MODELING HUMAN MEMORY WITH TASK-TRAINED NEURAL NETWORKS

Anonymous authors

Paper under double-blind review

ABSTRACT

Human memory is prone to multiple well-documented forgetting and confusion phenomena. We ask whether these effects also emerge from training general-purpose memory architectures, and what role architecture, training tasks and input statistics each play. We introduce a framework spanning classical cognitive models and modern machine learning architectures, built around a recurrent working memory and an associative episodic memory. Six representative models are trained on sequences with tunable input statistics (semantic structure, temporal structure, partial observability), using recognition, reconstruction and sequential recall tasks. They are tested on standard paradigms in human memory research to assess the qualitative emergence of nine phenomena. We find that some effects always emerge regardless of architecture or training regime (perceptual similarity, conjunction errors, fan effect, set size); semantic false memories emerge specifically from input statistics; and other phenomena emerge from either architecture, training tasks, or input statistics (contiguity, primacy, recency, repetition). Notably, the presence of these phenomena does not appear to depend strongly on performance: most of them persist even when accuracy is close to 1. Our results suggest that some human memory phenomena may be understood as adaptations to environmental constraints, and we provide concrete candidates for what such constraints may look like in a simplified setting. More broadly, our work shows how task-trained models can complement classical cognitive models, as tools to capture many phenomena at once and explore the conditions for their emergence.

1 INTRODUCTION

The human brain is the product of millions of years of evolution, further shaped over each individual's lifetime by learning from experience. One outcome of this process is a memory system exhibiting a rich and reproducible set of phenomena: items at the beginning and end of a list are remembered better than those in the middle (primacy and recency) (Jones & Oberauer, 2013; Kahana et al., 2022), presentation of one item tends to evoke neighboring items (contiguity) (Kahana & Caplan, 2002; Davis et al., 2008), related but never-presented items are falsely recognized (Roediger & McDermott, 1995; Webb et al., 2016), memory for repeated material improves with exposure (Jang & Lee, 2019), among many others (Schacter, 2021; Kahana et al., 2022).

Explaining these phenomena was historically done by constructing dedicated computational models, each built to capture a specific effect or a small family of related effects. Classical examples include the Temporal Context Model (TCM) Howard & Kahana (2002) and its generalizations, the Context Maintenance and Retrieval model (CMR) Polyn et al. (2009) and CMR-replay (Zhou et al., 2024), which use a slowly drifting contextual representation associated to studied items in order to account for recency and contiguity effects in free recall, among other effects. These models are typically built by hand, with a small number of free parameters, and are fit to reproduce specific experimental data. This approach has produced precise, interpretable accounts of individual phenomena, but each model is generally restricted to the task and stimulus format for which it was designed, and combining accounts of multiple phenomena into a single model remains difficult.

More recently, a number of studies have used flexible models, taken from or inspired by machine learning, to study human memory: rather than hand-designing a mechanism, a flexible model is

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.

trained on a task and then probed for the phenomenon of interest. Lu et al. (2022) train an LSTM with external memory to predict observations in a causally structured environment and relate its encoding/retrieval strategies to the human brain. Li et al. (2026) train a GRU with external memory directly on free recall of discrete items, and show that several distinct recall strategies emerge over training, some of which resemble those used by human participants. Salvatore & Zhang (2025) likewise train a memory-augmented RNN on free recall, and compare its behavior and mechanisms to the classical CMR model. Finally, Song et al. (2025) train a GRU with key-value memory to predict upcoming scenes in a television episode, comparing its behavior and activity to human data.

In this work, we build on this task-trained modeling approach, aiming not at a single model but at a general framework for constructing and comparing them. A trained model's behavior is jointly determined by three main choices: its *architecture*, the *task* it is trained to perform, and the *input distribution* it is trained on. We propose a framework built around this decomposition. First, the model architecture. Different studies propose different models, sometimes with different notations, so that each proposal looks like an isolated contribution, and it is unclear whether conclusions drawn from one architecture would generalize to another. We propose a unified model space that expresses both classical cognitive models and modern machine-learning architectures in a common notation, built around a recurrent working memory and an associative key-value episodic memory. Second, the training tasks. It may seem natural to train a model on a single task of interest, such as recognition or cued recall. But the brain was not shaped to solve any one task in isolation. It was optimized to support a much wider range of behaviors, potentially biasing which strategies and representations emerge. We propose a modular training setup in which multiple tasks, such as recognizing previously encountered stimuli, reconstructing them from partial cues, and retrieving their temporal neighbors, can be combined during training, so that a model's representations are shaped by more than one memory demand. Third, the input distribution. Our brain has evolved in a richly structured environment, far from the sequences of independent inputs typically used in experiments. We propose an input format that allows precise control over input statistics, including properties like semantic structure, temporal structure, and partial observability. The same input format also lets us simulate standard experimental paradigms, such as cued recall or recognition, so that trained models can be probed for phenomena of interest in essentially the same way as human participants.

Using this framework, we studied the emergence of a set of nine well-documented human memory phenomena. We found that the answer to which factor explains a given phenomenon is not uniform across phenomena: some effects emerged across almost all models and conditions we tested (perceptual similarity, conjunction errors, fan effect, set size); others emerged only from the structure of the environment, regardless of the model architecture (semantic false memories); and others were produced by specific architectural constraints or environment structure (contiguity, primacy, recency, repetition). In particular, our results showed how some phenomena classically explained with specifically tuned mechanisms can instead be understood in terms of the environment and training procedure, with no particular model architecture necessary to explain them. This points toward using task-trained models as a complementary approach to classical cognitive models, to capture many phenomena simultaneously and uncover the conditions leading to their emergence.

2 METHODS

2.1 FRAMEWORK

A memory model receives a sequence of study observations, followed by a recall cue, and produces an output to solve the task specified by the cue (Figure 1a). Three aspects of this setup can be varied independently to shape the behavior of the memory system: the architecture of the memory model (Section 2.2), the statistics of the input sequences (Section 2.3), and the training tasks that define what the model is trained to perform (Section 2.4). We evaluated trained models by testing whether they exhibit memory phenomena observed in humans. By varying these three components, we explored the conditions under which each phenomenon emerges.

2.2 UNIFIED SPACE OF MEMORY MODELS

To study the role of the model architecture in the emergence of memory phenomena, we considered varied memory models, drawing inspiration from both the cognitive psychology and machine learn-

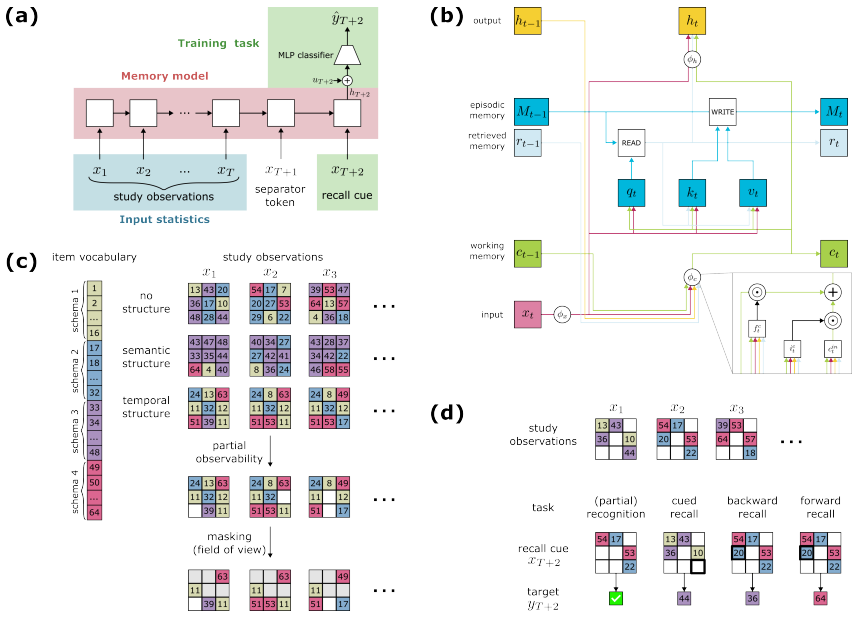


Figure 1: Overview of the framework. **(a) Training setup.** Input sequences consist of a study list followed by a fixed separator and a recall cue. Each input is embedded and processed by a recurrent memory model. During recall, the memory output h_{T+2} is combined with a task indicator u_{T+2} and fed to an MLP classifier that produces an output $\hat{y}_{T+2}$. All components are trained end-to-end. The main configurable components are the memory model, the input statistics, and the training tasks. **(b) Memory model architecture.** The observation x_t is embedded and integrated into a vector memory state c_t. Both x_t and c_t are used to compute keys and values written to a matrix memory M_t, as well as a query used to retrieve a vector r_t. The final memory output h_t is computed from x_t, c_t, and r_t. Arrows show typical dependencies between variables, $\odot$ denotes element-wise multiplication. **(c) Input format.** Each observation consists of a set of K co-occurring items drawn from a finite vocabulary. We considered four input regimes: (i) no structure, where items are sampled uniformly; (ii) schematic structure, where the vocabulary is partitioned into schemas and sampling is biased toward a dominant schema per sequence; (iii) temporal structure, where consecutive observations share a fraction of items; and (iv) temporal structure and partial observability, where some parts of the input sequence might be unobservable and must be reconstructed from neighboring views. Finally, a global random mask is applied, removing a subset of the K items consistently across all timesteps; these masked positions are irrelevant for this sequence. **(d) Training tasks.** Following the study sequence, a recall cue is provided as input along with a task indicator representing one of four possible tasks: recognition (identify whether a cue appeared in the study sequence), cued recall (recover the item at a specified position from a partial cue), backward recall (recover the item preceding a given cue), and forward recall (recover the item following a cue). Models can be trained on any subset of these tasks.

ing literature. We defined our model space as follows (Figure 1b). The main components are: (i) the **input** x_t: the observation at time t, of which an embedding x'_t is generated; (ii) the **working memory** c_t: a recurrent state that accumulates information over time, which can represent the cell state of an LSTM or the context representation in TCM (Howard & Kahana, 2002) or EGO (Giallanza et al., 2024); (iii) the **episodic memory** M_t: an associative memory storing associations between keys k_t and values v_t, which can be queried with a vector q_t to produce a retrieved vector r_t (write and read operations are described below); and (iv) the **memory output** h_t. Key-value memory stores are typically used in Transformers (Vaswani et al., 2023) and other machine learning models (Sukhbaatar et al., 2015; Miller et al., 2016; Pritzel et al., 2017; Ritter et al., 2018; Beck et al., 2024), and have also been related to hippocampal memory (Gershman et al., 2025; Whittington et al., 2022; Zheng et al., 2025).

These variables interact with each other with the following equations, where f_t^c is a forget gate, i_t^c is an input gate, and c_t^{in} is a candidate working memory. Input and forget gates can be scalars

between 0 and 1, or vectors applied element-wise. The functions ϕ may be fixed or learned and use any subset of their listed inputs:

$$
\begin{aligned}
c_t &= f_t^c \odot c_{t-1} + i_t^c \odot c_t^{in} & q_t &= \phi_q(x'_t, c_t) & r_t &= READ(M_{t-1}, q_t) \\
c_t^{in} &= \phi_{c^{in}}(c_{t-1}, x'_t, h_{t-1}, r_{t-1}) & k_t &= \phi_k(x'_t, c_t, r_t) & M_t &= WRITE(M_{t-1}, k_t, v_t) \\
f_t^c &= \phi_{f^c}(c_{t-1}, x'_t, h_{t-1}, r_{t-1}) & v_t &= \phi_v(x'_t, c_t, r_t) & h_t &= \phi_h(x'_t, c_t, r_t) \\
i_t^c &= \phi_{i^c}(c_{t-1}, x'_t, h_{t-1}, r_{t-1})
\end{aligned}
$$

The WRITE and READ operation are described below. For more details, please refer to Section A.2.

WRITE operation. The WRITE operation is the means by which an association between a key k_t and a value v_t is stored in the episodic memory M_t. Typically, the write operation can be expressed as simply appending k_t to a matrix of keys K, and v_t to a matrix of values V (equations in Section A.2). Depending on the READ operation used, storing these growing matrices explicitly is not necessary and a more biologically plausible memory representation for M_t can be used instead, e.g. $M_t \in \mathbb{R}^{D \times D}$ accumulates outer products of keys and values in Howard & Kahana (2002); Polyn et al. (2009); Katharopoulos et al. (2020); Beck et al. (2024).

READ operation. The READ operation is the means by which the model can access previously stored information in M_t using a query q_t. Typically, the READ operation computes the similarity between q_t and previously stored keys in K_{t-1}, then retrieves a weighted sum of stored values based on these similarities, i.e. $r_t = V_{t-1}^\top w_t$ with $w_{t,i} = \theta(q_t, k_i)$ (or normalized as $w_{t,i} = \frac{\theta(q_t,k_i)}{\sum_j \theta(q_t,k_j)}$) where θ is a function to compute the similarity between a query and a stored key.

Many existing models fit in this space. Together, these components make the model space sufficiently rich to capture many existing models. For instance, a Modern Hopfield Network (Ramsauer et al., 2020) can be expressed by dropping the working memory c_t and keeping only the episodic memory M_t. A Transformer (Vaswani et al., 2023) follows from the same construction, with q_t, k_t and v_t given by learned linear projections of the input embedding. Conversely, an LSTM would be expressed by dropping the episodic memory component M_t and keeping only the working memory c_t. TCM, on the other hand, uses both memory components with most of the ϕ transformations fixed, and a small number of free parameters. We report a more detailed list of existing models in Table S6 with the corresponding components expressed in this notation.

Models considered in this work. Models used in our experiments are summarized in Table 1. The first is a flexible model (Flexible), in which all transformations are learnable. From this we derived two ablations: Flexible-M, which removes the working memory component, and Flexible-c, which removes the episodic memory component. We additionally considered a constrained model (Constrained), where only the embedding and classifier are learned and all other transformations are fixed. For this constrained model, we further varied which variable (x'_t or c_t) is used as the keys and values in the READ/WRITE operations. Additionally, we considered versions of our models, namely Flexible-distract and Constrained-distract, in which the working memory c_t is replaced by random noise between the study and recall phases, simulating a distractor task (Lu et al., 2022; Li et al., 2026).

2.3 Input format

Typical experimental paradigms use sequences of independent, unrelated inputs: list of words, pairs or images. This is useful for testing and isolating specific phenomena, but it does not reflect the environment the brain evolved in. The natural environment is not a random stream of independent, fully observed items: semantically related objects tend to co-occur, successive observations tend to change smoothly, and observers may not have access to all aspects of a scene at every timestep. We hypothesize that a memory system trained under these regularities will learn to exploit or adapt to them, and that several memory phenomena may be a direct consequence of this. To study this, we designed an input format that is simple enough to manipulate precisely, yet flexible enough to capture aspects of these natural regularities and to emulate standard experimental paradigms.

Formally, models observe a sequence of T study observations. Each observation x_t consists of a set of co-occurring items drawn from a fixed vocabulary. With this format, we can shape the input distribution by adding different types of structure and regularities, each of which can be independently

Table 1: Models considered in this work, with the components that define their state update, write, and read operations. For all models, the READ operation is performed as $r_t = V_t w_t$ with $w_t = \mathrm{softmax}(K_t q_t)$, and the output is $h_t = l(x'_t) + l(c_t) + l(r_t)$, where the notation $l(x)$ indicates a learned linear map with its own parameters, e.g. $l(x'_t) = W_{x \to h} x'_t$. Absent components are indicated by "−".

Model	ϕ_c		**WRITE**		**READ**
	f_t^c, i_t^c	c_t^{in}	k_t	v_t	q_t
Flexible	$\sigma(l(x'_t) + l(c_{t-1}))$	$\tanh(l(x'_t) + l(c_{t-1}))$	$l(x'_t) + l(c_t)$	$l(x'_t) + l(c_t)$	$l(x'_t) + l(c_t)$
Flexible-M	−	−	$l(x'_t) + l(c_t)$	$l(x'_t) + l(c_t)$	$l(x'_t) + l(c_t)$
Flexible-c	$\sigma(l(x'_t) + l(c_{t-1}))$	$\tanh(l(x'_t) + l(c_{t-1}))$	−	−	−
Constrained	0.5	$\tanh(x'_t)$	c_t	c_t	c_t
Constrained-cxc	0.5	$\tanh(x'_t)$	c_t	x'_t	c_t
Constrained-xcc	0.5	$\tanh(x'_t)$	x'_t	c_t	c_t

enabled or disabled (Figure 1c): **semantic structure** partitions the vocabulary into groups of related items that are more likely to co-occur within a sequence; **temporal structure** introduces consistency across timesteps, such that neighboring observations are more likely to share items; **partial observability** (PO) randomly hides parts of an observation, which must then be reconstructed from neighboring observations. Additionally, we use a variable **field of view**, which determines the number of items visible at each timestep for each sequence, replacing the remaining items with mask tokens. This let us simulate classical experimental paradigms: masking all but one item simulates a list of single items (e.g., lists of words), masking all but two items simulates paired associates, and leaving more items visible models richer inputs such as images or scenes.

We considered four input regimes for training: no structure, semantic structure only, temporal structure only, and temporal structure with partial observability. In all cases, the field of view was randomly sampled for each sequence during training, so that models were exposed to all scenarios. For more details on the implementation, please refer to Section A.1.

2.4 Training and testing

Models observed sequences of study observations (Section 2.3), followed by recall cues which determine the tasks that should be performed. During training, we sampled recall cues from four possible tasks representing plausible memory functions: **recognition** requires deciding, given a (partial) cue, whether a matching observation has been seen before; **cued recall** requires recovering the missing part of a partial cue; and **backward/forward recall** requires retrieving the observation that preceded or followed a given cue in the study sequence. Each task can be switched on or off independently for training. For details about the tasks implementation, please refer to Section A.3.

We trained models for 500 epochs of 5000 sequences each, saving their state after each epoch. Unless stated otherwise, phenomena were evaluated at the epoch whose accuracy most closely matched human accuracy on paired-associates recognition (for phenomena assessed with the recognition task, see Table S4) or cued recall (for phenomena assessed with cued recall, see Table S5). Models whose accuracy never came within 0.1 of the human target were excluded from the main results. For testing, we removed any training-time structure and masking was fixed to match the target paradigm (single items, paired associates, or full scenes). The phenomena we tested for are summarized in Figure 2a. Please refer to Section A.4 for details about the training and testing procedures.

3 Results

We trained the models in Table 1 on the tasks described in Section 2.4, using the input regimes described in Section 2.3, and tested each trained model for nine memory phenomena (Figure 2a). We first summarize our main results and takeaways. We then examine three exemplar phenomena, chosen because they span a range of explanations for how a phenomenon emerges.

Some phenomena are universal, some depend on input statistics. We found that perceptual similarity effects, conjunction errors, the fan effect, and the set-size effect appear in every scenario we consider (Figure 2b). Some phenomena, by contrast, were more sensitive to input distribution. Semantic false memories depended on it entirely: whenever semantic structure was present in the input sequence, trained models showed the effect, irrespective of architecture (Figure 2c).

Architecture determines what is possible; training determines what is realized. A model's architecture can be tuned so that phenomena of interest are hardcoded into it, guaranteeing their appearance under any training regime. The Constrained model illustrates this case: because most of its mechanisms are fixed and only a few components are learned, it exhibited all phenomena in all scenarios (Figure 2e). Conversely, an architecture can be built such that certain phenomena are structurally excluded, whatever training is used. The Flexible-M model is an example of this: it processes and integrates inputs into episodic memory independently of one another (no working memory), so that temporal effects such as contiguity, primacy, and recency cannot arise (Figure 2d). Between these two extremes, an architecture can instead be compatible with all phenomena while remaining flexible enough that none is enforced by construction, leaving their emergence dependent on the fitting procedure. The Flexible model illustrates this case: by default, when trained on sequences of independent inputs (no input structure), only universal phenomena appeared; but when exposed to partially observable and temporally structured sequences, the trained model's behavior changed and additional phenomena emerged: contiguity, primacy, recency, and repetition (Figure 2g).

Results are stable across training. Focusing on the Flexible model, we found that effect sizes vary smoothly over training and that most phenomena remained present even once accuracy was close to 1 (Figure 2h). Our choice, in Figure 2(b-f), of a single training epoch matched to human-level accuracy (see Section 2.4) was therefore not critical: any epoch within a reasonable range around this target would yield comparable results.

3.1 Perceptual similarity effects are omnipresent

In humans, the probability of falsely recognizing an observation increases when perceptually similar observations have been studied. In our models, this effect appeared in every scenario we consider (Figure 2b). More precisely, the probability of answering OLD to a new recognition cue increased monotonically with the fraction of matching content between the cue and previously studied views (Figure 3a). The same structure appeared at the representational level: embedding similarity increased with matching content even in untrained, randomly initialized models (Figure 3b).

No need for an explicit mechanism. Classical models explain similarity effects via explicit feature similarity computation between a cue and memory traces (e.g. MINERVA2 (Hintzman, 1984), GCM (Nosofsky, 1986)). However, a variant with no key-value memory at all (Flexible-c in Figure 3a), and thus no explicit similarity computation, still showed the effect. This suggests that no dedicated similarity mechanism is required: the embedding similarity already present at initialization (Figure 3b), combined with a smooth readout of the memory state, which neural networks are biased toward learning (Rahaman et al., 2019), may suffice to produce it.

3.2 Semantic false memories are due to input structure only

False memories created from semantically similar items have been widely studied in the human memory literature (Roediger & McDermott, 1995; Gallo, 2010; Coane et al., 2021; Chang & Brainerd, 2021; Gatti et al., 2022; Webb et al., 2016). We modeled these semantic relationships between items by modifying their co-occurrence probabilities (see Section 2.3). We found that, even though semantic information only impacted the input distribution and was never targeted by any training task, as soon as models were trained on sequences with semantic structure, they showed such false memory effects (Figure 2c). Specifically, the probability of false recognition increased monotonically with the fraction of semantically related items in the study sequence (Figure 3c). The presence of this phenomenon depended on neither model architecture, nor training task. Its magnitude, however, varied between different models and tasks and over the course of optimization (Figure 2h). We also observed that learned embeddings of semantically related items were more similar than embeddings of unrelated items (Figure 3d). These results suggest that models grouped semantically related items into similar embeddings, which plausibly supported their recognition decisions.

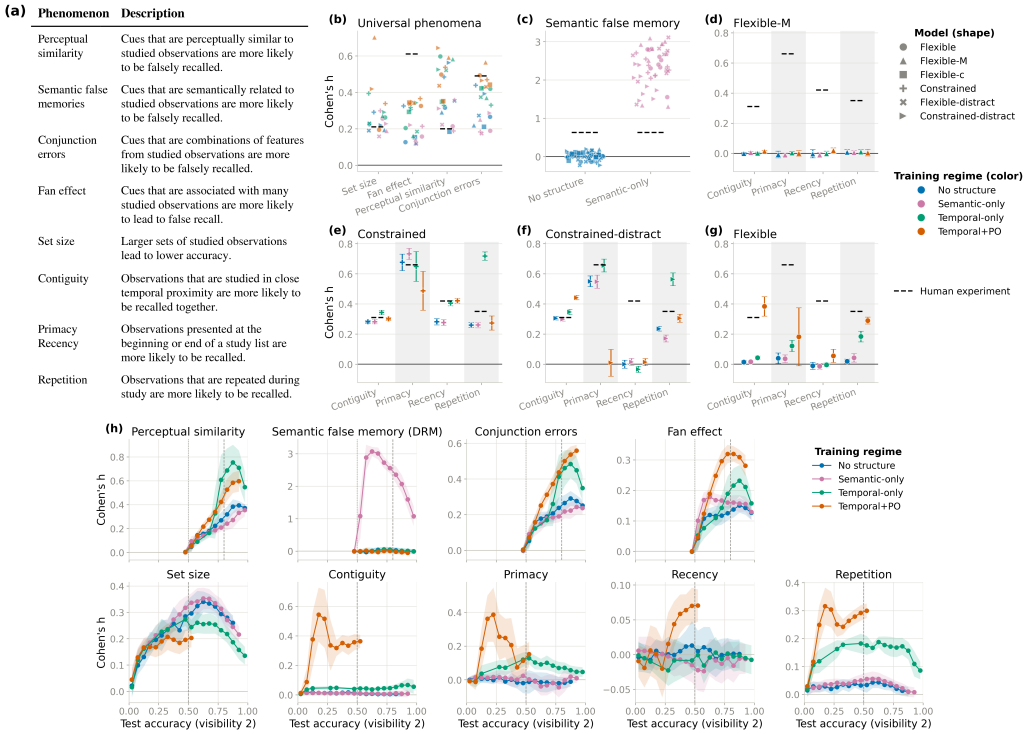


Figure 2: Presence of phenomena in trained models. (a) Description of all phenomena considered. Lists of experimental works reporting each phenomenon are shown in Table S4 and Table S5. (b) Effect size of four phenomena: set size, fan effect, perceptual similarity, and conjunction errors, for all model architectures and training regimes (trained on all tasks; results for single-task training are shown in Figure S2). For each model and training regime, we selected the training epoch closest to a target accuracy observed in humans and reported the average over 10 repetitions of the experiment (different random initializations) as a single dot. Effect sizes were measured using Cohen's h; the corresponding value from human experiments is shown for comparison. For details on the training and testing procedure, see Section A.4. (c) Semantic false memory effect for all model architectures, in training regimes with and without semantic structure. Each dot represents a single model repetition. (d)-(g) Effect size of four remaining phenomena (contiguity, primacy, recency, and repetition) for four different models: Flexible-M in (d), Constrained in (e), Constrained-distract in (f), and Flexible in (g). In each panel, values were averaged across repetitions for each training regime, with error bars indicating the standard deviation over repetitions. (h) Effect sizes across training. We focused on the Flexible model and reported the same values as in (b), (c), and (g), at different points during training, binned by the model's accuracy on the corresponding evaluation task (no input structure, 2 unmasked items per step). Dashed vertical lines show the target accuracy used to select epochs in (b)-(f). Results were averaged over 30 random seeds, and shaded areas indicate standard deviation.

3.3 Contiguity depends on architecture, training task, and input distribution

In humans, recall of one memory tends to trigger recall of temporally nearby memories, producing particular patterns of intrusions and errors (Roediger & McDermott, 1995; Webb et al., 2016; Kahana & Caplan, 2002), where the probability of an intrusion from an observation decreases with its temporal distance to the cued item. This contiguity effect is classically explained by Temporal Context Models (TCM) (Howard & Kahana, 2002; Polyn et al., 2009), in which a slowly drifting context vector is associated with each observation, so that temporally close items come to share similar context. In our setup, we were able to design a model (constrained model), with a similarly enforced mechanism, which showed contiguity in every scenario we considered (Figure 2e). Tuning the model architecture can even isolate forward-only or backward-only contiguity, as shown by the Constrained-cxc and Constrained-xcc models (Figure 4c). On the other hand, flexible models can, in principle, learn to ignore information about neighboring views entirely to solve recognition and

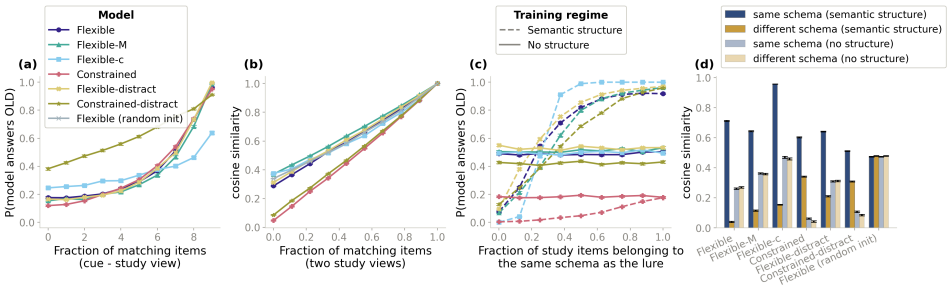


Figure 3: Perceptual and semantic similarity effects. **(a)-(b) Perceptual similarity effect.** Models were trained on all tasks in an environment with no input structure. (a) Probability that the model outputs OLD as a function of the fraction of matching items between the cue and a target study view. We used study sequences of length 8, where each study input had all 9 item positions unmasked, followed by a recognition cue that matched a target study view on some fraction of the positions and shared no content with the other study views. At a fraction of 1, and only there, was OLD the correct answer. (b) Cosine similarity between the embeddings of two study views as a function of the fraction of matching items. We considered two fully unmasked study views that match on some fraction of the positions. **(c)-(d) Semantic similarity effect.** Models were trained on all tasks, in environments with or without semantic structure. (c) Probability that the model answers OLD as a function of the fraction of study items belonging to the same schema as the cued item. We used study sequences of length 8, where only one item is visible at each step, followed by a recognition cue. All items were distinct, and OLD was never the correct answer. A typical DRM experiment (Roediger & McDermott, 1995) would consider scenarios with 100% of items matching the cued item's schema (critical lure) or 0% (unrelated lure). (d) Similarity between the embeddings of two study views each containing a single unmasked item, in the case where their item belong to the same schema or different schemas.

cued recall. This appears to be what they did in the 'no structure' input regime, with no contiguity effect appearing (Figure 2g). However, we found that the Flexible model can be incentivized to retrieve and use neighboring memories, by tuning the training task distribution (Figure 4a) or the input structure (Figure 4b). In both cases, contiguity emerged from training. This shows that contiguity does not need to be hardcoded into a memory system, it can emerge from the structure of the environment a model is trained in, and from the tasks it is optimized for.

Distinct forward and backward mechanisms. Unlike TCM, in which both directions of contiguity arise from a single retrieval mechanism, our models relied on two distinct ones: the forward bias arose at retrieval, while the backward bias arose at decoding, when interpreting retrieved memories. This is due to the fact that both keys and values are built causally: whatever memory is stored at time t' can only depend on items observed up to t', never on items observed later. **Forward mechanism:** at retrieval, the model observes a cue corresponding to a target study view x_t, and generates a query designed to find memories containing x_t. Only a key stored after t can contain information about x_t, and so can resemble the query. Retrieval is therefore biased towards memories stored *after* the target: a forward contiguity bias (Figure 4d). **Backward mechanism:** at decoding, once memories are retrieved, their values are read out. By the same logic, a retrieved value can only contain information of items from *before* it was stored, leading to a potential backward contiguity bias (Figure 4e). The backward contiguity bias can also depend on how easily different memories can be dissociated within a retrieved value: in the Constrained model, information about x_t is harder to recover from v_t (Figure 4f), which may increase the backward contiguity effect.

Our models thus suggest a mechanism for contiguity that differs from that of TCM. Whether the human brain uses analogous mechanisms remains open; testing this would require dissociating memory retrieval from decoding, which we leave for future work.

4 Conclusion

We introduced a framework for building task-trained models of memory, in which architecture, training tasks, and input distribution can each be manipulated independently. We used it to test

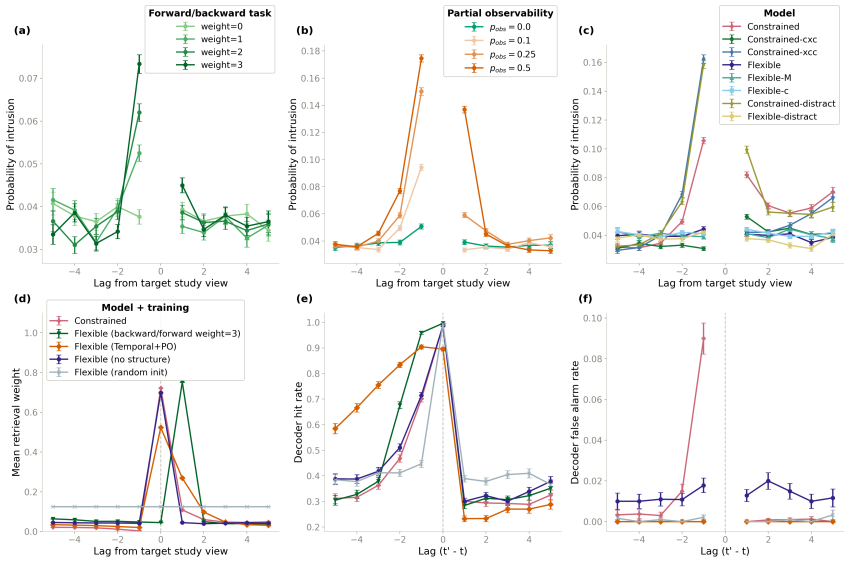


Figure 4: Contiguity effect in cued recall. (a)-(c) Probability of an intrusion from study view at $t + \text{lag}$ when the target view is at t. For testing, we used study sequences of length 8, where two items were visible at each step, and with *no input structure*. All study items were distinct. **(a) Effect of training tasks.** Results for the Flexible model trained with temporal structure on all tasks, under different task weightings. By default, during training, recall cues were drawn at random from one of the four tasks (recognition, cued recall, forward recall, backward recall), each with a sampling weight of 1. Here, we varied the sampling weights of the forward and backward tasks from 0 (never sampled) to 3 (three times more likely to be sampled than recognition or cued-recall cues). **(b) Effect of temporal structure and partial observability during training.** Results for the Flexible model trained with temporal structure and on all tasks, while varying the amount of partial observability during training, from 0 (all study items observable) to 0.5 (half of the study items are unobserved and need to be reconstructed from neighboring views). **(c) Effect of model architecture.** Results for different model architectures trained on all tasks, with no input structure. **(d) Forward mechanism: query-key retrieval.** We plot retrieval weights for stored memories (after applying softmax to query-key similarities) at recall time as a function of the lag to the target study view. **(e) Backward mechanism: decoding all items from a single value.** We used study sequences with only one unmasked position. We trained a MLP decoder to identify, from a value v_t stored at time t, the presence of all items appearing in the study sequence. We plot the decoder's hit rate for an item appearing at time t' as a function of the lag $t' - t$. Positive lags correspond to future items, which in principle cannot be decoded from the current value. Please refer to Section A.5 for more details. **(f) Backward mechanism: decoding the current item from a value.** Same procedure as in (e), but the decoder was trained to identify the single item observed at time t from the stored value v_t. The decoder achieved near-perfect accuracy for all models, except for the Constrained model, which produces a few false alarms at small negative lags.

whether nine human memory phenomena emerge under different conditions. We found that these phenomena can be built directly into a model's architecture: our constrained model shows all nine phenomena under all conditions. Most of these phenomena also emerged from training, through the task and input distribution, in flexible models with no such architectural bias. This suggests treating the parameter-fitting process as part of the modeling effort itself, rather than as a neutral means of instantiating a model to be validated afterward.

Some of the behaviors we observe in humans may be better explained by the environment the brain operates in than by anything internal to the brain, an idea developed in works on rational analysis and bounded or ecological rationality (Anderson, 2013; Gigerenzer & Todd, 1999; Lieder & Griffiths, 2020). Our framework offers a natural way to explore this idea, since environmental and architectural constraints can be manipulated independently. In this context, task-trained models are a valuable tool for studying human memory: they let us explore the conditions under which specific behaviors emerge, and can inspire new theories about the mechanisms that produce them.

AI USE STATEMENT

In this work, we used generative AI tools to aid in polishing the writing, to help find related works, and to assist with code implementation: the main code base and its structure were written by hand, generative AI tools were used to complete specific functions. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to design or provide feedback on the research methodology or experiments, or to support qualitative and thematic data analysis or the interpretation of results. The remaining tasks with required disclosure (generating synthetic datasets, cleaning and reformatting datasets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and assisting with translation) are not applicable to this work. We have reviewed all AI-assisted work. 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 made efforts to ensure that our results are reproducible. Sections 2, A.1, A.2, A.3, A.4 and A.5 describe our method in full with implementation details, including architectures, hyperparameters, training and evaluation procedures. The code used for this study is also available as a supplementary file.

REFERENCES

John R. Anderson. *The Adaptive Character of Thought*. Psychology Press, New York, January 2013. ISBN 978-0-203-77173-0. doi: 10.4324/9780203771730.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended Long Short-Term Memory, December 2024. URL http://arxiv.org/abs/2405.04517. arXiv:2405.04517 [cs].

M. Chang and C. J. Brainerd. Semantic and phonological false memory: A review of theory and data. *Journal of Memory and Language*, 119:104210, August 2021. ISSN 0749-596X. doi: 10.1016/j.jml.2020.104210. URL https://www.sciencedirect.com/science/article/pii/S0749596X20301248.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, September 2014. URL http://arxiv.org/abs/1406.1078. arXiv:1406.1078 [cs, stat].

Jennifer H. Coane, Dawn M. McBride, Mark J. Huff, Kai Chang, Elizabeth M. Marsh, and Kendal A. Smith. Manipulations of List Type in the DRM Paradigm: A Review of How Structural and Conceptual Similarity Affect False Memory. *Frontiers in Psychology*, 12:668550, May 2021. ISSN 1664-1078. doi: 10.3389/fpsyg.2021.668550. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC8200635/.

Jacob Cohen. CHAPTER 6 - Differences between Proportions. In *Statistical Power Analysis for the Behavioral Sciences*, pp. 179–213. Academic Press, January 1977. ISBN 978-0-12-179060-8. doi: 10.1016/B978-0-12-179060-8.50011-6. URL https://www.sciencedirect.com/science/article/pii/B9780121790608500116.

Orin C. Davis, Aaron S. Geller, Daniel S. Rizzuto, and Michael J. Kahana. Temporal associative processes revealed by intrusions in paired-associate recall. *Psychonomic Bulletin & Review*, 15 (1):64–69, February 2008. ISSN 1531-5320. doi: 10.3758/PBR.15.1.64. URL https://doi.org/10.3758/PBR.15.1.64.

David A. Gallo. False memories and fantastic beliefs: 15 years of the DRM illusion. *Memory & Cognition*, 38(7):833–848, October 2010. ISSN 1532-5946. doi: 10.3758/MC.38.7.833. URL https://doi.org/10.3758/MC.38.7.833.

Daniele Gatti, Luca Rinaldi, Marco Marelli, Giuliana Mazzoni, and Tomaso Vecchi. Decomposing the semantic processes underpinning veridical and false memories. *Journal of Experimental Psychology: General*, 151(2):363–389, 2022. ISSN 1939-2222. doi: 10.1037/xge0001079.

Samuel J. Gershman, Ila Fiete, and Kazuki Irie. Key-value memory in the brain. *Neuron*, 113(11): 1694–1707.e1, June 2025. ISSN 08966273. doi: 10.1016/j.neuron.2025.02.029. URL https://linkinghub.elsevier.com/retrieve/pii/S0896627325001722.

Tyler Giallanza, Declan Campbell, and Jonathan D. Cohen. Toward the Emergence of Intelligent Control: Episodic Generalization and Optimization. *Open Mind : Discoveries in Cognitive Science*, 8:688–722, May 2024. ISSN 2470-2986. doi: 10.1162/opmi_a_00143. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC11142636/.

Gerd Gigerenzer and Peter M. Todd. *Simple heuristics that make us smart*. Simple heuristics that make us smart. Oxford University Press, New York, NY, US, 1999. ISBN 978-0-19-512156-8. Pages: xv, 416.

Susanne Haridi, Eric Schulz, and Mirko Thalmann. Context Size and Set Size Effects: The Relevance of Specific Cues When Searching Long-Term Memory. *Computational Brain & Behavior*, 9(1):1–33, March 2026. ISSN 2522-087X. doi: 10.1007/s42113-025-00255-7. URL https://doi.org/10.1007/s42113-025-00255-7.

Douglas Hintzman. MINERVA 2: a simulation model of human memory. *Behav. Res. Methods Instrum. Comput.*, 16:96–101, March 1984. doi: 10.3758/BF03202365.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9 (8):1735–1780, November 1997. ISSN 1530-888X. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Marc W. Howard and Michael J. Kahana. A Distributed Representation of Temporal Context. *Journal of Mathematical Psychology*, 46(3):269–299, June 2002. ISSN 0022-2496. doi: 10.1006/jmps.2001.1388. URL http://dx.doi.org/10.1006/jmps.2001.1388.

Yoonhee Jang and Heungchul Lee. Item repetition and retrieval processes in cued recall: Analysis of recall-latency distributions. *Memory & Cognition*, 47(4):792–815, May 2019. ISSN 1532-5946. doi: 10.3758/s13421-019-00902-y. URL https://doi.org/10.3758/s13421-019-00902-y.

Tim Jones and Klaus Oberauer. Serial-position effects for items and relations in short-term memory. *Memory*, 21(3):347–365, April 2013. ISSN 0965-8211, 1464-0686. doi: 10.1080/09658211.2012.726629. URL http://www.tandfonline.com/doi/abs/10.1080/09658211.2012.726629.

Michael J. Kahana and Jeremy B. Caplan. Associative asymmetry in probed recall of serial lists. *Memory & Cognition*, 30(6):841–849, September 2002. ISSN 1532-5946. doi: 10.3758/BF03195770. URL https://doi.org/10.3758/BF03195770.

Michael J. Kahana, Nicholas B Diamond, and Ada Aka. Laws of Human Memory. *PsyArXiv Preprints*, April 2022. doi: 10.31234/osf.io/aczu9. URL http://dx.doi.org/10.31234/osf.io/aczu9.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention, August 2020. URL http://arxiv.org/abs/2006.16236. arXiv:2006.16236 [cs].

Moufan Li, Kristopher T. Jensen, Qiong Zhang, Qihong Lu, and Marcelo G. Mattar. A neural network model of free recall learns multiple memory strategies. *Nature Machine Intelligence*, 8(8):1238–1250, August 2026. ISSN 2522-5839. doi: 10.1038/s42256-026-01274-0. URL https://www.nature.com/articles/s42256-026-01274-0.

Falk Lieder and Thomas L. Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1, 2020. ISSN 0140-525X, 1469-1825. doi: 10.1017/S0140525X1900061X. URL https://www.cambridge.org/core/product/identifier/S0140525X1900061X/type/journal_article.

Qihong Lu, Uri Hasson, and Kenneth A Norman. A neural network model of when to retrieve and encode episodic memories. *eLife*, 11, February 2022. ISSN 2050-084X. doi: 10.7554/elife.74445. URL http://dx.doi.org/10.7554/eLife.74445.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-Value Memory Networks for Directly Reading Documents. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1400–1409, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1147. URL https://aclanthology.org/D16-1147/.

Robert M. Nosofsky. Attention, similarity, and the identification–categorization relationship. *Journal of Experimental Psychology: General*, 115(1):39–57, 1986. ISSN 1939-2222. doi: 10.1037/0096-3445.115.1.39.

Emilia P. Piwek, Mark G. Stokes, and Christopher Summerfield. A recurrent neural network model of prefrontal brain activity during a working memory task. *PLOS Computational Biology*, 19(10):e1011555, October 2023. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1011555. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1011555.

Sean M. Polyn, Kenneth A. Norman, and Michael J. Kahana. A context maintenance and retrieval model of organizational processes in free recall. *Psychological Review*, 116(1):129–156, 2009. ISSN 0033-295X. doi: 10.1037/a0014420. URL http://dx.doi.org/10.1037/a0014420.

A. Pritzel, B. Uria, S. Srinivasan, A. Puigdomènech, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell. Neural Episodic Control. *ArXiv e-prints*, March 2017. _eprint: 1703.01988.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 5301–5310. PMLR, May 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

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. *arXiv e-prints*, pp. arXiv:2008.02217, July 2020. URL https://arxiv.org/abs/2008.02217. _eprint: 2008.02217.

Mark Tippens Reinitz, William J. Lammers, and aBarbara Pitt's Cochran. Memory-conjunction errors: Miscombination of stored stimulus features can produce illusions of memory. *Memory & Cognition*, 20(1):1–11, January 1992. ISSN 1532-5946. doi: 10.3758/BF03208247. URL https://doi.org/10.3758/BF03208247.

Samuel Ritter, Jane Wang, Zeb Kurth-Nelson, Siddhant Jayakumar, Charles Blundell, Razvan Pascanu, and Matthew Botvinick. Been There, Done That: Meta-Learning with Episodic Recall. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4354–4363. PMLR, July 2018. URL https://proceedings.mlr.press/v80/ritter18a.html.

Henry L. Roediger and Kathleen B. McDermott. Creating false memories: Remembering words not presented in lists. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 21 (4):803–814, 1995. ISSN 1939-1285. doi: 10.1037/0278-7393.21.4.803.

Nikolaus Salvatore and Qiong Zhang. Sequence-to-sequence models with attention mechanistically map to the architecture of human memory search. *Communications Psychology*, 3(1): 146, October 2025. ISSN 2731-9121. doi: 10.1038/s44271-025-00322-6. URL https://www.nature.com/articles/s44271-025-00322-6.

Daniel L. Schacter. *The seven sins of memory: how the mind forgets and remembers*. Mariner Books, Boston, updated edition edition, 2021. ISBN 978-0-547-34745-5 978-0-358-32568-0 978-0-358-29699-7 978-0-358-29705-5.

Hayoung Song, Qihong Lu, Tan T. Nguyen, Janice Chen, Yuan Chang Leong, Monica D. Rosenberg, ShiNung Ching, and Jeffrey M. Zacks. A neural network with episodic memory learns causal relationships between narrative events, September 2025. URL https://www.biorxiv.org/content/10.1101/2025.09.01.673596v1. ISSN: 2692-8205 Pages: 2025.09.01.673596 Section: New Results.

Shauna M. Stark, Michael A. Yassa, Joyce W. Lacy, and Craig E.L. Stark. A task to assess behavioral pattern separation (BPS) in humans: Data from healthy aging and mild cognitive impairment. *Neuropsychologia*, 51(12):2442–2449, October 2013. ISSN 0028-3932. doi: 10.1016/j.neuropsychologia.2012.12.014. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC3675184/.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-To-End Memory Networks, November 2015. URL http://arxiv.org/abs/1503.08895. arXiv:1503.08895 [cs].

Endel Tulving and Tannis Y. Arbuckle. Sources of intratrial interference in immediate recall of paired associates. *Journal of Verbal Learning and Verbal Behavior*, 1(5):321–334, February 1963. ISSN 00225371. doi: 10.1016/S0022-5371(63)80012-2. URL https://linkinghub.elsevier.com/retrieve/pii/S0022537163800122.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. URL http://arxiv.org/abs/1706.03762. arXiv:1706.03762 [cs.CL].

Michael F. Verde. Associative interference in recognition memory: A dual-process account. *Memory & Cognition*, 32(8):1273–1283, December 2004. ISSN 1532-5946. doi: 10.3758/BF03206318. URL https://doi.org/10.3758/BF03206318.

Christina E. Webb, Indira C. Turney, and Nancy A. Dennis. What's the gist? The influence of schemas on the neural correlates underlying true and false memories. *Neuropsychologia*, 93:61–75, December 2016. ISSN 00283932. doi: 10.1016/j.neuropsychologia.2016.09.023. URL https://linkinghub.elsevier.com/retrieve/pii/S0028393216303608.

James C. R. Whittington, Joseph Warren, and Timothy E. J. Behrens. Relating transformers to models and neural representations of the hippocampal formation, March 2022. URL http://arxiv.org/abs/2112.04035. arXiv:2112.04035 [cs].

Yicong Zheng, Nora Wolf, Charan Ranganath, Randall C. O'Reilly, and Kevin L. McKee. Flexible Prefrontal Control over Hippocampal Episodic Memory for Goal-Directed Generalization, 2025. URL https://arxiv.org/abs/2503.02303. Version Number: 3.

Zhenglong Zhou, Michael J. Kahana, and Anna C. Schapiro. A unifying account of replay as context-driven memory reactivation. *eLife*, 13, September 2024. doi: 10.7554/eLife.99931.1. URL https://elifesciences.org/reviewed-preprints/99931.

A APPENDIX

A.1 INPUT FORMAT DETAILED

Input format. At each timestep t, the model receives an observation $x_t = (\omega_t^1, \ldots, \omega_t^K)$, a set of K co-occurring items drawn from a finite vocabulary Ω (Figure 1c). A sequence consists of T successive observations $x_1, \ldots, x_T$. The rest of this section shows how this format lets us add different types of structures and regularities, and simulate different kinds of experiments.

Semantic structure. The item vocabulary Ω is partitioned into n_{schemas} disjoint schemas. A schema is sampled once per sequence, and each item is drawn from that schema with probability p_{schema} and uniformly from the rest of the vocabulary otherwise. Larger p_{schema} increases the tendency of co-occurring items to belong to the same schema.

Temporal structure. For each of the K positions independently, the item at timestep t is copied from timestep $t-1$ with probability p_{temporal}, and resampled otherwise. Larger p_{temporal} increases similarity between neighboring observations.

Partial observability. Independently of the structure above, each individual item ω_t^i is replaced by a mask token with probability p_{obs}. Importantly, these items are unobserved but can still be targeted by recall cues (see Section 2.4). This exposes the model to sequences in which any given item may be temporarily unavailable, and must be reconstructed from neighboring timesteps, exploiting the temporal structure.

Masking (field of view). Finally, a subset of the K positions is masked for the *entire* sequence. At training time each position is independently masked with probability p_{masked}; at test time, the masked positions are chosen to match the target paradigm: masking all but one position simulates a list of single items, masking all but two simulates paired associates, and leaving more positions unmasked models richer inputs such as images or scenes.

Regimes considered. In our simulations, we consider four main input structure regimes, which are defined in Table S1.

Table S1: Study sequence parameters for the different input regimes.

Regime	p_{schema}	p_{temp}	p_{obs}
No structure	$1/n_{\text{schemas}}$	0	0
Semantic-only	0.75	0	0
Temporal-only	$1/n_{\text{schemas}}$	0.75	0
Temporal + partial observability	$1/n_{\text{schemas}}$	0.75	0.5

Implementation details. In our simulations, we use a vocabulary size of $|\Omega| = 64$ and $n_{\text{schemas}} = 4$. Each input x_t contains up to $K = 9$ co-occurring items. In addition to the study sequence and recall cue, input sequences contains two special inputs: a start input x_0 filled with special token ω_{start} and a separator input x_{T+1} filled with special token $\omega_{\text{separator}}$. During training, the sequence length T varies between 2 and 16.

A.2 MODELS DETAILED

At each timestep, the model's components are updated as:

$$
\begin{aligned}
x'_t &= \phi_x(x_t) \\
&= W_2^x \,\text{ReLU}\left(W_1^x \,\text{concat}(\omega_t'^1, \ldots, \omega_t'^K) + b_1^x\right) + b_2^x \\
c_t &= f_t^c \odot c_{t-1} + i_t^c \odot c_t^{in}
\end{aligned}
$$

$$
\begin{aligned}
r_t &= READ(M_{t-1}, q_t) & q_t &= \phi_q(x'_t, c_t) \\
M_t &= WRITE(M_{t-1}, k_t, v_t) & k_t &= \phi_k(x'_t, c_t, r_t) \\
& & v_t &= \phi_v(x'_t, c_t, r_t)
\end{aligned}
$$

$$
h_t = W^h \text{concat}(x'_t, c_t, r_t) + b^h
$$

where all $\omega' \in \mathbb{R}^D$ are learned embedding of items from the vocabulary Ω, $W_1^x \in \mathbb{R}^{2D \times KD}$, $W_2^x \in \mathbb{R}^{D \times 2D}$, $W^h \in \mathbb{R}^{D \times 3D}$, $b_1^x \in \mathbb{R}^{2D}$, $b_2^x \in \mathbb{R}^D$, $b^h \in \mathbb{R}^D$. The READ and WRITE operations are described in the main text (Section 2.2). The gates f_t^c and i_t^c may vary between different models (Table 1, Table S6). Models use a hidden dimension of $D = 64$.

The WRITE operation consists in appending k_t to a matrix of keys, and v_t to a matrix of values:

$$
K_t = \begin{bmatrix} k_1^\top \\ k_2^\top \\ \vdots \\ k_t^\top \end{bmatrix} \qquad V_t = \begin{bmatrix} v_1^\top \\ v_2^\top \\ \vdots \\ v_t^\top \end{bmatrix}
$$

A.3 TASKS DETAILED

As described in Section 2.3, a model observes a sequence of study inputs $x_1, \ldots, x_T$, where each $x_t = (\omega_t^1, \ldots, \omega_t^K)$ is a set of K co-occurring items, followed by a separator input x_{T+1} and a recall cue x_{T+2}. The model produces an output state h_{T+2}. Then, a MLP classifier takes h_{T+2} alongside a task indicator $u_{T+2} = (u^{\text{task}}, u^{\text{pos}})$ and outputs a probability distribution over $\Omega \cup \{\text{OLD}, \text{NEW}\}$. Here, $u^{\text{task}} \in \{0, 1, 2, 3\}$ specifies which memory task to perform, and $u^{\text{pos}} \in \{1, \ldots, K\}$ indexes a particular position within the observation. The four memory tasks with corresponding targets are depicted in Figure 1d and summarised in Table S2.

The memory model's output and the task indicator are integrated into a final MLP classifier as follows:

$$
\begin{aligned}
\tilde{u}^{\text{pos}} &= W_1^u u^{\text{pos}} \\
\tilde{u}^{\text{task}} &= W_2^u u^{\text{task}} \\
h'_{T+2} &= h_{T+2} + \tilde{u}^{\text{pos}} + \tilde{u}^{\text{task}} \\
\hat{y}_{T+2} &= W_2^y \,\text{ReLU}\left(W_1^y \, h'_{T+2} + b_1^y\right) + b_2^y
\end{aligned}
$$

where z is one-hot encoded (or all-zero when position is irrelevant for the task), u^{task} is one-hot encoded, $W^z \in \mathbb{R}^{D \times K}$, $W^i \in \mathbb{R}^{D \times 4}$,$W_1^y \in \mathbb{R}^{2n_{\text{outputs}} \times D}$, $W_2^y \in \mathbb{R}^{n_{\text{outputs}} \times 2n_{\text{outputs}}}$, $b_1^y \in \mathbb{R}^{2n_{\text{outputs}}}$, $b_2^y \in \mathbb{R}^{n_{\text{outputs}}}$ with $n_{\text{outputs}} = |\Omega| + 2$ (vocabulary size and OLD/NEW).

A.4 TRAINING AND TESTING

We train models with cross-entropy loss between the predicted distribution $\hat{y}$ and the target label y, for 500 epochs each composed of 5000 sequences. After each epoch, we save the state of the models and evaluate their performance on two target tasks: recognition and cued recall of pairs of items, with no input structure. For evaluating phenomena, unless stated otherwise, we select the epoch where the model was closest to human accuracy on the target tasks. Human accuracy for

Task	u^{task}	**Target** y	**Description**
Recognition	0	OLD / NEW	The cue x_{T+2} is either a negative cue filled with random items (matching the masking pattern of the study sequence) or a positive cue corresponding to a random study view. In both cases, we apply additional masking to the cue. The task is to distinguish positive from negative cues.
Cued recall	1	$\omega_{\tau}^{u^{\text{pos}}}$	The cue x_{T+2} corresponds to a random study view x_τ (with any potential partial observability removed) with a random position u^{pos} masked. The task is to reconstruct the item at position u^{pos}.
Forward recall	2	$\omega_{\tau+1}^{u^{\text{pos}}}$	The cue x_{T+2} is equal to a random past observation x_τ (avoiding last study observation), removing any potential partial observability, and u^{pos} corresponds to a random unmasked position. The task it to predict the item at position u^{pos} in the subsequent observation.
Backward recall	3	$\omega_{\tau-1}^{u^{\text{pos}}}$	The cue x_{T+2} is equal to a random past observation x_τ (avoiding first study observation), removing any potential partial observability, and u^{pos} corresponds to a random unmasked position. The task is to predict the item at position u^{pos} in the preceeding observation.

Table S2: Summary of the four memory tasks.

recognition task varies depending on experiments and modalities, with hit rate ranging from around 0.6 to close to 0.9, and false alarm rate ranging from around 0.1 to around 0.4. We choose a target accuracy of 0.8 for recognition of paired associates. Similarly, accuracy for cued recall task varies between experiments, ranging from around 0.3 to 0.8. We choose a target accuracy of 0.5 for cued recall of paired associates. We repeat the training and testing processes 10 times with different seeds. Models whose accuracy never came within 0.1 of the target are excluded from the main results.

Implementation details To accelerate training and testing, we batch multiple recall cues: for a given study sequence of length T, we give T different recall cues to the model in parallel. Each phenomenon is evaluated using 2000 test sequences.

Effect size computations. All phenomena are evaluated with Cohen's h Cohen (1977), a measure of the difference between two proportions. For two probabilities $p_1, p_2 \in [0, 1]$, it is defined as

$$h = 2\arcsin(\sqrt{p_1}) - 2\arcsin(\sqrt{p_2}), \tag{1}$$

The sign of h indicates the direction of the effect: $h > 0$ when $p_1 > p_2$. The settings used and the probabilities compared for each phenomenon are described in Table S3.

A.5 Contiguity decoding experiments

In Figure 4, we performed some analyses with a MLP decoder. We describe them in more details here. In Figure 4e, we presented the model with sequences of $T = 8$ study views, each containing a single item at a random, unmasked pixel position (all other pixels masked), with all items drawn without replacement so no item repeats within a sequence. We recorded the model's value vector v_t, the representation it computes to write into memory at study step t, at every step $t \in 1, \ldots, T$. We used 1000 such sequences (80/20 train/test split), and trained a single multi-layer perceptron decoder (one hidden layer of 128 units, ReLU, linear output over all vocabulary items) to predict, from v_t alone, a multi-hot label over the full vocabulary. An item is labeled positive if it occurs *anywhere* in the sequence, regardless of position. The decoder is trained with binary cross-entropy (one sigmoid output per item), with positive/negative class balancing. At test time, we use held out sequences never used in training and ask: for a given lag $\ell = t' - t$, does the decoder's output from

Phenomenon	Task	Number of unmasked items	p_1	p_2
Perceptual similarity	Recognition	8	$p(\text{OLD}\|$ cue matches half of a study view)	$p(\text{OLD}\|$ no item match between cue and any study view)
Semantic false memory	Recognition	1	$p(\text{OLD}\|$ cue is from the same schema as study items)	$p(\text{OLD}\|$ cue is from a different schema)
Conjunction errors	Recognition	8	$p(\text{OLD}\|$ cue mixes two study views)	$p(\text{OLD}\|$ cue matches half of a study view)
Fan effect	Recognition	2	$p(\text{OLD}\|$ a cued item matches two study views, the other matches no study view)	$p(\text{OLD}\|$ a cued item matches one study view, the other matches no study view)
Set size effect	Cued recall	2	$p(\text{correct}\|T \in \{4, 5, 6\})$	$p(\text{correct}\|T \in \{10, 11, 12\})$
Contiguity	Cued recall	2	$\frac{1}{2}\sum_{l\in\{+1,-1\}} p(\text{lag} = l)$	$\frac{1}{\|L\|}\sum_{l\in L} p(\text{lag} = l)$
Primacy	Cued recall	2	$p(\text{correct}\|$ cue targets first study views (1 or 2))	$p(\text{correct}\|$ cue targets middle study views (4 or 5))
Recency	Cued recall	2	$p(\text{correct}\|$ cue targets last study views (7 or 8))	$p(\text{correct}\|$ cue targets middle study views (4 or 5))
Repetition	Cued recall	9	$p(\text{correct}\|$ cued observation appeared twice in the study sequence)	$p(\text{correct}\|$ cued observation appeared once in the study sequence)

Table S3: Settings used to evaluate the effect size (Cohen's h) of all phenomena. The tests always compare two probabilities p_1 and p_2. In all cases, we use a sequence length of $T = 8$, except for the set size effect where T is variable. In all cases, study views are independent (no input structure), except for semantic false memories where all study items are drawn from the same schema. For contiguity, L contains all possible lags except $-1, 0, +1$, and $p(\text{lag} = l) = \frac{\#\text{ intrusions at lag } l}{\#\text{ times lag } l \text{ is possible}}$.

v_t correctly flag the item that was actually shown at study position $t' = t + \ell$? Averaging this hit rate over all valid (t, ℓ) pairs and test sequences gives the curve in panel (e).

Figure 4f uses an identical probe sequences, decoder architecture, and training procedure, with one change: the label at each step t is positive only for the item shown *at that step*, and negative for every other item, including ones that occur elsewhere in the same sequence. Hence, results at any lag $l \neq 0$ is a false alarm.

Table S4: Exemplar papers on memory phenomena based on recognition. Effect sizes are computed as described in Section A.4.

Phenomenon	**Paper**	**Task+modality**	**Seq. length**	**Hit rate**	p_1 **(FA for lures)**	p_2 **(FA for foil)**	**Cohen's h**
Perceptual similarity	Stark et al. (2013)	Recognition of images	192	0.814	0.512	0.412	0.2
Semantic false memories	Webb et al. (2016)	Recognition of words	26	0.63	0.54	0.24	0.63
Conjunction errors	Reinitz et al. (1992)	Recognition of images	6	0.87	0.46	0.23	0.49
Fan effect	Verde (2004)	Recognition of sentence	64	0.73	0.31	0.08	0.61

Table S5: Exemplar papers on memory phenomena based on cued recall. Effect sizes are computed as described in Section A.4.

Phenomenon	**Paper**	**Task+modality**	**Seq. length**	**Accuracy**	p_1	p_2	**Cohen's h**
Set size effect	Haridi et al. (2026)	Cued recall of word pairs	2-32	0.82	0.86	0.78	0.21
Contiguity	Davis et al. (2008)	Cued recall of word pairs	12	0.60	0.17	0.07	0.31
Primacy	Tulving & Arbuckle (1963)	Cued recall of (digit, word)	10	0.424	0.634	0.310	0.66
Recency	Tulving & Arbuckle (1963)	Cued recall of (digit, word)	10	0.424	0.517	0.310	0.42
Repetition	Jang & Lee (2019)	Cued recall of word pairs	60	0.29	0.46	0.29	0.35

Table S6: Existing memory models expressed in a common notation (see Section 2.2), with the components that define their state update, write, and read operations. For all models with an episodic memory, the READ operation is performed as $r_t = V_t^\top w_t$. Notation: $l(\cdot)$: learned linear map; $\odot$: element-wise product, "$-$": component absent. Notes and details for specific models are provided below, including variations to the equations in Section 2.2.

Model	ϕ_c			**WRITE**		**READ**		ϕ_h
	f_t^c	i_t^c	c_t^{in}	k_t	v_t	q_t	w_t	h_t
Vanilla RNN	0	1	$\tanh(l(x_t)+l(c_{t-1}))$	$-$	$-$	$-$	$-$	c_t
LSTM (Hochreiter & Schmidhuber, 1997)	$\sigma(l(x_t)+l(h_{t-1}))$	$\sigma(l(x_t)+l(h_{t-1}))$	$\tanh(l(x_t)+l(h_{t-1}))$	$-$	$-$	$-$	$-$	$o_t \odot \tanh(c_t)$
GRU (Cho et al., 2014)	$\sigma(l(x_t)+l(c_{t-1}))$	$1-f_t^c$	$\tanh(l(x_t)+l(u_t \odot c_{t-1}))$	$-$	$-$	$-$	$-$	c_t
Transformer (Vaswani et al., 2023)	$-$	$-$	$-$	$l(x_t)$	$l(x_t)$	$l(x_t)$	$\text{softmax}(\frac{K_t q_t}{\sqrt{d_k}})$	$\text{LayerNorm}(\tilde{h}_t+\tilde{r}_t)$
Linear transformer (Katharopoulos et al., 2020)	$-$	$-$	$-$	$l(x_t)$	$l(x_t)$	$l(x_t)$	$\frac{\theta(K_t)\theta(q_t)}{\sum \theta(K_t)\theta(q_t)}$	$f_l(r_t+x_t)$
TCM (Howard & Kahana, 2002)	ρ_t	β	r_t^c	c_t x_t	x_t c_t	c_{t-1} x_t	$K_t q_t$	r_t^x
CMR (study)	ρ_t	β	$M_{\text{pre}}^{XC} x_t$	c_{t-1} x_t	x_t c_{t-1}	$-$	$-$	$-$
CMR (recall) (Polyn et al., 2009)	ρ_t'	β'	$(1-\gamma)M_{\text{pre}}^{XC} x_{t-1}+\gamma r_{t-1}^c$	$-$	$-$	c_t x_t	$K_t q_t$	r_t^x
Lu et al. (2022)	$\sigma(l(x_t)+l(h_{t-1}))$	$\sigma(l(x_t)+l(h_{t-1}))$	$\tanh(l(x_t)+l(h_{t-1}))$	c_t+r_t	c_t+r_t	c_t	$\text{LCA}(\hat{K}_t \hat{q}_t)$	$\text{relu}(l(o_t \odot \tanh(c_t+r_t)))$
Li et al. (2026) (study)	$\sigma(l(x_t)+l(c_{t-1}))$	$1-f_t^c$	$\tanh(l(x_t)+l(u_t \odot c_{t-1}))$	c_t	c_t	$-$	$-$	$-$
(recall)	$\sigma(l(x_t)+l(c_{t-1}))$	$1-f_t^c$	$\tanh(l(x_t)+l(u_t \odot c_{t-1})+r_t)$	$-$	$-$	c_{t-1}	$\text{softmax}(\hat{K}_t \hat{q}_t)$	c_t
Salvatore & Zhang (2025) (study)	0	1	$\tanh(l(x_t)+l(c_{t-1}))$	c_t	c_t	$-$	$-$	$-$
(recall)	0	1	$\tanh(l(x_t)+l(c_{t-1}))$	$-$	$-$	c_t	$\text{softmax}(K_t q_t)$	$\tanh(l(c_t : r_t))$
Modern Hopfield Network (Ramsauer et al., 2020)	$-$	$-$	$-$	x_t	x_t	x_t	$\text{softmax}(K_t q_t)$	r_t
DND (Pritzel et al., 2017)	$-$	$-$	$-$	x_t	$\hat{Q}(s_t, a)$	x_t	$w_{t,i} = \frac{\theta(q_t, k_i)}{\sum_j \theta(q_t, k_j)}$	r_t
epLSTM (Ritter et al., 2018)	$\sigma(l(x_t)+l(h_{t-1}))$	$\sigma(l(x_t)+l(h_{t-1}))$	$\tanh(l(x_t)+l(h_{t-1}))$ r_t	$\tilde{x}_t$	c_t	$\tilde{x}_t$	$w_{t,i} = \frac{\theta(q_t, k_i)}{\sum_j \theta(q_t, k_j)}$	$o_t \odot \tanh(c_t)$
Song et al. (2025)	$\sigma(l(x_t)+l(c_{t-1}))$	$1-f_t^c$	$\tanh(l(x_t)+s_t \odot l(c_{t-1}))$	$l(x_t)$	c_t	$l(x_t)$	$\text{softmax}(\frac{K_t q_t}{\tau \sqrt{d_k}})$	$\alpha c_t+(1-\alpha)r_t$
Piwek et al. (2023)	0	1	$\text{ReLU}(l(c_{t-1})+l(x_t))$	$-$	$-$	$-$	$-$	$l(c_t)$

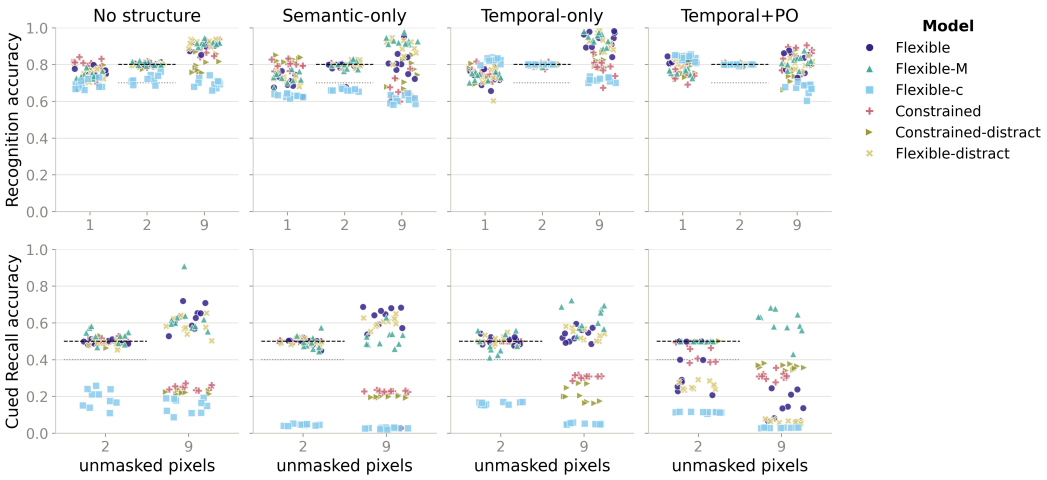
In models from Table S6, several variations on the equations in Section 2.2 are worth noting explicitly:

- **Positional encoding.** Some models (Vaswani et al., 2023; Katharopoulos et al., 2020) use positional encoding to capture temporal relations between inputs in a sequence. This consists of adding a fixed or learned encoding of each input's position in the sequence to x_t, giving the model access to timestep information.
- **Read-before-update.** Some models perform the READ operation before updating c_t (e.g. in Howard & Kahana (2002), Li et al. (2026), Ritter et al. (2018)) rather than after.
- **Gated writing.** Some models have a mechanism to decrease the memory traces corresponding to older inputs relative to recent ones when writing to M_t, using a forget gate, and to tune how much importance to give to the newly generated key-value pair, using an input gate. These gates can be fixed scalars (Howard & Kahana, 2002; Polyn et al., 2009), or learned (Beck et al., 2024).
- **Compact memory representations.** Depending on the READ operation used, storing growing matrices of keys and values explicitly is not necessary, and a compact representation for M_t can be used instead. For instance, we have $M_t \in \mathbb{R}^{D\times D}$ in Howard & Kahana (2002); Polyn et al. (2009); Katharopoulos et al. (2020); Beck et al. (2024), which use $w_{t,i} = \theta(q_t, k_i) = q_t k_i^\top$, i.e. dot product similarity. Expressing such models as growing lists of keys and values is nonetheless functionally equivalent, and makes comparison across models easier.

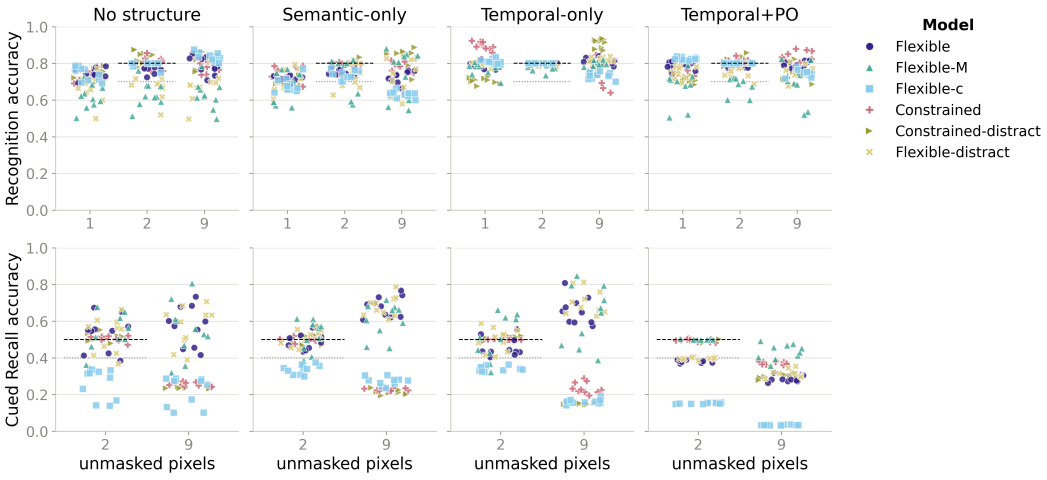
Model specific details and comments:

- **Vanilla RNN**: no gate structure; $c_t = c_t^{\text{in}} = \tanh(l(x_t) + l(c_{t-1}))$ directly.
- **LSTM** (Hochreiter & Schmidhuber, 1997): $o_t = \sigma(l(x_t) + l(h_{t-1}))$.
- **GRU** (Cho et al., 2014): the reset gate $u_t = \sigma(l(x_t) + l(c_{t-1}))$ modulates the recurrent term in c_t^{in}.
- **Transformer** (Vaswani et al., 2023): $\tilde{r}_t = \text{LayerNorm}(x_t + r_t)$, $\tilde{h}_t = l(\text{ReLU}(l(\tilde{r}_t)))$; x_t includes positional encoding.
- **Linear transformer** (Katharopoulos et al., 2020): $\theta(x) = \text{elu}(x) + 1$, and the function f_l can typically be implemented by a two-layer feedforward network.
- **TCM** (Howard & Kahana, 2002): the two stacked entries in WRITE and READ correspond to M^{CX} (top) and M^{XC} (bottom) respectively. The vector r_t^x is the retrieved item representation (top memory write and read); r_t^c is the retrieved context (bottom memory write and read). The WRITE and READ operations for M^{XC} are simplified here, they are in fact more complex than a simple outer-product accumulation: $M_t^{XC} = M_{t-1}^{XC}\tilde{P}_{x_t} + A_t M_{t-1}^{XC} P_{x_t} + B_t c_t x_t^\top$, and $r_t^c = M_{t-1}^{XC} x_t$. β is a fixed free parameter; ρ_t enforces $\|c_t\| = 1$.
- **CMR** (Polyn et al., 2009): the free parameter γ weights pre-experimental vs. retrieved context in the recall phase. The two stacked entries in WRITE and READ correspond to M^{CX} (top) and M^{XC} (bottom).
- Lu et al. (2022): LCA = Leaky Competing Accumulator on cosine similarities of n nearest neighbors, with learned gating weights $g_t = \sigma(l(\text{concat}(\tilde{h}_t, c_t)))$; $o_t = \sigma(l(x_t) + l(h_{t-1}))$. Notation: $\hat{q}_t = \frac{q_t}{\|q_t\|}$ and $\hat{K}_{t,i} = \frac{k_i}{\|k_i\|}$.
- Li et al. (2026): in the READ operation, $\hat{q}_t = \frac{q_t}{\|q_t\|}$ and $\hat{K}_{t,i} = \frac{k_i}{\|k_i\|}$. In recall, r_t from memory is blended into the recurrent term of c_t^{in}.
- Salvatore & Zhang (2025): The parameters in the linear transformations (l) are different between the study and recall phase. Also, during the recall phase, x_t is set to be the previously recalled item.
- **Modern Hopfield Network** (Ramsauer et al., 2020): no recurrent context state (c_t), purely content-addressable; keys and values are unprojected copies of the input. Typically, during recall, x_t is a partial or noisy version of a previously stored pattern.

- **DND** (Pritzel et al., 2017): proposed for reinforcement learning; v_t stores a bootstrapped action-value estimate $\hat{Q}(s_t, a)$ for the current state, and x_t is an embedding of the observation. The kernel is $\theta(q, k) = (\|q - k\|_2^2 + 10^{-3})^{-1}$, and $w_{t,i}$ above is restricted to the n nearest neighbors of q_t.
- **epLSTM** (Ritter et al., 2018): LSTM-like recurrence with an additional episodic term gated separately, $c_t = f_t^c \odot c_{t-1} + i_t^c \odot c_t^{in} + \tilde{i}_t^c \odot r_t$, shown as the second line in the c_t^{in} columns. Memory is written once at the end of a task/episode rather than every step, while reads happen every step. The kernel θ and nearest-neighbor restriction are the same as in DND.
- Song et al. (2025): GRU-like recurrence augmented with a transformer-style key/value memory; τ is a softmax temperature, and h_t linearly interpolates between the recurrent state and the retrieved memory content via α.

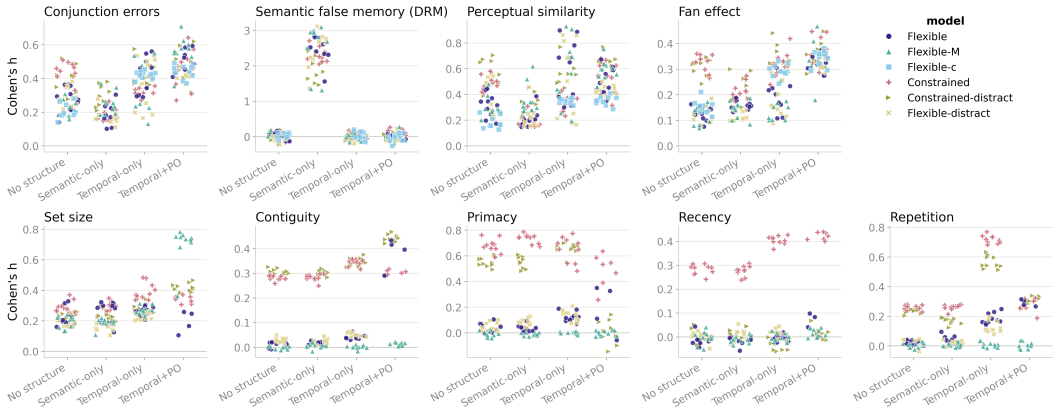


(a) Recognition and cued recall performance for models trained on all tasks.

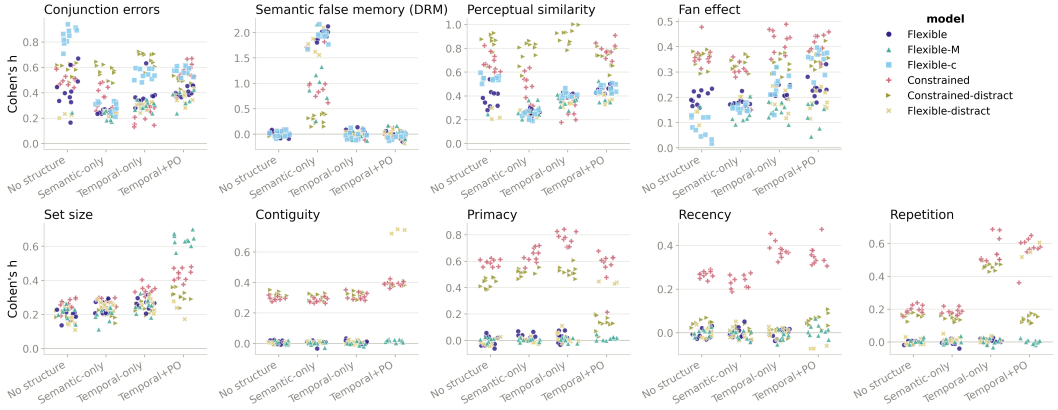


(b) Recognition and cued recall performance for models trained on a single task.

Figure S1: Performance achieved by models at the selected training epoch. As described in Section A.4, for each model we select the epoch whose accuracy is closest to the target human accuracy; this is the epoch used to compute the effect sizes reported in Figure 2. Here we report the actual accuracy achieved by the selected models, for different numbers of unmasked (visible) items per study observation. Dashed lines indicate the target accuracy, and dotted line indicate the minimal accuracy required to be considered in main analyses. (a) Performance of models trained on all tasks. Epoch selection was based on accuracy with 2 unmasked items only; performance with 1 and 9 unmasked items is shown for the same selected model, for comparison. (b) Performance of models trained on a single task. Not all models reach the target accuracy.



(a) Presence of phenomena in models trained on all tasks.



(b) Presence of phenomena in models trained on a single task.

Figure S2: All results for different model architecture trained in different input regimes. For each model and training regime, we select the training epoch closest to the target accuracy (Section A.4). Multiple dots show repetitions with different random initializations. (a) Models are trained on all tasks. (b) Models are trained on a single task.