$ in one-shot in the batch storage setting, in the online setting, we update the weight matrix incrementally as each pattern $\xi$ comes via $W^{(t)} = W^{(t-1)} + (\xi^{(t)} - a)(\xi^{(t)} - a)^\top$ with $a$ being the sparsity level.
2. **Modern Hopfield networks** (Ramsauer et al., 2021). The original model is developed for the batch storage setting and store all samples in a memory matrix. For the online storage setting, following the protocol from (Krotov & Hopfield, 2016), we randomly initialize the memory matrix and train the network to minimize the reconstruction error given the current pattern using backpropgation
3. **Hopfield Networks with Storkey learning rule** (Storkey, 1997): The Storkey rule is a local and incremental learning rule that improves over the Hebbian rule by subtracting an estimate of the cross-talk noise from previously-stored patterns through a local-field correction term. As each pattern $\xi^{(t)}$ arrives, the weights are updated via

$$W_{ij}^{(t)} = W_{ij}^{(t-1)} + \xi_i^{(t)}\xi_j^{(t)} - \xi_i^{(t)}h_{ji}^{(t)} - h_{ij}^{(t)}\xi_j^{(t)},$$

   where $h_{ij}^{(t)} = \sum_{k\neq i,j} W_{ik}^{(t-1)}\xi_k^{(t)}$ is the local field at neuron $i$ ignoring the contribution from neuron $j$, and $N$ is the number of neurons. Because the rule is inherently incremental, it is directly suitable to the online storage setting without modification, and is known to achieve higher capacity than the standard Hebbian rule while preserving locality.
4. **Hopfield Networks with a Random Expansion Layer**: To account for the impact of dimensionality expansion, we also include a variant of Hopfield networks with the dimensionality expansion step via random projection and with the same number encoded units as our approach.

## A.2 ADDITIONAL DETAILS

### A.2.1 SKIP CONNECTION

The cue is available again when the memory is read out: in the hippocampus, EC projects directly to CA1 as well as through DG and CA3. We model this with a skip connection that combines the EC cue $\tilde{\boldsymbol{\xi}}$ with the network's EC reconstruction $\hat{\boldsymbol{\xi}}$ at readout time. It changes neither storage nor the CA3 retrieval dynamics.

**Sparse binary patterns.** For sparse binary patterns (the synthetic datasets and MNIST) we use an additive skip followed by $K$-WTA:

$$\boldsymbol{\xi}^{\text{out}} = \text{kWTA}_K\left(\hat{\boldsymbol{\xi}} + \gamma\,(2\tilde{\boldsymbol{\xi}} - \mathbf{1})\right), \tag{9}$$

where $\gamma \geq 0$ is the skip strength. Each unit that is active in the cue receives a bonus of $+\gamma$, and each inactive unit receives $-\gamma$. When the network's readout is confident, the skip has little effect; when two units are close in the readout, the cue decides which one wins. The $K$-WTA step guarantees the output is again a binary pattern with exactly $K$ active units.

**Continuous latents (ImageNet).** For occlusion cues (pixel drop and right mask), missing latent entries are exactly zero, so $\mathbf{o} = \mathbb{1}[\tilde{\boldsymbol{\xi}} \neq 0]$ marks the entries that were observed. On those entries we clamp the output toward the cue:

$$\boldsymbol{\xi}^{\text{out}} = (\mathbf{1} - \gamma\,\mathbf{o}) \odot \hat{\boldsymbol{\xi}} + \gamma\,\mathbf{o} \odot \tilde{\boldsymbol{\xi}}, \tag{10}$$

With $\gamma = 1$, observed entries are copied directly from the cue, and the memory fills in only the missing ones. For white-noise cues every entry is corrupted, so no entry can be trusted more than the readout. Clamping would simply copy the noisy cue, so we use no skip connection and set $\boldsymbol{\xi}^{\text{out}} = \hat{\boldsymbol{\xi}}$.

The skip is applied in the same way to every method, including the Hopfield and modern Hopfield baselines, so that comparisons are fair. Unless stated otherwise, we use $\gamma = 0.25$ for the synthetic and MNIST experiments and $\gamma = 1$ for ImageNet.

# B Evaluation Metrics

**Mean Retrieval Accuracy** At each evaluation interval, we report the mean retrieval accuracy of all previously seen patterns. For each pattern, the retrieval is successed if the cosine similarity score (in case of binary patterns) is higher than a threshold or the mean-squared error (in case of continuous patterns) is lower than a threshold. Concretely, let $\{\xi^{(1)}, \ldots, \xi^{(T)}\} \in \mathbb{R}$ denote the set of patterns stored up to evaluation step $T$, and let $\hat{\xi}^{(t)}$ denote the pattern retrieved by the model when queried with a (possibly noisy or partial) cue $\tilde{\xi}^{(t)}$. We define the per-pattern success indicator as

$$s^{(t)} \;=\; \begin{cases} \mathbb{1}\Big[\cos\big(\hat{\xi}^{(t)}, \xi^{(t)}\big) \;\geq\; \tau_{\cos}\Big], & \text{(binary patterns)} \\ \mathbb{1}\Big[\dfrac{1}{d}\,\big\|\hat{\xi}^{(t)} - \xi^{(t)}\big\|_2^2 \;\leq\; \tau_{\text{mse}}\Big], & \text{(continuous patterns)} \end{cases} \tag{11}$$

where $\cos(u, v) = \frac{\langle u, v\rangle}{\|u\|_2\,\|v\|_2}$ is the cosine similarity, and $\tau_{\cos}, \tau_{\text{mse}}$ are fixed retrieval thresholds ($\tau_{\cos} = 0.95$ and $\tau_{\text{mse}} = 0.01$). The mean retrieval accuracy at evaluation step $T$ is then

$$\text{Acc}(T) \;=\; \frac{1}{T}\sum_{t=1}^{T} s^{(t)} \tag{12}$$

We report $\text{Acc}(T)$ averaged over 5 random seeds, with shaded regions indicating standard error across seeds.

**Online Retrieval Capacity** While mean retrieval capacity $\text{Acc}(T)$ characterize the retrieval performance at a fixed number of stored patterns, we summarize the model's overall storage capability in the online associative memory task as a single scalar quantity, the *sequential capacity* $m^*$. We define $m^*$ the number of stored patterns at which mean retrieval accuracy first falls below $0.5$ as patterns accumulate. Concretely, we evaluate $\text{Acc}(T)$ along the increasing-$T$ sweep (every $100$ time steps) and define $m^\star$ as the crossing point where accuracy drops from $\geq 0.5$ to $< 0.5$. Since the sweep is sampled on a grid, we estimate $m^\star$ by linear interpolation between the last evaluation point with $\text{Acc} \geq 0.5$ and the first with $\text{Acc} < 0.5$:

$$m^\star = m_{\text{hi}} + (m_{\text{lo}} - m_{\text{hi}})\,\frac{\text{Acc}_{\text{hi}} - 0.5}{\text{Acc}_{\text{hi}} - \text{Acc}_{\text{lo}}}, \tag{13}$$

where $(m_{\text{hi}}, \text{Acc}_{\text{hi}})$ and $(m_{\text{lo}}, \text{Acc}_{\text{lo}})$ are the accuracies between the threshold. Intuitively, $m^\star$ is the number of memories a model can store before more than half can no longer be reliably retrieved. We report $m^\star$ as mean $\pm$ standard error over $5$ seeds.

# C Evaluation Dataset

## C.1 Binary Synthetic Patterns

**Random Synthetic Data.** Random patterns are sparse binary vectors with a fixed number of active units. Given a pattern dimension $D$ and an activity level $a \in (0, 1]$, each pattern $\boldsymbol{\xi}^\mu \in \{0, 1\}^D$,

$\mu = 1, \ldots, N$, has exactly $K = \lfloor aD \rfloor$ entries set to 1 and all other entries set to 0. The active set $\mathcal{A}^\mu \subset \{1, \ldots, N\}$, with $|\mathcal{A}^\mu| = K$, is drawn uniformly at random without replacement, and is sampled independently for each pattern. Because active sets are drawn independently, patterns share no structure beyond chance overlap.

**Correlated Synthetic Data.** Correlated patterns are generated by injecting perturbation to a small set of prototypes. We first draw $C$ prototype active sets $\mathcal{B}^1, \ldots, \mathcal{B}^C$, each consisting of $K = \lfloor aD \rfloor$ indices sampled uniformly at random without replacement, independently across prototypes. Each prototype then gives rise to $\lfloor N/C \rfloor$ patterns. A similarity level $s \in [0, 1]$ sets the number of perturbed units to $M = \lfloor (1 - s)K \rfloor$. To generate pattern $\mu$ from prototype $c$, we:

1. Retain a uniformly random subset $\mathcal{R}^\mu \subset \mathcal{B}^c$ of $K - M$ prototype units.
2. Draw $M$ new active units $\mathcal{N}^\mu$ uniformly without replacement from $\{1, \ldots, N\} \setminus \mathcal{R}^\mu$.
3. Set $\xi_i^\mu = 1$ for $i \in \mathcal{R}^\mu \cup \mathcal{N}^\mu$ and $\xi_i^\mu = 0$ otherwise.

Every pattern therefore keeps exactly $K$ active units, so the sparsity matches the random dataset, and it shares at least $K - M \approx sK$ active units with its prototype. Patterns derived from the same prototype are strongly correlated.

**Low-rank Synthetic Data.** Low-rank patterns are obtained by binarizing a set of continuous scores that lie in an $r$-dimensional subspace of $\mathbb{R}^D$. We draw a fixed loading matrix $\mathbf{W} \in \mathbb{R}^{D \times r}$ with i.i.d. entries $W_{ij} \sim \mathcal{N}(0, 1/r)$, shared across all patterns, and an independent latent code $\mathbf{z}^\mu \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_r)$ for each pattern. The score vector $\mathbf{u}^\mu = \mathbf{W}\mathbf{z}^\mu \in \mathbb{R}^D$ is binarized with a $K$-winners-take-all rule, where $K = \lfloor aD \rfloor$:

$$\xi_i^\mu = \begin{cases} 1 & \text{if } u_i^\mu \text{ is among the } K \text{ largest entries of } \mathbf{u}^\mu, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Every pattern therefore has exactly $K$ active units, matching the sparsity of the other datasets. The score matrix $[\mathbf{u}^1, \ldots, \mathbf{u}^N]$ has rank at most $r$. The binary pattern matrix is not itself low-rank, but its structure is inherited from this $r$-dimensional latent space.

### C.2 MNIST Dataset

**MNIST Pre-processor.** We use the MNIST test split (10,000 grayscale $28 \times 28$ digits). Each image is flattened to $\mathbf{x}^\mu \in [0, 1]^{784}$ using the standard `ToTensor` scaling, with no binarization or thresholding. Since raw pixel vectors are dense and highly correlated, before storage we map them into a sparse binary EC code with a fixed random feature map followed by $K$-winners-take-all:

$$\mathbf{h}^\mu = \mathrm{ReLU}(\mathbf{W}_{\mathrm{pp}}\, \mathbf{x}^\mu), \qquad \xi_i^\mu = \begin{cases} 1 & \text{if } h_i^\mu \text{ is among the } K \text{ largest entries of } \mathbf{h}^\mu, \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

where $\mathbf{W}_{\mathrm{pp}} \in \mathbb{R}^{D \times 784}$ has i.i.d. entries $(\mathbf{W}_{\mathrm{pp}})_{ij} \sim \mathcal{N}(0, 1/784)$, is drawn once per seed and kept frozen, and has no bias term. We use $D = 2000$ and $a = 0.05$ ($K = 100$), so MNIST codes have the same format and sparsity as the synthetic patterns.

To map a retrieved EC code back to pixel space for visualization and pixel-level error, we fit a linear decoder $\mathbf{D}_{\mathrm{pp}} \in \mathbb{R}^{784 \times D}$ by closed-form ridge regression:

$$\mathbf{D}_{\mathrm{pp}} = \arg\min_{\mathbf{D}} \sum_{\nu} \left\| \mathbf{x}^\nu - \mathbf{D}\, \boldsymbol{\xi}^\nu \right\|_2^2 + \lambda \left\| \mathbf{D} \right\|_F^2, \qquad \lambda = 10, \tag{16}$$

The fit uses the remaining $10{,}000 - N$ test images, which are disjoint from the stored set. The encoder and decoder are fixed before any memory is stored and are never updated while the network learns.

### C.3 ImageNet Dataset

We evaluate on natural images using ImageNet-1k. For each seed we sample $C = 20$ classes uniformly without replacement from the 1,000 classes. From each class we then draw $N/C$ images

uniformly without replacement, and we shuffle the resulting $N$ images so the classes are interleaved across the storage timestep. We use $N = 1000$, i.e. 50 images per class. Each image is converted to RGB, resized to $128 \times 128$ (without cropping), scaled to $[0, 1]$, and then normalized to $[-1, 1]$.

Rather than storing pixels, we store the latent of a frozen pretrained autoencoder: the Stable Diffusion VAE (`stabilityai/sd-vae-ft-ema`). The encoder maps each image to a posterior $q(\mathbf{z} \mid \mathbf{x}^\mu)$. We draw one sample from it, multiply by the VAE scaling factor $0.18215$, and obtain a latent tensor $\mathbf{Z}^\mu \in \mathbb{R}^{4\times16\times16}$. This tensor is flattened (channel-major) into the stored pattern $\boldsymbol{\xi}^\mu = \mathrm{vec}(\mathbf{Z}^\mu) \in \mathbb{R}^D$ with $D = 1024$. Unlike MNIST, no random-feature encoder or $K$-WTA is applied, so ImageNet patterns are continuous-valued. To reconstruct an image, a retrieved pattern is reshaped to $4 \times 16 \times 16$, divided by the scaling factor, and passed through the frozen VAE decoder.

### C.4 Corruption Types

Retrieval is cued with a corrupted version of a stored item. The corruption level $c \in [0, 1]$ is shared across all types, and we use $c = 0.25$ unless stated otherwise. Each cue is generated once per pattern with an independent random draw. Examples of corrupted samples can be found in Fig. 7.

- For MNIST, the corruption is applied to the pixel image $\mathbf{x}^\mu$. The corrupted image $\tilde{\mathbf{x}}^\mu$ then goes through the same fixed encoder (random ReLU features followed by $K$-WTA) to give the EC cue $\tilde{\boldsymbol{\xi}}^\mu$.
- For ImageNet, the corruption is applied directly to the VAE latent $\mathbf{Z}^\mu \in \mathbb{R}^{4\times16\times16}$ to give $\tilde{\mathbf{Z}}^\mu$, and then $\tilde{\boldsymbol{\xi}}^\mu = \mathrm{vec}(\tilde{\mathbf{Z}}^\mu)$. We corrupt in latent space because a pixel-space occlusion, once passed through the nonlinear encoder, can produce a large and uncontrolled shift in the latent. The spatial masks are shared across all 4 latent channels.

**White noise.** For MNIST, i.i.d. Gaussian noise is added to every pixel and the result is clipped to the valid range: $\tilde{\mathbf{x}}^\mu = \mathrm{clip}\big(\mathbf{x}^\mu + c\,\boldsymbol{\epsilon},\, 0,\, 1\big)$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{784})$. For ImageNet, the noise is scaled to the spread of each latent so that $c$ is a relative, scale-free noise level: $\tilde{\mathbf{Z}}^\mu = \mathbf{Z}^\mu + c\,\sigma(\mathbf{Z}^\mu)\,\boldsymbol{\epsilon}$, where $\sigma(\mathbf{Z}^\mu)$ is the standard deviation of the entries of $\mathbf{Z}^\mu$ and $\boldsymbol{\epsilon}$ is standard Gaussian.

**Pixel drop.** A random subset of locations is occluded, i.e. set to zero. For MNIST, each of the 784 pixels is dropped independently with probability $c$. For ImageNet, each of the $16 \times 16$ latent spatial locations is dropped independently with probability $c$, and the drop applies to all 4 channels at that location.

**Right mask.** The rightmost $\mathrm{round}(cW)$ columns are set to zero, where $W$ is the input width. For MNIST $W = 28$ pixels, so 7 columns are masked at $c = 0.25$. For ImageNet $W = 16$ latent columns, masked across all channels, so 4 columns are masked at $c = 0.25$. Unlike pixel drop, this removes a contiguous region: the right side of the item is missing entirely and must be completed from memory.

## D Neural Usage Interpretation

The neural usage definition $u_j(t)$ in Eq. (1) measures how much neuron $j$'s activity deviates from its expected level under the sparsity prior $a_{CA3}$. Let $m_j(t)$ denote the number of patterns that neuron $j$ responds to (i.e., the patterns $i$ for which $\eta_j^{(i)} = 1$ in the first $t$ patterns). Concretely, we have:

$$m_j(t) = \big|\{\, i : 1 \le i \le t,\ \eta_j^{(i)} = 1 \,\}\big| = \sum_{i=1}^{t} \mathbb{1}\Big(\eta_j^{(i)} = 1\Big) \tag{17}$$

We can express $u_j(t) = m_j(t) - a_{CA3}\, t$ which explicitly measures the deviation between the actual number of patterns $m_j(t)$ that neuron $j$ responds to versus the expected number of patterns $a_{CA3}\, t$ under the defined sparsity level.

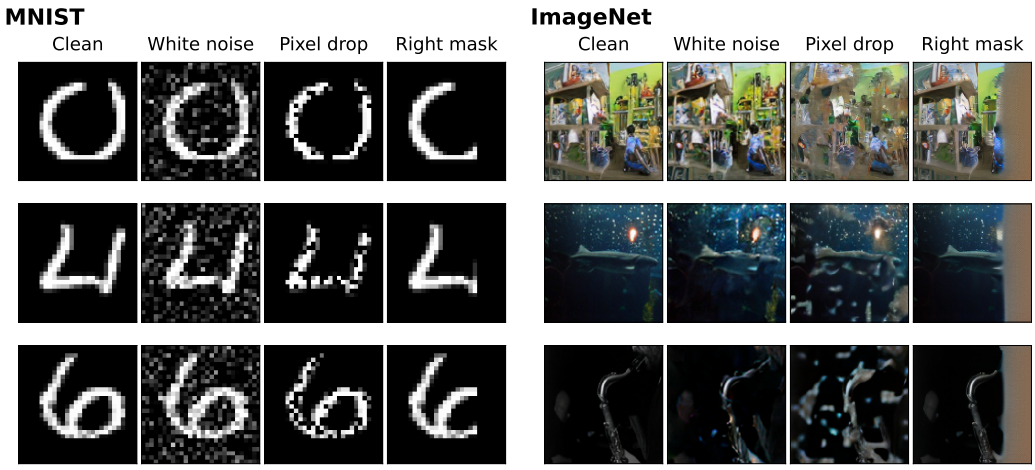



Figure 7: **Examples of the corruption types used to generate retrieval cues**, at corruption level $c = 0.25$. Each row shows one stored item: first the clean item, then the cue produced by each corruption type. **Left (MNIST):** corruptions are applied in pixel space. **Right (ImageNet):** corruptions are applied to the $4 \times 16 \times 16$ VAE latent.

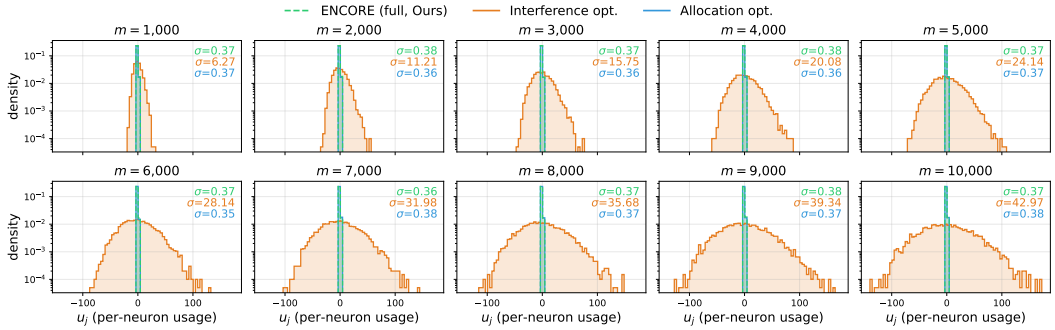



Figure 8: **Distribution of per-neuron CA3 usage $u_j$ as memory load $m$ increases**. With allocation, both ENCORE (full) and Allocation opt. keep $u_j$ tightly concentrated around zero at every load ($\sigma \approx 0.37$), meaning storage is spread evenly across CA3 neurons. Interference opt. has no allocation, and its usage distribution widens steadily with load ($\sigma$ grows from 6.3 to 43.0), so some neurons are heavily overused and others underused. Each panel lists the standard deviation $\sigma$ for each variant.

$$u_j(t) = \sum_{i=1}^{t} \left[ \mathbb{1}\left(\eta_j^{(i)} = 1\right) - a_{CA3} \right] \tag{18}$$

$$= \sum_{\substack{1 \le i \le t \\ \eta_j^{(i)}=1}} (1 - a_{CA3}) \; + \sum_{\substack{1 \le i \le t \\ \eta_j^{(i)}=0}} (-a_{CA3}) \tag{19}$$

$$= m_j(t)\,(1 - a_{CA3}) + \big(t - m_j(t)\big)(-a_{CA3}) \tag{20}$$

$$= m_j(t) - a_{CA3}\, t \tag{21}$$

We can see that $u_j(t)$ measures the absolute deviation of neuron usage from the expected usage. We can also define the relative deviation from the expected usage as

$$\tilde{u}_j(t) = \frac{u_j(t)}{a_{CA3} t} \tag{22}$$

# E INTERPRETATION OF OTHER ENERGY FUNCTIONS

All experiments in this paper use the sparse Hopfield energy of Tsodyks & Feigel'man (1988) (Eq. 4) as the interference term. The encoding formulation itself does not depend on this choice. Algorithm 1 uses the energy only through its gradient $\nabla_\eta E^{(t-1)}$ in the drive $h_j$ (Eq. 6), so any differentiable energy over CA3 codes can be substituted without changing the fidelity term, the allocation term, or the iterative $k$-WTA selection. In this appendix we show how the formulation extends to other associative-memory energies. We (i) write a family of energies in a common form, (ii) derive the recurrent-inhibition term each one produces, and (iii) describe the encoding behaviour each one implies. We do not evaluate these variants empirically and leave them to future work.

## E.1 A UNIFIED FORM: ENERGIES AS FUNCTIONS OF PATTERN OVERLAPS

Let $\eta^{(1)}, \dots, \eta^{(t-1)}$ be the stored CA3 codes and $a \equiv a_{\mathrm{CA3}}$. The centered overlap of a candidate code $\eta$ with stored pattern $i$ is

$$m_i(\eta) \;=\; \big(\eta^{(i)} - a\mathbf{1}\big)^\top \eta \;=\; \big|\mathrm{supp}(\eta^{(i)}) \cap \mathrm{supp}(\eta)\big| \;-\; a\,k_{\mathrm{CA3}}. \tag{23}$$

The second equality holds under the constraint $\|\eta\|_0 = k_{\mathrm{CA3}}$. Thus $m_i$ counts the active units that $\eta$ shares with $\eta^{(i)}$ beyond the $a\,k_{\mathrm{CA3}}$ expected by chance. We consider energies of the form

$$E^{(t-1)}(\eta) \;=\; -\,\Phi\big(m_1(\eta), \dots, m_{t-1}(\eta)\big) \;+\; \eta^\top u^{(t-1)}, \tag{24}$$

Here $\Phi$ aggregates the overlaps, and the threshold term $\eta^\top u^{(t-1)}$ is the same as in Eq. (4). Because $\partial m_i / \partial \eta = \eta^{(i)} - a\mathbf{1}$, the gradient is

$$\nabla_\eta E^{(t-1)}(\eta) \;=\; -\sum_{i<t} \omega_i(\eta)\,\big(\eta^{(i)} - a\mathbf{1}\big) \;+\; u^{(t-1)}, \qquad \omega_i(\eta) \;=\; \frac{\partial \Phi}{\partial m_i}. \tag{25}$$

Substituting this into Eq. (6) gives the general iterative $k$-WTA drive

$$h_j^{(t,l)} \;=\; \underbrace{d_j^{(t)}}_{\text{external input}} \;-\; \underbrace{(\alpha-\beta)\,u_j^{(t-1)}}_{\text{threshold}} \;-\; \underbrace{\beta \sum_{i<t} \omega_i\big(\eta^{(t,l-1)}\big)\,\big(\eta_j^{(i)} - a\big)}_{\text{recurrent inhibition}}. \tag{26}$$

The choice of energy therefore changes only one thing: the weights $\omega_i$ that set how strongly each stored memory inhibits the units it uses. Every energy below keeps the same external-input and threshold terms and the same $k$-WTA selection step.

When $\Phi$ is convex in $\eta$ , maximizing $E$ is equivalent to minimizing a convex function. The full objective in Eq. (5) then stays convex once the binary $k$-sparse set is relaxed to its convex hull $\{\eta \in [0,1]^{D_{\mathrm{CA3}}} : \mathbf{1}^\top \eta = k_{\mathrm{CA3}}\}$. On this set the top-$k$ selection is exactly the linear minimization oracle used by Frank & Wolfe (1956).

## E.2 DEGREE-ONE ENERGY: THE ALLOCATION TERM

Take the linear aggregate $\Phi(\mathbf{m}) = \sum_i m_i$. Then $\omega_i = 1$ for every $i$, and $\sum_{i<t} \big(\eta^{(i)} - a\mathbf{1}\big) = u^{(t-1)}$ by Eq. (1). The interaction term therefore reduces to $-\eta^\top u^{(t-1)}$. Interference measured by summed first-order overlaps is the same as the neural-usage term of Eq. (2). This gives a unified reading of the objective: *allocation* is the population-level (degree-one) part of interference and penalizes units that have been used often, whereas the energies below penalize overlap with specific stored patterns.

## E.3 QUADRATIC ENERGY: SPARSE HOPFIELD NETWORK (MAIN TEXT)

The quadratic aggregate $\Phi(\mathbf{m}) = \frac{1}{2}\sum_i m_i^2$ recovers Eq. (4). With $W^{(t-1)} = \sum_{i<t}(\eta^{(i)} - a\mathbf{1})(\eta^{(i)} - a\mathbf{1})^\top$ we have $\frac{1}{2}\eta^\top W^{(t-1)}\eta = \frac{1}{2}\sum_i m_i^2$, so

$$\omega_i = m_i(\eta), \qquad \sum_{i<t} m_i(\eta)\big(\eta^{(i)} - a\mathbf{1}\big) = W^{(t-1)}\eta. \tag{27}$$

Each stored pattern inhibits its units in proportion to its current overlap with the candidate, and the sum is computed in one pass through the recurrent weights. Maximizing $E$ means minimizing the *sum of squared* overlaps with all stored memories. Because every overlap contributes, the new code is pushed away from all stored patterns on average. Many small, chance-level overlaps can together outweigh a single large overlap.

### E.4 Polynomial Energy: Dense Associative Memory

Dense associative memories (Krotov & Hopfield, 2016) replace the quadratic interaction with a higher-order separation function, $\Phi(\mathbf{m}) = \frac{1}{p}\sum_i F_p(m_i)$ with $F_p(m) = m^p$ ($p$ even) or the rectified form $F_p(m) = [m]_+^p$. For the rectified form,

$$\omega_i \;=\; [m_i(\eta)]_+^{p-1}. \tag{28}$$

As $p$ grows, the weights concentrate on the stored patterns that overlap most with the candidate. Patterns at chance-level overlap contribute almost nothing, and memories that share many units with $\eta$ dominate the inhibition. Encoding then avoids the few nearest attractors rather than the average crosstalk. This is the regime in which higher-order energies give super-linear capacity (Krotov & Hopfield, 2016): attractors are sharper, so a new code only needs to clear its nearest neighbours. With the rectified form, overlaps below chance do not reward the candidate, so the new code is never pulled toward *anti*-correlated memories. Unlike the quadratic case, the inhibition can no longer be written as a single matrix–vector product $W\eta$. It needs one term per stored pattern, which corresponds to the higher-order (or hidden memory-unit) interactions of Krotov & Hopfield (2021).

### E.5 Log-Sum-Exp Energy: Modern Hopfield Network

The exponential and modern Hopfield networks (Lucibello & Mézard, 2024; Ramsauer et al., 2021) use

$$\Phi(\mathbf{m}) \;=\; \frac{1}{\lambda}\log\sum_{i<t}\exp\big(\lambda\, m_i\big), \qquad \omega_i \;=\; \mathrm{softmax}_i\big(\lambda\,\mathbf{m}(\eta)\big), \tag{29}$$

where $\lambda > 0$ is an inverse temperature. The recurrent inhibition is a convex combination of stored patterns, weighted by attention over their overlaps with the candidate. The temperature interpolates between the two previous regimes:

- $\lambda \to 0$: $\omega_i \to 1/(t-1)$, so the inhibition is proportional to $u^{(t-1)}$ and the interference term reduces to the degree-one allocation term.
- $\lambda \to \infty$: $\Phi(\mathbf{m}) \to \max_i m_i$, so maximizing $E$ becomes the minimax problem

$$\eta^{(t)} = \arg\min_{\eta}\ \max_{i<t}\ m_i(\eta) \quad \text{s.t. } \mathrm{supp}(\eta) \subseteq \mathcal{C}(\xi),\ \ \|\eta\|_0 = k_{\mathrm{CA3}}, \tag{30}$$

  and at each iteration only the currently closest memory inhibits the drive.

Under this energy, encoding places each new memory as far as possible from its *nearest* stored neighbour, which is the separation criterion that governs basin collisions in high-capacity associative memories.

### E.6 Summary

Table 2: Energy functions and the recurrent-inhibition weights $\omega_i$ they induce in the iterative $k$-WTA drive (Eq. 26). Only the sparse Hopfield energy is evaluated in this paper.

| Energy | $\Phi(\mathbf{m})$ | $\omega_i$ | What encoding avoids |
|---|---|---|---|
| Allocation (degree one) | $\sum_i m_i$ | $1$ | over-used units |
| Sparse Hopfield (Eq. 4) | $\frac{1}{2}\sum_i m_i^2$ | $m_i$ | total squared crosstalk |
| Dense associative memory | $\frac{1}{p}\sum_i [m_i]_+^p$ | $[m_i]_+^{p-1}$ | strongly overlapping memories |
| Modern Hopfield (LSE) | $\frac{1}{\lambda}\log\sum_i e^{\lambda m_i}$ | $\mathrm{softmax}_i(\lambda\mathbf{m})$ | the nearest memory |

All four energies fit the same encoding procedure (Algorithm 1). They differ only in how interference from individual memories is weighted, from uniform (allocation), to proportional to overlap (Hopfield), to concentrated on the closest memories (dense and modern Hopfield). In every case the drive remains a local computation: feedforward input, an adaptive threshold, and feedback inhibition. It is still compatible with the gamma-cycle interpretation in Section 2.4.