THE ART OF REMEMBERING: SCALING LAWS FOR TRANSFORMER ASSOCIATIVE MEMORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory is a fundamental component of intelligent systems. Classical neural memories using Hopfield networks showed that retrieval is best done by energy minimization, where a partial or noisy query is iteratively refined until it settles on a stored memory. However, modern architectures largely ignore energy minimization and instead retrieve stored memories primarily through feed-forward propagation, raising the question of whether energy-based retrieval still matters once neural memories are scaled. To answer this, we compare four retrieval processes: a single forward pass in Feed-Forward Transformers, a diffusion process in Diffusion Transformers, iterative refinement in Recurrent Transformers, and energy minimization in Energy-Based Transformers (EBTs). For each architecture, we train Transformer associative memories and characterize scaling laws and trends across three fundamental properties of neural memories: capacity, generalization to corrupted queries, and continual learning. Across all three properties, we find that EBTs consistently scale best, and that their benefits grow with scale. For example, EBTs have a capacity scaling exponent over 40% higher than other architectures, and store $3.5\times$ more memories than the next-best architecture at scale. Beyond capacity, EBTs generalize better than models $8\times$ larger, and continually learn new memories over $5\times$ faster than other Transformer models. Together, these results show that the classical advantages of energy-based retrieval hold for modern Transformers at scale, offering guidance for designing memory systems in language models, world models, and other architectures.

1 INTRODUCTION

Storing and retrieving memories is a prerequisite for intelligent behavior. In modern neural networks such as Transformers, memory often takes two forms. First, *implicit* memory is commonly embedded in the parameters of models, such as with attention, which retrieves stored patterns by their similarity to a query Ramsauer et al. (2020); Bricken & Pehlevan (2022), and feed-forward layers, which often behave as key-value stores that map input patterns to the facts associated with them Geva et al. (2021); Bietti et al. (2023); Nichani et al. (2024). Second, *explicit* memories are realized through a separate module, such as with external memory banks given to language models and world models Graves et al. (2014); Weston et al. (2015); Dai et al. (2019); Xiao et al. (2026b); Wayne et al. (2018); Parisotto & Salakhutdinov (2017); Behrouz et al. (2024). In either case, stored memories are accessed by queries being compared against what has been stored, with the closest match being returned.

Such forms of memory are only useful if stored values can be recalled from *new* queries at test time that models receive. Because queries at test time are usually incomplete or noisy Willshaw et al. (1969); Kaelbling et al. (1998), robust neural memories need to recover patterns from partial or corrupted evidence. This is the problem of associative memory Kohonen (1972); Anderson (1972), and its classical solution is the Hopfield network Hopfield (1982), which stores each memory as a minimum of an energy function and retrieves by energy minimization LeCun et al. (2006); Du & Mordatch (2019). In Hopfield networks, a corrupted query is treated as a starting point on an energy landscape, and is refined step by step, moving downhill until it settles into the nearest stored memory (Figure 1). As a result, queries only need to start close enough to the right stored memories for descent to reach them, rather than matching stored memories exactly.

However, the dominant recipes for modern Transformers make no use of *explicit* energy minimization, retrieving instead by feed-forward propagation. For example, softmax attention Bahdanau et al.

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.

(2014); Vaswani et al. (2017) compares the query to every stored key once and returns a weighted mix of the matches, which is exactly one step of Hopfield retrieval with no further refinement Ramsauer et al. (2020). Similarly, explicit neural memories are usually read once and passed forward, as in Neural Turing Machines or memory-based agents Graves et al. (2014; 2016); Wayne et al. (2018), without an energy function, leaving no explicit attractor dynamics to pull a corrupted query toward the nearest stored memory.

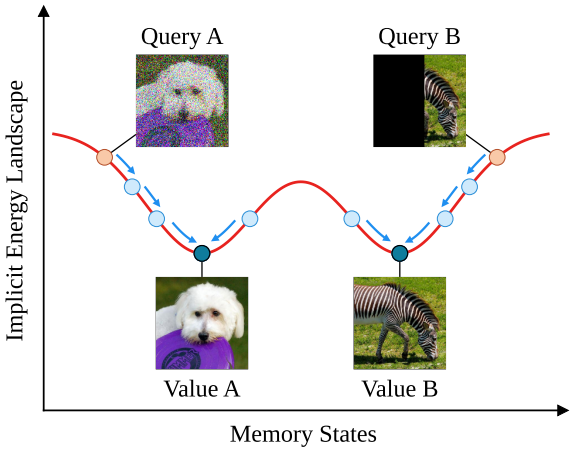


Figure 1: **Attractor Dynamics for Memory Retrieval.** Stored memories (A, B) can be implemented as attractor states at the minima of a retrieval objective. A corrupted query, either noisy (A) or masked (B), is refined step by step, moving downhill until it settles into the nearest stored memory. Energy-based models implement these attractor dynamics directly by minimizing a learned energy, while diffusion and recurrent models follow update rules that only implicitly descend such a landscape.

While more complex retrieval processes have been designed and studied as memories, including diffusion models Sohl-Dickstein et al. (2015); Ho et al. (2020) reinterpreted as associative memories Hoover et al. (2023b); Ambrogioni (2024); Pham et al. (2025), recurrent networks that retrieve by iterating a learned map Radhakrishnan et al. (2020); Jiang & Pehlevan (2020), and energy-based models Bartunov et al. (2019); Hoover et al. (2023a), the effect of the retrieval mechanism on memory performance has rarely been isolated, so it is unclear whether retrieval processes matter once neural memories are learned and scaled. Therefore, to isolate the effect of retrieval mechanism on memory performance, we train Transformer associative memories including Feed-Forward Transformers, which retrieve in a single forward pass, Diffusion Transformers, which follow a learned flow from the corrupted query to the memory, Recurrent Transformers, which recur iteratively to retrieve, and Energy-Based Transformers (EBTs), which retrieve by gradient descent on a learned energy (Figure 2, Section 3). We focus on measuring three key properties of memory where current models are bottlenecked. The first is capacity, the number of memories that can be stored and retrieved, which matters because models store only a few bits of knowledge per parameter and are therefore capped by their size Morris et al. (2025); Allen-Zhu & Li (2025). The second is generalization, the ability to recover memories from queries more corrupted than any seen in training, which is becoming increasingly important as data becomes the limiting resource Frank (2023); Vegesna et al. (2026); Kim et al. (2026). The third is continual learning, which we measure as the speed of writing new memories, and which is important because models struggle to learn new information after training without forgetting what they already know Kirkpatrick et al. (2017); Abbas et al. (2023a); Wang et al. (2024a).

Across all investigated properties, we find that Energy-Based Transformers (EBTs) consistently outscale all other architectures, and that their advantage grows with model size. For example, EBTs have a scaling exponent over 40% higher than other architectures for capacity, and at the largest size tested store $3.5\times$ more memories than the next best architecture. EBTs also generalize better to heavily corrupted queries, recalling more memories than baselines with $8\times$ more parameters, and this advantage grows with scale. In continual learning, the cost of writing new memories grows 47% more slowly for EBTs than for the next-best model, and EBTs reach reliable recall of 1,000 new memories over $5\times$ faster than all other models, while Feed-Forward and Recurrent Transformers fail to converge. Notably, despite the fact that Diffusion and Recurrent Transformers also refine their predictions over several steps, they still fall behind EBTs at scale, demonstrating that *explicit energy minimization* is a key property of scalable neural memory systems. As a result, our results tie modern Transformer memories back to the design of classical Hopfield networks, and suggest that energy-based retrieval could improve language models, world models, and other general neural memories.

Our contributions are as follows:

- To understand how memory retrieval mechanisms affect the performance of neural memories, we compare four types of mechanisms implemented by Feed-Forward, Diffusion, Recurrent, and Energy-Based Transformers.
- For each memory retrieval mechanism, we fit scaling laws and trends for capacity, generalization to corrupted queries, and continual learning as neural memories grow in parameters and training compute.
- We find that retrieval by explicit energy minimization consistently scales better than feed-forward, diffusion, or recurrent memory retrieval, with advantages that grow with model scale.

2 Background

Memory systems must succeed at both writing and retrieving information. Conventional address-based memories assume that these queries specify a precise key or storage location, but queries in the real world are often incomplete or noisy (Hinton & Anderson, 1981; Millidge et al., 2022). Associative memories address this mismatch by retrieving stored patterns from partial or corrupted queries based on how well the queries match stored content, rather than from an exact address (Hopfield, 1982; Millidge et al., 2022; Krotov et al., 2025). This ability to reconstruct a complete stored pattern from incomplete or corrupted evidence without requiring an exact storage location makes associative memories a compelling model for understanding how information can be reliably stored and retrieved (Hopfield, 1982; Krotov & Hopfield, 2021; Ramsauer et al., 2021; Krotov et al., 2025).

Early neural associative memories used explicit storage rules to write memories into a network's weights (Steinbuch, 1961; Kohonen, 1972; Nakano, 1972; Amari, 1972). Hopfield networks used this approach for storage but characterized retrieval as descent on an energy function, where, given a partial or corrupted query, the network minimizes its energy by gradually moving toward a stable state (attractor) corresponding to a similar, complete memory (Figure 1) (Hopfield, 1982; 1984; Cohen & Grossberg, 1983). Classical work evaluated the performance of these networks through *capacity*—how many memories could be stored while still being retrieved accurately (Amit et al., 1985; McEliece et al., 1987; Gardner, 1988)—and showed that capacity and retrieval stability depend strongly on the exact storage rule used to write memories (Hopfield et al., 1983; Personnaz et al., 1985; Krauth & Mezard, 1987; Storkey, 1997). More recent Hopfield variants show that capacity can also be greatly improved by changing the shape of the energy function itself (Krotov, 2023; Krotov et al., 2025).

However, many of these works largely leave open how neural memories behave empirically at scale. The classical results are analytical, or come from simulations with tens to roughly a thousand neurons and weights prescribed by closed-form rules or fit to random patterns rather than learned from real data. More recent work brings these ideas closer to modern practice; for example, deep energy-based memories learn an energy function whose minima are the stored examples and retrieve by descending it from a corrupted query (Bartunov et al., 2021), the Energy Transformer applies the same descent to complete masked inputs from learned rather than explicitly stored memories (Hoover et al., 2023a), predictive-coding memories retrieve by iteratively reducing prediction errors between layers (Salvatori et al., 2021; Yoo & Wood, 2022), and overparameterized autoencoders store training examples as attractors of their own forward map (Radhakrishnan et al., 2020). These studies use modest memory sets (tens to roughly a few thousand patterns) and either compare across model families that differ in storage as well as retrieval or study a single family in isolation, so they ignore isolating retrieval mechanisms directly. Millidge et al. (2022) take a step in this direction by fixing a memory matrix and varying the similarity and separation functions of single-step retrieval, though without learning or iterative refinement. It is therefore unclear whether the advantages of energy-based retrieval persist once neural memories are learned and scaled. We study this in Section 3 by using Transformers and varying retrieval mechanisms.

3 Approach

We study autoassociative retrieval, where a stored memory is recovered from a corrupted version of itself. Given a memory set

$$\mathcal{M} = \{x_i\}_{i=1}^{N},$$

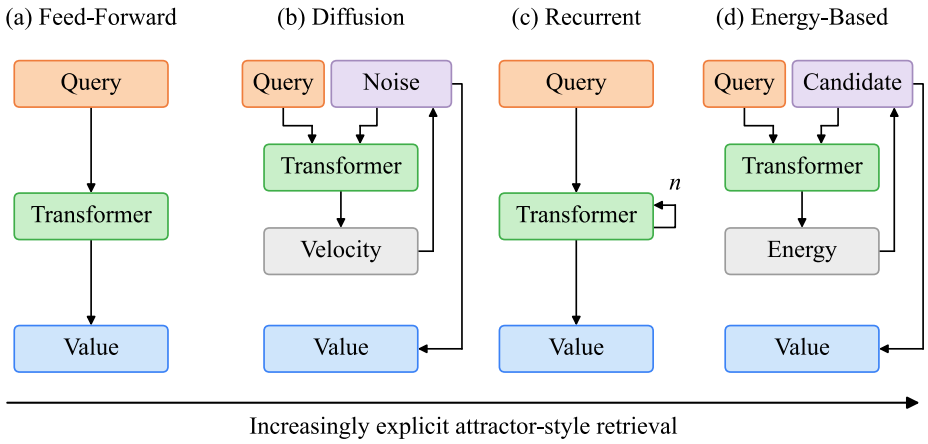


Figure 2: **Transformer Memory Retrieval Variants.** (a) A feed-forward transformer (FFT) predicts a value in a single pass. (b) A diffusion transformer (DiT) predicts a velocity that gradually turns a noisy candidate into a value. (c) A recurrent transformer (RT) iteratively recurs on its own latent state to predict a value. (d) An energy-based transformer (EBT) scores a candidate with an energy and updates it to lower that energy. Methods (b–d) refine their output over multiple steps, with the primary difference from DiTs and RTs being that DiTs have velocity dynamics whereas RTs learn the dynamics.

we learn parameters θ so that a corrupted query $\tilde{x}$ derived from a stored memory $x \in \mathcal{M}$ produces a reconstruction

$$\hat{x} = R_\theta(\tilde{x}) \approx x,$$

where R_θ denotes the retrieval procedure. Each model learns its own parameters, which remain fixed during retrieval.

We compare four transformer-based retrieval procedures: Feed-Forward Transformers (FFTs), Diffusion Transformers (DiTs), Recurrent Transformers (RTs), and Energy-Based Transformers (EBTs). We use similar transformer backbones across methods (Vaswani et al., 2017), with adaptations required by each retrieval mechanism. Experimental details are given in Section 4, with full model configurations in Appendix B.2 and training objective details in Appendix B.3. We also test Predictive Coding Networks but choose not to include them in our main experiments, for reasons listed in Appendix C.

Feed-Forward Transformers (FFTs). The simplest approach is to recover a memory in a single forward pass (Vaswani et al., 2017). Given a corrupted query $\tilde{x}$, a Feed-Forward Transformer directly predicts

$$\hat{x} = F_\theta(\tilde{x}).$$

Retrieval therefore requires no intermediate refinement. FFTs provide the baseline for asking whether associative memory benefits from additional computation during retrieval.

Diffusion Transformers (DiTs). Iterative retrieval offers an alternative to direct reconstruction by allowing the prediction to evolve over multiple steps. Diffusion-style models provide one such mechanism through a time-dependent trajectory, making them a natural way to test whether gradual refinement improves associative retrieval. We therefore adapt Diffusion Transformers (Peebles & Xie, 2023) using a flow-matching formulation (Lipman et al., 2023; Ma et al., 2024).

For a stored memory x and corrupted query $\tilde{x}$, we define intermediate states

$$x_t = (1-t)x + t\tilde{x}, \qquad t \in [0, 1],$$

which connect the memory at $t = 0$ to the corrupted query at $t = 1$. A time-conditioned transformer predicts the velocity along this path:

$$v_\theta(x_t, t) \approx \tilde{x} - x.$$

At retrieval time, the model starts from $\tilde{x}$ and numerically integrates the learned velocity field from $t = 1$ to $t = 0$, returning the final state as the reconstruction. DiTs therefore test whether gradual, time-dependent refinement improves retrieval over direct prediction.

Recurrent Transformers (RTs). Multiple retrieval steps do not necessarily need to follow a time-dependent flow, so we test Recurrent Transformers, which instead reuse the same predictor across

iterations. This follows the parameter-sharing principle used in Universal Transformers (Dehghani et al., 2019), looped transformers (Yang et al., 2024), and recurrent-depth models (Geiping et al., 2025).

Starting from the corrupted query,

$$\hat{x}^{(0)} = \tilde{x}, \qquad \hat{x}^{(k+1)} = F_\theta\left(\hat{x}^{(k)}\right),$$

each step uses the current reconstruction to predict the next one. Because the same parameters are reused across iterations, additional retrieval steps increase computation without increasing parameter count. RTs therefore test whether repeated refinement with a shared update rule and without a hard-coded velocity flow improves associative retrieval.

Energy-Based Transformers (EBTs). Iterative models can also differ in how they compute and perform each update. Rather than directly making predictions, Energy-Based Transformers (Gladstone et al., 2025) learn a scalar energy whose gradient guides refinement, following the broader energy-based modeling framework (LeCun et al., 2006).

Given a candidate reconstruction $\hat{x}$, the model assigns a scalar energy $E_\theta(\hat{x})$, where lower energy indicates a more plausible stored memory. Retrieval initializes the candidate at the corrupted query, $\hat{x}^{(0)} = \tilde{x}$, and repeatedly updates all of its dimensions by gradient descent on the energy,

$$\hat{x}^{(k+1)} = \hat{x}^{(k)} - \alpha \nabla_{\hat{x}^{(k)}} E_\theta\left(\hat{x}^{(k)}\right),$$

where $\alpha > 0$ is the retrieval step size.

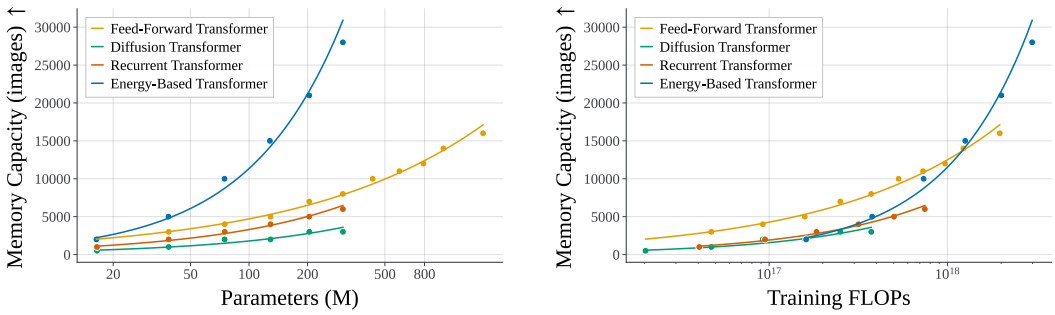
RTs and EBTs therefore both use repeated computation, but differ in how each update is determined. RTs predict the next update directly, whereas EBTs define the update via the gradient of a learned scalar energy, so comparing them isolates whether energy-guided refinement provides benefits beyond repeated computation. We hypothesize that energy minimization should help all three properties we study, since a memory needs to only be a minimum of E_θ rather than a fitted input–output pair (capacity), a corrupted query needs to only start within that minimum's basin for descent to reach it (generalization), and writing a new memory needs to only carve a new minimum without disturbing existing ones (continual learning).

4 Experimentation

We compare four transformer-based retrieval procedures on an autoassociative image-reconstruction task: Feed-Forward Transformers (FFTs), Diffusion Transformers (DiTs), Recurrent Transformers (RTs), and Energy-Based Transformers (EBTs). We first evaluate *capacity*, measuring how many memories each model can store and reliably retrieve as parameter count and training compute increase. We then evaluate *retrieval generalization*, testing whether stored memories remain recoverable from queries with greater corruption than encountered during training. Finally, we evaluate *continual learning*, measuring the optimization steps required to incorporate new memories while preserving reliable retrieval of those already stored.

Across experiments, we use a common image-reconstruction setup. We resize COCO images (Lin et al., 2015) to 128×128 and encode them with a frozen Stable Diffusion KL autoencoder (Rombach et al., 2022), similar to prior transformer-based image models that operate on pretrained latents (Peebles & Xie, 2023; Ma et al., 2024). This compressed representation reduces input dimensionality and makes comparisons across model scales computationally tractable. The resulting 16×16 latent grid is divided into non-overlapping 8×8 patches, giving four tokens per image. Queries mask a subset of latent positions, and the model reconstructs the complete representation.

To measure retrieval, we consider a memory correctly recalled when at least 90% of its scalar latent dimensions have absolute error at most 0.1 relative to the stored representation, a threshold that corresponds well to visually correct reconstructions. *Recall* is the fraction of memories satisfying this criterion, and *capacity* is the largest tested memory set achieving at least 99% recall. To study how this capacity changes with scale, we jointly increase model depth, hidden dimension, and attention heads, analyzing models ranging from 5M parameters to 1.6B parameters. Because some architectures perform repeated computation during retrieval, parameter count alone does not capture their computational cost, so we compare scaling with both parameter count and estimated training FLOPs. Full model configurations, training details, and compute accounting are provided in Appendix B.



(a) Parameter Scaling.

(b) Training-Compute Scaling.

Figure 3: **Memory Capacity Scaling with Parameters and Training Compute.** (a) At the largest matched model size (`Large`), EBT stores $3.5\times$ more memories and exhibits over 40% stronger capacity scaling than the next best models. (b) EBT's stronger scaling allows it to overtake FFT and store 31% more memories at a comparable compute budget. Overall, EBT's memory capacity scales best with both parameter count and training compute.

4.1 Memory Capacity Scaling

How does memory capacity scale with model size and compute? A useful associative memory should convert additional model capacity and training compute into the ability to store more memories. We therefore measure memory capacity across a common range of model sizes and fit its relationship with parameter count using

$$C_{\text{mem}} \propto P^r,$$

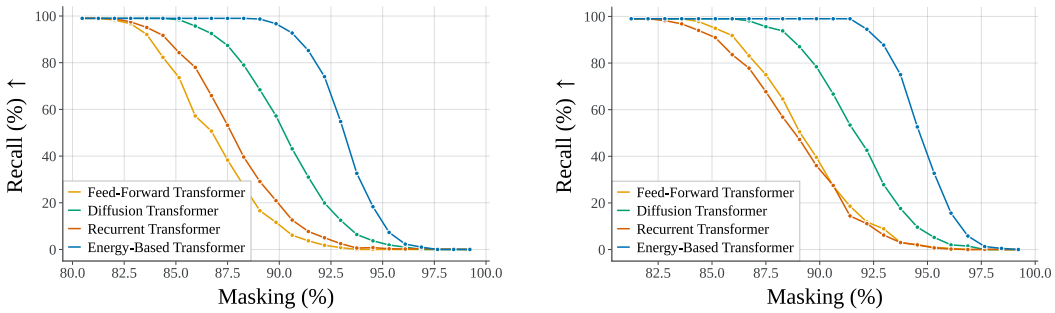
where C_{mem} denotes memory capacity and P the number of parameters. This power-law form provides the best fit to the observed scaling trend, and its exponent r summarizes scaling efficiency: doubling the parameter count increases capacity by a factor of 2^r. Larger values of r therefore indicate faster capacity growth with scale. Model sizes tested, fitted exponents, and capacity sweeps are reported in Appendix B.5.

Using this fitted form, EBT shows the strongest growth as model size increases (Figure 3a). At the largest matched model size (`Large`), EBT stores $3.5\times$ more memories than the next-best architecture, and its fitted capacity scaling is over 40% stronger than that of any other model. This gap also holds at larger scale, where even the `5XL` FFT with more than $5\times$ as many parameters still stores substantially fewer memories than the `Large` EBT. Thus, although capacity grows sublinearly for every architecture, EBT is significantly more efficient as model size increases.

The other iterative architectures help clarify EBT's advantage. DiT achieves the lowest absolute capacity across the tested scales, suggesting that generative objectives designed for denoising and reconstruction may be less suited for precise image memorization. RT provides a more direct comparison because it also performs iterative refinement, yet EBT stores $4.7\times$ more memories at the largest matched scale. This suggests that repeated computation alone is not sufficient to explain EBT's advantage. Unlike RT, EBT retrieves memories by minimizing a learned energy function in a manner similar to Hopfield networks, where retrieval moves states toward stored energy minima (Hopfield, 1982). Thus, EBT's capacity advantage appears to be more strongly associated with energy minimization than with iterative refinement alone.

The comparison changes slightly once the additional computation required by iterative retrieval is taken into account via measuring FLOPs instead of parameters (Figure 3b). FFT is initially more efficient at small scale because retrieval requires only a single forward pass, whereas EBT performs repeated energy-based updates. As compute increases, however, EBT's stronger scaling overcomes this initial cost and reaches 31% higher capacity than FFT at a comparable high-compute budget.

Takeaway: EBT scales best with both parameter count and training compute, storing $3.5\times$ more memories and over 40% stronger capacity scaling than the next-best models. RT scales poorly despite also refining iteratively, suggesting energy minimization drives EBT's advantage.



(a) `2xs` Models (b) `Large` Models

Figure 4: **Retrieval Generalization Across Corruption Levels and Model Sizes.** At corruption levels where EBT retains approximately 75% recall, its recall is 3.7× that of the next-best architecture at `2XS` and 4.2× at `Large`, demonstrating consistent generalization gains. The `2XS` EBT also outperforms `Large` baselines for other architectures, which use approximately 8× as many parameters.

4.2 Retrieval Generalization

How well do architectures generalize to more corrupted queries? Capacity measures the *amount* of information a model can store, but not how *much* of a memory must be observed before that information can be recovered. To isolate retrieval generalization from capacity, we train FFT, DiT, RT, and EBT at several model sizes on the same fixed memory set and corruption level. At retrieval time, we keep the models fixed and progressively reduce the amount of visible information in each query, testing how far each architecture can generalize beyond the corruption levels seen during training. Tested model sizes, masking levels, and evaluation settings are provided in Appendix B.6.

EBT exhibits the strongest retrieval generalization across model sizes, and its advantage grows with scale (Figure 4). At masking percentages where EBT retains approximately 75% recall, it achieves 3.7× the recall of the next-best architecture at `2XS` and 4.2× at `Large`. These comparisons show that EBT's robustness cannot be explained by model size alone, and again, suggest that energy minimization is fundamental to strong memory performance. Interestingly, retrieval generalization also differs substantially from storage capacity, where DiT consistently tolerates stronger corruption than both other transformer architectures despite storing far fewer memories, suggesting that its generative denoising objective supports more robust retrieval. RT is initially more robust than FFT, but improves little with scale, allowing FFT to catch up at larger model sizes. Thus, architectures that store more memories do not necessarily recover them more reliably from incomplete queries, making retrieval generalization distinct from memory capacity. In addition to reporting recall for different levels of generalization, in Figure 6, we visualize retrieved model memories, demonstrating how EBTs consistently retrieve images while other models struggle at high masking ratios.

Takeaway: EBT generalizes best to corrupted queries, achieving 4.2× recall of the next-best model while retaining approximately 75% recall, and outperforms baselines with 8× more parameters. DiT's performance shows that retrieval generalization is distinct from capacity.

4.3 Continual Learning

How quickly can an associative memory learn new information? The previous experiments focused on storing and retrieving an existing set of memories. In practice, however, a memory must also learn new information efficiently without forgetting what it already knows (Kirkpatrick et al., 2017; Abbas et al., 2023b; Wang et al., 2024b). We therefore study a particular form of *continual learning*, measuring how the cost of writing new memories grows with the number of new memories learned.

Starting from checkpoints that reliably recall the same existing memory set, we introduce N new images and continue optimization over the combined old and new memories. We evaluate two forms of convergence. *New-memory convergence* measures the optimization steps required for the new memories to reach reliable recall, while *joint convergence* additionally requires the original memories to maintain the same recall threshold. We characterize how the number of steps required

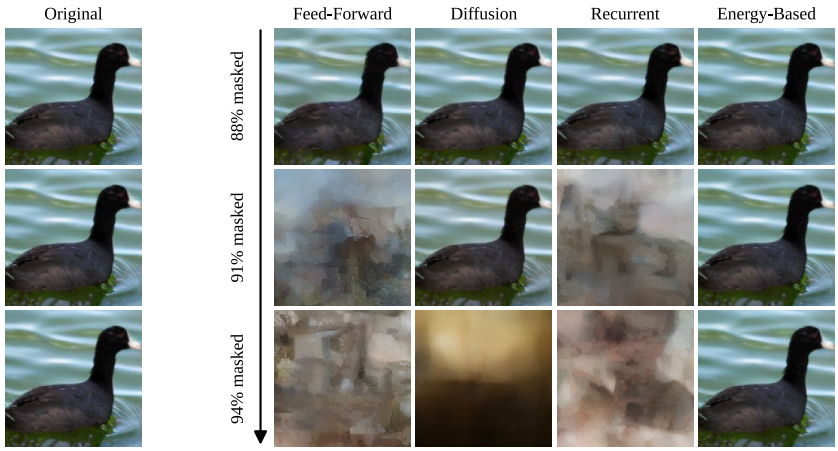


Figure 6: **Denoised Images Across all Four Models and Corruption Levels.** Each model reconstructs the original image from a partially observed query. EBTs consistently retrieve memories better than other architectures, with Diffusion Transformers performing the second best at memory retrieval.

for convergence grows with memory update size by fitting

$$\text{steps} \propto N^b,$$

where N is the number of newly added memories and b is the scaling exponent. We use this functional form because it provides a good empirical fit to the observed relationship between write size and convergence cost. A smaller b indicates that update cost grows more slowly as more memories are introduced. Fitting details and complete convergence results are provided in Appendix B.7.

The continual-learning results reveal large differences in how easily each architecture can incorporate new information while preserving what is already stored (Figure 5). EBT's convergence cost grows much more slowly with update size, with a fitted scaling rate 47% lower than the next-best model. This difference becomes substantial for larger updates: when learning 1,000 new memories, EBT reaches joint convergence in $5.1\times$ fewer optimization steps than DiT, while FFT and RT fail to converge within the training budget. These results suggest that EBT's ability to balance learning new information with retaining existing memories degrades relatively little as more memories are introduced.

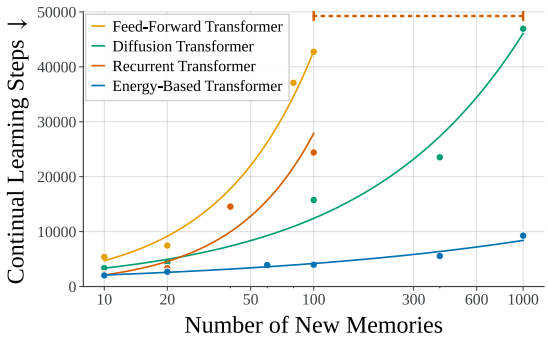
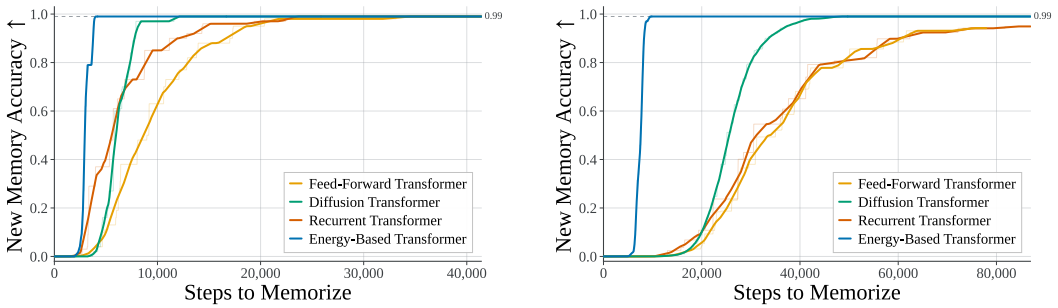


Figure 5: **Continual Learning Required Write Steps** EBTs scale best for learning new memories, with a convergence-cost rate 47% lower than the next-best model. This yields $5.1\times$ fewer optimization steps compared to DiT for learning $1{,}000$ new memories. The dotted interval indicates that RT fails to learn beyond 100 new memories.

The new-memory results help identify where this advantage arises (Figure 7). EBT reaches reliable recall of the new memories $4.0\times$ faster than DiT for the smaller update and $5.1\times$ faster for 1,000 new memories. Moreover, for both EBT and DiT, new-memory and joint convergence occur at nearly the same point: with 1,000 new memories, EBT reaches them at 9,235 and 9,268 steps, while DiT reaches them at 46,930 and 46,933 steps. This suggests that, once the new memories are learned, preserving the original memories requires little additional optimization; the primary difference is how efficiently each architecture acquires the new information. DiT also learns new memories more efficiently than the substantially higher-capacity FFT, showing that greater storage capacity does not necessarily imply a better associative memory.



(a) $N = 100$. (b) $N = 1{,}000$.

Figure 7: **New Memory Recall During Continual Learning.** (a) When learning 100 new memories, EBT reaches reliable recall $4.0\times$ faster than the next-best model. (b) With 1,000 new memories, this advantage grows to over $5\times$, while FFT and RT fail to reach reliable recall. EBT continually learns new memories fastest, with its advantage widening as more memories are introduced.

Takeaway: EBT performs best at continual learning, with convergence cost growing substantially more slowly as new memories are added. Its scaling rate is 47% lower than the next-best model, reaching joint convergence $5.1\times$ faster when learning 1,000 new memories.

5 Future Works and Broader Impact

Our results suggest several directions for extending energy-based retrieval to improve the memory of larger-scale systems. Video world models are a natural application, as maintaining a long context length over hour-long videos is computationally expensive, so an explicit episodic memory that retrieves the full scene through energy minimization could provide persistent memory of an environment Xiao et al. (2026a). In language models, memory currently takes the form of feed-forward layers that store facts as key-value pairs (Geva et al., 2021) and external neural memories that are retrieved by similarity to the query (Behrouz et al., 2024). Both retrieve in a single pass, so replacing them with energy-based memories could improve memory recall and improve continual learning. Testing these ideas will require extending our analysis to heteroassociative memory retrieval, language, longer sequences, and studying the learned energy landscapes directly to understand why energy minimization scales well.

6 Limitations and Conclusion

Limitations. Our experiments focused on autoassociative memory retrieval tasks, where the query and value are part of the same image, so whether our findings extend to heteroassociative retrieval where keys and values are different modalities or to other modalities remains to be tested. While we tuned baselines to maximally improve performance (including for Feed-Forward, Recurrent, and Diffusion Transformers), we did not tune Energy-Based Transformer hyperparameters such as the energy-descent step size or energy landscape regularization, so improved results with better-tuned Energy-Based Transformers are possible. Finally, our continual-learning setting measures write speed under full replay of both old and new memories, so future work can focus on more realistic continual learning setups.

Conclusion. We compared four Transformer retrieval mechanisms, spanning feed-forward prediction, diffusion, recurrent refinement, and energy minimization, and fit scaling laws and trends for capacity, generalization to corrupted queries, and continual learning. Across all three properties, Energy-Based Transformers consistently scale or perform best, and their advantage grows with model size. We hypothesize that this advantage stems from explicit energy minimization shaping stored memories into attractors, whose basins allow heavily corrupted queries to descend to the correct memory. The performances of Diffusion and Recurrent Transformers are consistent with this perspective, as both architectures refine their predictions iteratively without an explicit energy and still scale worse than EBTs. Together, our findings tie modern Transformer memories back to the design of classical Hopfield networks, and suggest that energy-based retrieval could improve memory in language models, world models, and other neural architectures.

AI USE STATEMENT

In this work, we used generative AI tools to help formulate mathematical claims and to assist in the writing of proofs. We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, or interpret results, and translation, dataset cleaning and reformatting, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to write and edit software code, to edit parts of the manuscript to improve readability, and to fact-check claims and references. All AI-generated code was reviewed and tested for correctness by the authors, and all AI-assisted text and fact-checks were verified and revised 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.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure that our results can be reproduced. The retrieval procedures for all four architectures are defined in Section 3, and the common image-reconstruction setup, recall criterion, and capacity definition are described in Section 4. Appendix B provides full experimental details, including data processing and query construction (Appendix B.1), model configurations and hyperparameters (Appendix B.2), training objectives and retrieval dynamics for each architecture (Appendix B.3), and our compute-accounting procedure (Appendix B.4). The raw measurements and fitting procedures underlying our scaling results are reported for memory capacity (Appendix B.5), retrieval generalization (Appendix B.6), and continual learning (Appendix B.7). Finally, we include our full source code for training, retrieval, and evaluation in the supplementary materials.

REFERENCES

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C Machado. Loss of plasticity in continual deep reinforcement learning. In *Conference on lifelong learning agents*, pp. 620–636. PMLR, 2023a. 2

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C. Machado. Loss of plasticity in continual deep reinforcement learning, 2023b. URL https://arxiv.org/abs/2303.07507. 7

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. In *International Conference on Learning Representations*, 2025. arXiv:2404.05405. 2

S.-I. Amari. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers*, C-21(11):1197–1206, 1972. doi: 10.1109/T-C.1972.223477. 3

Luca Ambrogioni. In search of dispersed memories: Generative diffusion models are associative memory networks, 2023. URL https://arxiv.org/abs/2309.17290. 16

Luca Ambrogioni. In search of dispersed memories: Generative diffusion models are associative memory networks. *Entropy*, 26(5):381, 2024. 2

Daniel Amit, Hanoch Gutfreund, and Haim Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55:1530–1533, 09 1985. doi: 10.1103/PhysRevLett.55.1530. 3

James A Anderson. A simple neural network generating an interactive memory. *Mathematical biosciences*, 14 (3-4):197–220, 1972. 1

Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past, 2016. URL https://arxiv.org/abs/1610.06258. 16

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 1

Sergey Bartunov, Jack W Rae, Simon Osindero, and Timothy P Lillicrap. Meta-learning deep energy-based memory models. *arXiv preprint arXiv:1910.02720*, 2019. 2

Sergey Bartunov, Jack W Rae, Simon Osindero, and Timothy P Lillicrap. Meta-learning deep energy-based memory models, 2021. URL https://arxiv.org/abs/1910.02720. 3

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2024. URL https://arxiv.org/abs/2501.00663. 1, 9

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint, 2023. URL https://arxiv.org/abs/2306.00802. 1, 16

Trenton Bricken and Cengiz Pehlevan. Attention approximates sparse distributed memory, 2022. URL https://arxiv.org/abs/2111.05498. 1, 16

Vivien Cabannes, Elvis Dohmatob, and Alberto Bietti. Scaling laws for associative memories, 2024. URL https://arxiv.org/abs/2310.02984. 16

Michael A. Cohen and Stephen Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5): 815–826, 1983. doi: 10.1109/TSMC.1983.6313075. 3

Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. How to compute hessian-vector products? In *ICLR Blogposts 2024*, 2024. URL https://iclr-blogposts.github.io/2024/blog/bench-hvp/. https://iclr-blogposts.github.io/2024/blog/bench-hvp/. 19

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019. URL https://arxiv.org/abs/1901.02860. 1, 16

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2019. URL https://arxiv.org/abs/1807.03819. 5, 16

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in neural information processing systems*, 32, 2019. 1, 16

Michael C Frank. Bridging the data gap between children and large language models. *Trends in Cognitive Sciences*, 27(11):990–992, 2023. 2

Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P. Murphy, and Tim Salimans. Diffusion meets flow matching: Two sides of the same coin. 2024. URL https://diffusionflow.github.io/. 18

E. J. Gardner. The space of interactions in neural network models. *Journal of Physics A*, 21:257–270, 1988. URL https://api.semanticscholar.org/CorpusID:15378089. 3

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025. 5, 16, 17

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021. 1, 9, 16

Alexi Gladstone, Ganesh Nanduru, Md Mofijul Islam, Peixuan Han, Hyeonjeong Ha, Aman Chadha, Yilun Du, Heng Ji, Jundong Li, and Tariq Iqbal. Energy-based transformers are scalable learners and thinkers. *arXiv preprint arXiv:2507.02092*, 2025. 5, 16, 17, 22

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014. URL https://arxiv.org/abs/1410.5401. 1, 2, 16

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016. 2

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. 16

Joshua Hess and Quaid Morris. Associative memory and generative diffusion in the zero-noise limit, 2025. URL https://arxiv.org/abs/2506.05178. 16

Geoffrey E. Hinton and James A. Anderson (eds.). *Parallel Models of Associative Memory*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1981. 3

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 16

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. 16

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed J. Zaki, and Dmitry Krotov. Energy transformer, 2023a. URL https://arxiv.org/abs/2302.07253. 2, 3, 16

Benjamin Hoover, Hendrik Strobelt, Dmitry Krotov, Judy Hoffman, Zsolt Kira, and Duen Horng Chau. Memory in plain sight: Surveying the uncanny resemblances of associative memories and diffusion models. *arXiv preprint arXiv:2309.16750*, 2023b. 2

J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. doi: 10.1073/pnas.79.8.2554. URL https://www.pnas.org/doi/abs/10.1073/pnas.79.8.2554. 1, 3, 6

John Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81:3088–3092, 06 1984. doi: 10.1073/pnas.81.10.3088. 3

John Hopfield, D Feinstein, and R Palmer. 'unlearning' has a stabilizing effect in collective memories. *Nature*, 304:158–9, 07 1983. doi: 10.1038/304158a0. 3

Yibo Jiang and Cengiz Pehlevan. Associative memory in iterated overparameterized sigmoid autoencoders. In *International conference on machine learning*, pp. 4828–4838. PMLR, 2020. 2

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998. 1

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 16

Konwoo Kim, Suhas Kotha, Percy Liang, and Tatsunori Hashimoto. Pre-training under infinite compute. In *International Conference on Learning Representations*, volume 2026, pp. 74596–74636, 2026. 2

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980. 22

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114. 2, 7

Teuvo Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, C-21(4):353–359, 1972. doi: 10.1109/TC.1972.5008975. 1, 3

W Krauth and M Mezard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A: Mathematical and General*, 20(11):L745, aug 1987. doi: 10.1088/0305-4470/20/11/013. URL https://doi.org/10.1088/0305-4470/20/11/013. 3

Dmitry Krotov. A new frontier for hopfield networks. *Nature Reviews Physics*, 5:366–367, 05 2023. doi: 10.1038/s42254-023-00595-y. 3

Dmitry Krotov and John Hopfield. Large associative memory problem in neurobiology and machine learning, 2021. URL https://arxiv.org/abs/2008.06996. 3

Dmitry Krotov, Benjamin Hoover, Parikshit Ram, and Bao Pham. Modern methods in associative memory, 2025. URL https://arxiv.org/abs/2507.06211. 3

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006. 1, 5, 16

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312. 5, 17

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747. 4, 16

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers, 2024. URL https://arxiv.org/abs/2401.08740. 4, 5, 16

R. McEliece, E. Posner, E. Rodemich, and S. Venkatesh. The capacity of the hopfield associative memory. *IEEE Transactions on Information Theory*, 33(4):461–482, 1987. doi: 10.1109/TIT.1987.1057328. 3

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023. URL https://arxiv.org/abs/2202.05262. 16

Beren Millidge, Tommaso Salvatori, Yuhang Song, Thomas Lukasiewicz, and Rafał Bogacz. Universal hopfield networks: A general framework for single-shot associative memory models. *Proceedings of machine learning research*, 162:15561 – 15583, 2022. URL https://api.semanticscholar.org/CorpusID:246680084. 3

John X Morris, Chawin Sitawarin, Chuan Guo, Narine Kokhlikyan, G Edward Suh, Alexander M Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. How much do language models memorize? *arXiv preprint arXiv:2505.24832*, 2025. 2

Kaoru Nakano. Associatron-a model of associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(3):380–388, 1972. doi: 10.1109/TSMC.1972.4309133. 3

Eshaan Nichani, Jason D. Lee, and Alberto Bietti. Understanding factual recall in transformers via associative memories, 2024. URL https://arxiv.org/abs/2412.06538. 1, 16

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning, 2017. URL https://arxiv.org/abs/1702.08360. 1

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. 4, 5, 16

L. Personnaz, Isabelle Guyon, and Gérard Dreyfus. Information storage and retrieval in spin-glass like neural networks. *Journal de Physique Lettres*, 46:359–365, 04 1985. doi: 10.1051/jphyslet:01985004608035900. 3

Bao Pham, Gabriel Raya, Matteo Negri, Mohammed J Zaki, Luca Ambrogioni, and Dmitry Krotov. Memorization to generalization: Emergence of diffusion models from associative memory. *arXiv preprint arXiv:2505.21777*, 2025. 2

Bao Pham, Gabriel Raya, Matteo Negri, Mohammed J. Zaki, Luca Ambrogioni, and Dmitry Krotov. Memorization to generalization: Emergence of diffusion models from associative memory, 2026. URL https://arxiv.org/abs/2505.21777. 16

Adityanarayanan Radhakrishnan, Mikhail Belkin, and Caroline Uhler. Overparameterized neural networks implement associative memory. *Proceedings of the National Academy of Sciences*, 117(44):27162–27170, October 2020. ISSN 1091-6490. doi: 10.1073/pnas.2005013117. URL http://dx.doi.org/10.1073/pnas.2005013117. 2, 3

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020. 1, 2

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need, 2021. URL https://arxiv.org/abs/2008.02217. 3, 16

Rajesh Rao and Dana Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2:79–87, 02 1999. doi: 10.1038/4580. 21

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752. 5, 17, 22

Tommaso Salvatori, Yuhang Song, Yujian Hong, Simon Frieder, Lei Sha, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding, 2021. URL https://arxiv.org/abs/2109.08063. 3, 21, 22

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers, 2021. URL https://arxiv.org/abs/2102.11174. 16

Matthew Smart, Alberto Bietti, and Anirvan M. Sengupta. In-context denoising with one-layer transformers: connections between attention and associative memory retrieval, 2025. URL https://arxiv.org/abs/2502.05164. 16

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015. 2

Karl Steinbuch. Die lernmatrix. *Kybernetik*, 1(1):36–45, 1961. 3

Amos Storkey. Increasing the capacity of a hopfield network without sacrificing functionality. In *International Conference on Artificial Neural Networks*, pp. 451–456. Springer, 1997. 3

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. Augmenting self-attention with persistent memory, 2019. URL https://arxiv.org/abs/1907.01470. 16

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 4, 16

Akshay Vegesna, Samip Dahal, Chinmay Karkar, Bishwas Mandal, Shmuel Berman, and Zhiwei Xu. Slowrun: Language modeling with infinite compute, fixed data, 2026. URL https://github.com/qlabs-eng/slowrun. 2

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024a. 2

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024b. URL https://arxiv.org/abs/2302.00487. 7

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z. Leibo, Adam Santoro, Mevlana Gemici, Malcolm Reynolds, Tim Harley, Josh Abramson, Shakir Mohamed, Danilo Rezende, David Saxton, Adam Cain, Chloe Hillier, David Silver, Koray Kavukcuoglu, Matt Botvinick, Demis Hassabis, and Timothy Lillicrap. Unsupervised predictive memory in a goal-directed agent, 2018. URL https://arxiv.org/abs/1803.10760. 1, 2

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks, 2015. URL https://arxiv.org/abs/1410.3916. 1, 16

James Whittington and Rafal Bogacz. An approximation of the error back-propagation algorithm in a predictive coding network with local hebbian synaptic plasticity, 12 2016. 21

David J Willshaw, O Peter Buneman, and Hugh Christopher Longuet-Higgins. Non-holographic associative memory. *Nature*, 222(5197):960–962, 1969. 1

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers, 2022. URL https://arxiv.org/abs/2203.08913. 16

Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. Worldmem: Long-term consistent world simulation with memory. *Advances in Neural Information Processing Systems*, 38:49632–49652, 2026a. 9

Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. Worldmem: Long-term consistent world simulation with memory, 2026b. URL https://arxiv.org/abs/2504.12369. 1

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms, 2024. URL https://arxiv.org/abs/2311.12424. 5, 16

Jinsoo Yoo and Frank Wood. Bayespcn: A continually learnable predictive coding associative memory. *Advances in Neural Information Processing Systems*, 35:29903–29914, 2022. 3, 21, 22, 23

A Related Work

A.1 Memory and Transformers

One approach to neural memory is to give a model an explicit separate memory store, as in Neural Turing Machines (Graves et al., 2014), Memory Networks (Weston et al., 2015), and retrieval from previously computed representations (Dai et al., 2019; Wu et al., 2022). These designs make memory an explicit component whose contents can be accessed independently of the model itself. Transformers provide a complementary possibility: since they already learn representations, associations, and transformations from data, the model itself may be able to store and retrieve information implicitly without relying on a separate memory module. Prior work supports this view at the level of individual components: attention can implement Hopfield-style associative retrieval (Ramsauer et al., 2021; Bricken & Pehlevan, 2022; Smart et al., 2025), linear attention can act as a fast-weight key–value memory (Ba et al., 2016; Schlag et al., 2021), and feed-forward layers or learned parameters can store factual associations (Geva et al., 2021; Meng et al., 2023; Bietti et al., 2023; Nichani et al., 2024; Sukhbaatar et al., 2019). These results suggest that memory need not be isolated in an external store, but can instead be distributed throughout the transformer's own attention and feed-forward layers.

A.2 Learned Retrieval Mechanisms

If memory is distributed throughout the transformer, then retrieval also is, as the full model must act as the associative memory, recovering a complete stored pattern from a corrupted query. The simplest approach for achieving this is direct prediction, where a standard transformer maps the query to its reconstruction in a single forward pass (He et al., 2021; Vaswani et al., 2017). This provides a natural baseline, but limits retrieval to a single computation. A natural alternative is to let the model refine its prediction over multiple steps.

Flow and diffusion models provide one form of iterative retrieval by learning a time-dependent process that gradually moves a corrupted input toward a target state (Ho et al., 2020; Lipman et al., 2023; Peebles & Xie, 2023; Ma et al., 2024). Recent work has connected these dynamics to associative memory, showing that iterative generative processes can recover or move between memorized samples (Ambrogioni, 2023; Hess & Morris, 2025; Pham et al., 2026). Rather than changing the update rule over time, recurrent transformers reuse the same transformation at every step (Dehghani et al., 2019; Yang et al., 2024; Geiping et al., 2025). This more closely resembles classical Hopfield retrieval, where the same dynamics are repeatedly applied until the state settles into a stored attractor. Both approaches therefore retrieve through gradual refinement, but differ in whether the update varies across steps or is shared throughout retrieval.

Energy-based models make the connection to Hopfield networks more direct. Rather than predicting each update explicitly, they learn a scalar energy function whose gradient determines how the current state should change (LeCun et al., 2006; Du & Mordatch, 2019; Hoover et al., 2023a; Gladstone et al., 2025). Retrieval then proceeds by moving toward lower-energy states, analogous to the energy-minimizing dynamics of Hopfield networks. The key difference is that the energy landscape is learned from data rather than fixed by a prescribed storage rule.

These four procedures—direct prediction, time-dependent refinement, repeated shared refinement, and optimization through a learned energy landscape—have largely been developed under different architectures, objectives, and tasks, making the effect of the retrieval procedure itself difficult to isolate. We therefore implement all four within a common transformer framework (Figure 2) and evaluate them on the same associative-memory tasks.

A.3 Scaling Learned Memories

A comparison at a single model size does not reveal how retrieval mechanisms behave as the memory system grows. Neural scaling studies have shown that performance can change predictably with parameters, data, and compute (Kaplan et al., 2020; Hoffmann et al., 2022). More directly, Cabannes et al. (2024) derive scaling laws for associative memories based on outer products of random, fixed embeddings, showing how memory performance changes with both the number of stored associations and model size. However, It remains unclear whether these trends hold across retrieval mechanisms once the full transformer serves as the associative memory.

Scale is also more than parameter count for iterative retrieval; as flow-based, recurrent, and energy-based models can perform multiple refinement steps using the same parameters, allowing models

of similar size to use different amounts of computation. Recurrent-depth transformers, for example, can improve with additional inference-time iterations (Geiping et al., 2025), while Energy-Based Transformers study scaling across parameters, computation, and inference-time refinement (Gladstone et al., 2025). Parameter count alone may therefore give an incomplete picture of memory efficiency. We compare associative-memory capacity across both model size and estimated training compute, asking whether retrieval mechanisms that store more memories per parameter retain that advantage once their computational cost is taken into account.

Taken together, prior work has studied memory in transformers, learned retrieval mechanisms, and memory scaling largely in separate settings. We bring these questions into a common autoassociative framework by treating the full transformer as the memory system and comparing retrieval procedures across model scales. We additionally test how retrieval behaves under stronger query corruption and as new memories are introduced over time.

B Additional Experimental Details

This section gives the data processing, model configurations, training procedures, and evaluation protocols used in our experiments, along with the values underlying the reported results.

B.1 Data and Query Construction

We use images from the COCO 2014 training split Lin et al. (2015). Each image is resized and center-cropped to 128×128, normalized to $[-1, 1]$, and encoded with a frozen Stable Diffusion KL autoencoder (`sd-vae-ft-ema`) Rombach et al. (2022). We use the encoder's latent mean, scaled by 0.18215, as the fixed reconstruction target $x \in \mathbb{R}^{4\times16\times16}$. No random image augmentation is applied. The Transformer divides this latent grid into four non-overlapping 8×8 patches, producing four tokens per image. Query masking is applied before tokenization over the 256 spatial positions of the latent grid, each containing four channels.

A memory set of size N contains the first N images in a fixed order shared across architectures and model sizes, so larger sets contain all images from smaller sets. In continual learning, new memories are taken from the images immediately following the original $1{,}000$. To construct a query $\tilde{x}$, we retain v uniformly sampled latent positions and zero all four channels at the remaining positions, corresponding to a masking ratio of $1 - v/256$. Masks are resampled each time an image is presented, during both training and evaluation. Memory-capacity and continual-learning experiments use $v = 32$ (87.5% masking) for both training and evaluation, while retrieval-generalization experiments train with $v = 64$ (75% masking) and evaluate with $v \in \{64, 62, \ldots, 2\}$.

B.2 Model Configurations and Hyperparameters

All four architectures use bidirectional Transformer backbones with an FFN expansion factor of 4. Table B.1 lists their depths, hidden dimensions, and trainable parameter counts. FFT and RT have identical parameter counts because RT reuses the same weights across passes, but the two models are trained separately. Each experiment uses a subset of the listed sizes; only FFT is evaluated above Large.

Table B.2 lists the shared settings. Models are trained from scratch for $200{,}000$ steps, except during continual learning, which starts from a trained model and uses the settings in Appendix B.7. We use one training run per experimental configuration.

Table B.3 lists architecture-specific settings, held fixed across sizes and experiments. We tested 2, 3, and 4 recurrent passes for RT and 10, 50, and 100 flow-matching integration steps for DiT. Additional passes reduced RT performance, while additional integration steps did not change DiT performance. We therefore used 2 passes for RT and 10 integration steps for DiT. EBT used its default settings without architecture-specific tuning, leaving it open whether additional tuning could further improve EBT.

B.3 Training Objectives and Evaluation

Let $x \in \mathbb{R}^{4\times16\times16}$ denote the clean image latent and $\tilde{x}$ its masked query. We write $\hat{x}$ for a reconstructed latent and $\hat{x}^{(k)}$ for the intermediate prediction after k recurrent passes or energy-descent updates. Each latent contains $d = 1{,}024$ scalar dimensions, and θ denotes the trainable parameters of

Table B.1: Model configurations and trainable parameter counts, excluding the frozen autoencoder. A dash indicates a configuration not evaluated. Above Large, the FFT / RT column reports FFT only.

				Parameters (M)		
Size	Blocks	Heads	Hidden dim.	EBT	FFT / RT	DiT
4XS	4	4	256	4.9	5.0	–
3XS	6	6	384	16.4	16.5	16.5
2XS	8	8	512	38.5	38.6	38.6
XS	10	10	640	74.8	75.0	74.9
Small	12	12	768	128.9	129.1	128.9
Medium	14	14	896	204.3	204.6	204.2
Large	16	16	1,024	304.6	304.9	304.3
XL	18	18	1,152	–	433.5	–
2XL	20	20	1,280	–	594.1	–
3XL	22	22	1,408	–	790.2	–
4XL	24	24	1,536	–	1,025.3	–
5XL	28	28	1,792	–	1,626.6	–

Table B.2: Shared model and training settings. Continual-learning settings differ in training setup, and are given in Appendix B.7. Loss targets are given in Appendix B.3.

Setting	Value
Attention	Bidirectional
Attention-head dimension	64
FFN expansion factor	4
Latent patch size	8×8
Transformer tokens per image	4
Optimizer	AdamW
AdamW coefficients	$\beta_1 = 0.9$, $\beta_2 = 0.999$
Peak learning rate	10^{-4}
Warmup	Linear, 10,000 steps
Learning-rate decay	Cosine, ending at 10^{-5}
Weight decay	0.01
Gradient clipping	Global norm of 1.0
Batch size	256 images
Training budget	200,000 steps
Loss function	Smooth-ℓ_1, $\beta = 1$

the corresponding model. All architectures use smooth-ℓ_1 loss with $\beta = 1$. For a scalar error e,

$$h(e) = \begin{cases} \frac{1}{2}e^2, & |e| < 1, \\ |e| - \frac{1}{2}, & \text{otherwise.} \end{cases}$$

For a prediction a and target b with d scalar dimensions,

$$\ell(a, b) = \frac{1}{d} \sum_{j=1}^{d} h(a_j - b_j).$$

The training loss is averaged over the batch.

FFT and RT. Let f_θ denote one application of the reconstruction model. FFT predicts the clean latent from the masked query:

$$\mathcal{L}_{\text{FFT}} = \ell(f_\theta(\tilde{x}), x)\,.$$

RT applies the same backbone twice and computes loss only on the second output:

$$\hat{x}^{(1)} = f_\theta(\tilde{x}), \qquad \hat{x}^{(2)} = f_\theta(\hat{x}^{(1)}), \qquad \mathcal{L}_{\text{RT}} = \ell(\hat{x}^{(2)}, x).$$

Gradients pass through both applications of the backbone.

DiT. Training samples $t \sim \mathcal{U}[0, 1]$ and constructs the interpolated latent (we use the Flow interpolation of Diffusion, as they are commonly the same Gao et al. (2024)) as

$$x_t = (1 - t)x + t\tilde{x}.$$

The model $u_\theta(x_t, t)$ predicts the target velocity $\tilde{x} - x$:

$$\mathcal{L}_{\text{DiT}} = \ell(u_\theta(x_t, t),\ \tilde{x} - x)\,.$$

Table B.3: Architecture-specific settings. Training objectives and gradient computation are specified in Appendix B.3.

Architecture	Setting	Value
FFT	Forward passes per retrieval	1
RT	Weight-tied passes per retrieval	2
DiT	Training-time distribution	$t \sim \mathcal{U}[0, 1]$
	Integration method	Euler
	Integration steps	10, uniformly spaced
	Retrieval direction	$t = 1$ to $t = 0$
	Initial state	Masked query
EBT	Energy-descent steps	4
	Step size α	2,500
	Initial candidate	Masked query
	Weight normalization	Spectral norm on attention and MLP projections

Retrieval starts from the masked query at $t = 1$ and integrates to $t = 0$ using ten uniform Euler steps:

$$x_{t-\Delta t} = x_t - \Delta t\, u_\theta(x_t, t), \qquad \Delta t = 0.1.$$

EBT. EBT uses a bias-free linear output head to produce one scalar per token. Their mean defines the energy $E_\theta(\hat{x})$. Starting from $\hat{x}^{(0)} = \tilde{x}$, it applies four updates:

$$\hat{x}^{(k+1)} = \hat{x}^{(k)} - \alpha \nabla_{\hat{x}} E_\theta(\hat{x}^{(k)}), \qquad \alpha = 2{,}500.$$

Training averages reconstruction loss over the four updated candidates:

$$\mathcal{L}_{\text{EBT}} = \frac{1}{4} \sum_{k=1}^{4} \ell(\hat{x}^{(k)}, x).$$

We apply spectral normalization to the attention and MLP projections of EBTs for training stability; in preliminary runs, EBTs without spectral normalization reached similar performance but trained less reliably.

Recall criterion. The final reconstruction $\hat{x}$ is counted as correct when at least 90% of its $1{,}024$ scalar latent dimensions satisfy $|\hat{x}_j - x_j| < 0.1$. Recall is the fraction of queried memories that meet this criterion. Evaluation covers the full memory set, with a freshly sampled mask for every memory at every evaluation. We use 99% recall as the threshold for capacity measurement and convergence.

B.4 Compute Accounting and Resources

We estimate training compute as

$$\text{FLOPs} = \kappa\, 6PD,$$

where P is the trainable parameter count and D counts input tokens before repeated model passes. For capacity experiments,

$$D = 200{,}000 \times 256 \times 4 = 2.048 \times 10^8,$$

corresponding to $200{,}000$ steps, batches of 256 images, and four tokens per image.

The multipliers are $\kappa = 1$ for FFT and DiT, 2 for RT, and 8 for EBT Dagréou et al. (2024). DiT uses one velocity prediction per training example (its integration steps occur only during retrieval), RT applies its backbone twice, and EBT's multiplier approximates four energy evaluations and four candidate-gradient computations. The estimates exclude the frozen autoencoder.

For the compute-scaling figure, we convert each parameter count to estimated FLOPs and apply the same power-law fitting procedure and point exclusions as in Appendix B.5. The reported crossover is obtained by solving for the intersection of the fitted EBT and FFT curves.

B.5 Memory-Capacity Scaling

For each architecture and model size, we train a separate model for every tested memory-set size, with 32 visible latent positions during training and evaluation. We report capacity as the largest tested set reaching 99% recall within the training budget (Table B.4).

Fitting procedure. We use a power law to summarize the scaling trends because it closely follows the observed measurements. The fit $C(P) = AP^r$ uses ordinary least squares in log–log space, with P measured in millions of parameters. Both the capacity plots and their fits exclude 4XS for all

Table B.4: Largest tested memory sets achieving at least 99% recall at 87.5% masking. A dash indicates a configuration not evaluated.

Size	FFT	DiT	RT	EBT
4XS	500	–	500	500
3XS	1,000	500	1,000	2,000
2XS	3,000	1,000	2,000	5,000
XS	4,000	2,000	3,000	10,000
Small	5,000	2,000	4,000	15,000
Medium	7,000	3,000	5,000	21,000
Large	8,000	3,000	6,000	28,000
XL	10,000	–	–	–
2XL	11,000	–	–	–
3XL	12,000	–	–	–
4XL	14,000	–	–	–
5XL	16,000	–	–	–

Table B.5: Capacity-fit coefficients for $C(P) = AP^r$, with P in millions of parameters. R^2 is computed in log–log space.

Model	A	r	R^2	Fitted sizes
FFT	551.02	0.4657	0.9930	2XS–5XL
DiT	99.83	0.6268	0.9404	3XS–Large
RT	202.20	0.6063	0.9864	3XS–Large
EBT	178.92	0.9011	0.9907	3XS–Large

architectures and FFT's 3XS measurement; these measurements remain in Table B.4. Table B.5 gives the coefficients used for the plotted curves.

B.6 Retrieval under Increased Corruption

We evaluate all four architectures at 2XS, Small, and Large. Each model is trained on the same 1,000 memories with 64 of the 256 latent positions visible (75% masking). Every 2,000 training steps, we evaluate at 32 masking levels, $v \in \{64, 62, \ldots, 2\}$, with fresh masks and no parameter updates.

The reliable-retrieval boundary is the smallest tested number of visible positions at which recall reaches at least 99%. Table B.6 reports these boundaries and their corresponding masking percentages. Recall values are capped at 0.99 in the plots.

Figure B.1 provides the Small-model curves omitted from the main-text figure.

B.7 Continual Learning

Selection of continual-learning settings. Preliminary runs showed that all architectures rapidly forgot the original memories when trained only on new ones. We therefore continued training on the combined set of 1,000 original memories and N new memories, measuring how quickly models learn the additions while maintaining recall of the original set. Here, we match parameter counts rather than compute because our focus is how quickly models incorporate new memories into a fixed amount of trainable storage (write speed), rather than the computational cost of doing so. For each architecture, we also compared starting from the first saved checkpoint with 99% recall or the final checkpoint at 200,000 steps, tested learning rates from 10^{-7} to 10^{-1}, and either retained or reset the AdamW moment estimates. We selected the final checkpoint, a learning rate of 10^{-4}, and reset moments for all reported runs.

Protocol. Each experiment starts from a Small model trained for 200,000 steps on 1,000 memories, with at least 99% recall at 87.5% masking. For each write size N, we initialize a separate run from this base model and add the next N images in the fixed dataset order. Training samples batches uniformly from all $1{,}000 + N$ images, including both the original and new memories.

We reset the AdamW moment estimates and use a constant learning rate of 10^{-4} without warmup or decay. Training and evaluation use 32 visible latent positions. Other settings follow Appendix B.2. $N = 50$ is evaluated only for EBT.

Table B.6: Retrieval boundaries at 99% recall. All models train with 64 visible positions. Masking percentages are computed as $100(1 - v/256)$.

	Minimum visible positions			Maximum masking (%)		
Model	2XS	Small	Large	2XS	Small	Large
FFT	48	46	42	81.3	82.0	83.6
DiT	40	36	36	84.4	85.9	85.9
RT	48	44	46	81.3	82.8	82.0
EBT	30	26	22	88.3	89.8	91.4

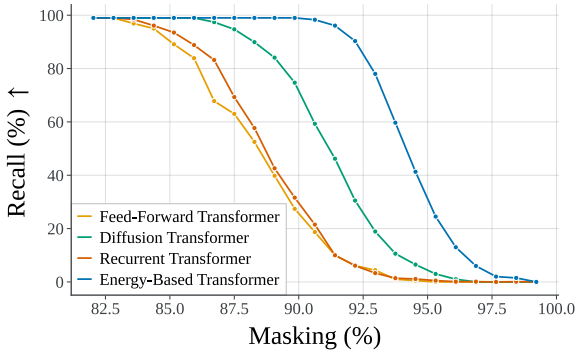


Figure B.1: Recall across masking levels for Small models trained on 1,000 memories with 75% masking.

Evaluation and stopping. Recall is measured separately on old and new memories after every optimization step. New-memory convergence is the first step at which new-memory recall reaches 99%. Joint convergence is the first step at which old- and new-memory recall simultaneously reach 99%. Step counts are measured from the start of continued training. A run stops at joint convergence or after 50,000 additional steps, whichever occurs first.

Fitting procedure. We fit $\text{steps} = AN^b$ using ordinary least squares in log–log space, where N is the number of new memories. Table B.7 lists the fitted coefficients and R^2 for joint convergence. The learning curves and reported speedups at $N = 100$ and $N = 1{,}000$ use direct measurements and do not depend on these fits.

C Generative Predictive Coding Networks

Motivation. The transformer-based associative memories considered in Sec. 4 achieve strong performance across our benchmarking axes. However, they all learn their memories through back-propagation, which raises the question: *Can similar associative memory systems emerge from a different learning mechanism?*

Generative Predictive Coding Networks (GPCNs) provide a way to test this question (Salvatori et al., 2021). Predictive coding was originally motivated by models of neural prediction and error correction (Rao & Ballard, 1999), and GPCNs build on this idea by learning from prediction errors computed locally across a hierarchy of latent representations (Whittington & Bogacz, 2016). Prior work has also studied GPCNs directly as associative memories, where corrupted inputs are iteratively reconstructed by converging toward stored attractor states (Salvatori et al., 2021). Because GPCNs use a fundamentally different learning mechanism from transformers while still supporting associative recall, they provide a useful baseline for this comparison.

Predictive coding as associative memory. A GPCN represents an input with a hierarchy of latent states $x^{0:L}$, where x^0 is the input. Each state x^{l+1} predicts the preceding state x^l through weights W^l and a nonlinearity f, producing a local prediction error. The network energy sums these errors across the hierarchy (Yoo & Wood, 2022):

$$E(x^{0:L}; W^{0:L}) = \frac{1}{2} \left\| x^L - W^L \right\|_2^2 + \frac{1}{2} \sum_{l=0}^{L-1} \left\| x^l - f(x^{l+1}) W^l \right\|_2^2 .$$

Table B.7: Joint-convergence fits for steps $= AN^b$ using selected observations. R^2 is computed in log–log space.

Model	A	b	R^2
FFT	518.6	0.9584	0.9812
DiT	898.7	0.5700	0.9813
RT	157.8	1.1237	0.9325
EBT	1,045	0.3017	0.9708

The first term captures the top-level error, while the remaining terms measure prediction errors between adjacent states. During the *write* phase, the weights are updated so that stored examples become low-energy configurations. Alternatively, during *read*, the weights remain fixed while the latent states are optimized to reduce the energy. Starting from a corrupted input, this optimization can converge toward a stored attractor state (Salvatori et al., 2021; Yoo & Wood, 2022).

This retrieval procedure seems similar to the Energy-Based Transformer (EBT), which also performs associative recall through iterative energy minimization. However, GPCN energy is explicitly defined by local prediction errors between neighboring latent states, whereas the EBT learns a scalar energy over token representations and minimizes it using iterative gradient descent (Gladstone et al., 2025).

Setup. To implement the GPCN, we use the maximum-likelihood (`ml`) configuration of the BayesPCN codebase (Yoo & Wood, 2022), which follows the original GPCN learning rule rather than the Bayesian weight update introduced by BayesPCN. We also use a single network instead of a particle ensemble and set $\beta = 0$ to disable forgetting. [1]

During the *write* phase, memories are processed in minibatches through multiple passes over the stored set, following the offline GPCN setting, which performs substantially better than the one-shot online setting in BayesPCN Yoo & Wood (2022). Then, during *read*, the weights remain fixed while the latent states are optimized from the masked query using Adam Yoo & Wood (2022); Kingma & Ba (2017).

However, a key difference arises during the *read* phase. Prior GPCN work on image reconstruction keeps the initially unmasked portions of the query fixed while inferring the missing patches (Salvatori et al., 2021; Yoo & Wood, 2022), whereas our transformer evaluation allows all patches to be updated. We therefore evaluate two retrieval settings.

In the *free* setting, all latent dimensions can be optimized during inference, matching our transformer evaluation. In contrast, the *clamped* setting keeps the observed dimensions fixed at their ground-truth values throughout iterative refinement, while only the masked dimensions are optimized. This makes retrieval easier because the observed dimensions remain exact throughout inference, providing an anchor for reconstruction. We therefore use the free setting for a direct comparison with transformers and report the clamped setting as a more favorable GPCN evaluation, with the main results in this section using clamped retrieval.

Aside from these GPCN-specific choices, we match the transformer experiments as closely as possible. Each 128×128 image is encoded with the same frozen Stable Diffusion KL autoencoder into a 1024-dimensional latent representation (Rombach et al., 2022), and we use the same memorize-then-reconstruct task. An image is deemed correctly reconstructed when at least 90% of latent dimensions fall within an L_1 distance of 0.1 from the ground-truth latent.

Results. We first evaluate the *free* read, matching the transformer retrieval setting. Under this protocol, the GPCN does not reliably store even 30 images. Because the observed portion of the query is not fixed, optimization leads the unmasked patches to move away from their original values and converge to an incorrect low-energy state.

Next, we evaluate the *clamped* read, which keeps the observed dimensions fixed throughout inference. Clamped retrieval substantially improves performance, achieving perfect recall through 100 stored images. However, **recall drops to 0.048 at 125 images** (Table C.2). Thus, even with clamped retrieval, the GPCN's performance is significantly worse than our transformer models.

[1] BayesPCN can gradually weaken older memories to free capacity for new ones, at a rate controlled by β. Setting $\beta = 0$ disables this mechanism.

We next tested whether additional depth could improve this performance. Increasing GPCN depth from four to eight layers did not improve retrieval, so we report results using the eight-layer model. This is consistent with the BayesPCN ablation, which found limited gains from additional depth but improved performance with increased width (Yoo & Wood, 2022). Our eight-layer GPCN has width 1024 and approximately 7.3M parameters, about $1.5\times$ as many as our smallest `4XS` transformer, despite achieving substantially lower memory capacity.

Beyond its limited capacity, GPCN training and retrieval are also computationally expensive in our experiments. A single run completed approximately 1.7k training steps over 17 hours, while each validation pass used $10{,}000$ inference iterations per query image and took approximately 7–10 minutes.

Therefore, the GPCN evaluated here does not exhibit the same memory scaling as our transformer models. Free retrieval fails at small memory sizes, while the more favorable clamped setting still underperforms our smallest transformer despite using more parameters. Retrieval is also substantially more computationally expensive.

Thus, we exclude GPCNs from the larger-scale experiments. These results are specific to the GPCN formulation and retrieval procedure evaluated here, and should not be interpreted as a limitation of predictive coding or non-backpropagation learning more broadly.

Hyperparameter	Value
Total layers	8
Layer width	1024
Activation	ReLU
Particles	1
Update	`ml` (maximum likelihood, 1 weight step per write iter.)
Weight LR η	10^{-4}
σ_W / σ_x	1.0 / 0.01
Activation steps (write / read)	100 / 10,000
Activation LR	10^{-3} (Adam)
Forget β	0

Table C.1: GPCN hyperparameters.

Stored images	75	100	125
Recall ↑	1.00	1.00	0.048
MSE ↓	0.0003	0.008	0.0124

Table C.2: Clamped recall vs. number of stored images. MSE is in VAE-latent space.