Retention Is Allocation: Isolating What Makes a Writable Neural Memory Persist

Neil Gilani
Independent
dlake003@gmail.com

September 12, 2026

Abstract

Writable memory is usually motivated as a cure for catastrophic forgetting, and the mechanism credited with the cure is usually the write: a learned, gradient-computed update that places new information into a state without disturbing what is already there. We show, on a continual-learning stream where every ingredient can be held fixed but one, that the write is not what produces retention. Allocation is. A system that computes a gradient write into a single shared state retains 0.072 of what it learned; a system that allocates a private slot per memory but computes no write at all retains 0.988; the combination retains 1.000. Varying only the number of pages available, across a $128\times$ range, leaves immediate accuracy at exactly 1.000 at every setting while retained accuracy moves from 0.068 to 1.000 — and the retained accuracy is predicted, with no free parameters, by the fraction of memories that still hold a page of their own, to within 0.0039 accuracy across all 6 settings. Retention in this system is not a graded property of a learned update rule; it is a bookkeeping property of whether a memory was given somewhere to live. We then report the consequence that follows: because retention is allocation, the whole problem moves to *addressing*. With an oracle over page identity the system is exact; with learned content addressing it sits on a false-merge / missed-revisit frontier that meta-training improves but does not clear, and in a multi-tenant language model the gap between oracle and learned routing is the entire remaining error. We report the failures in the same detail as the successes, including two benchmarks where a plain episodic store beats us outright, and one conclusion of our own that a later measurement overturned.

1 Introduction

A network trained on a stream of tasks forgets the early ones (McCloskey and Cohen, 1989; French, 1999). The standard responses either constrain the update (Kirkpatrick et al., 2017), replay stored data (Rebuffi et al., 2017; Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2019), or move the new information out of the weights entirely, into an external memory the model reads and writes (Graves et al., 2014, 2016; Weston et al., 2015; Sukhbaatar et al., 2015; Santoro et al., 2016). The third family is the one we study, and it comes with a standard story about why it works: the memory is written by a *learned* rule — a gradient step, an attention-weighted erase-and-add, a fast-weight outer product (Schmidhuber, 1992; Ba et al., 2016) — and the learned rule is what lets a new write coexist with an old one.

That story attributes retention to the write. It is testable, and we find it is wrong, at least in the regime we can measure cleanly. The experiment that decides it is a 2×2 that the literature does not usually run, because the two factors are almost always varied together: whether the memory is *allocated* (each memory gets a region of state it does not share) and whether the write is *computed* (a gradient descends an objective, rather than a value simply being stored). Three of the four cells are familiar systems. Fast weights compute a write into one shared dense state: write, no allocation. An episodic store allocates a slot per item and does no computation at write time at all: allocation, no write. Our page memory does both.

The result is a clean dissociation (Figure 1). The cell with the computed write and no allocation retains 0.072 of a 100-task stream. The cell with allocation and no computed write retains 0.988. Whatever the gradient write is buying, it is not persistence.

Having isolated allocation, we can vary it directly, and this is the paper's central measurement (Figure 2). Holding the architecture, the objective, the optimiser, the pretraining and the stream fixed, and changing only how many pages exist, immediate accuracy is exactly 1.000 at every page count from 1 to 128. Retained accuracy moves from 0.068 to 1.000 over the same sweep. And it moves along a line with no fitted parameters: a memory is retained exactly when it still holds a page, so retained accuracy should equal chance $+$ (surviving fraction)$(1 -$ chance$)$. Measured minus predicted has a maximum absolute error of 0.0039 and a mean of 0.0031 across all 6 settings. There is no residual partial credit: an evicted memory is not degraded, it is gone, and a surviving one is not approximate, it is exact.

We take three things from this, and the third is the one that costs us something.

1. **Retention is a capacity-and-bookkeeping property, not a learned one.** If you want a writable memory to remember N things, the design question is whether N things can be allocated, not how clever the update rule is. Reporting retention without reporting the ratio of memories to available slots does not describe the system.
2. **The evaluation implication.** Immediate accuracy is uninformative about retention here — it is pinned at 1.000 across the entire range over which retention varies by a factor of fifteen. A continual-learning result that reports only end-of-stream average accuracy at one capacity setting is reporting a point on a curve whose shape it does not show.
3. **The problem moves, it does not disappear.** If retention is allocation, then the open problem is deciding *which* page a new piece of information belongs to — and that we have not solved. Section 5 reports how far short we fall.

Contributions. (i) A controlled dissociation separating allocation from the computed write, showing allocation is the factor that produces retention (Section 4.1). (ii) A parameter-free quantitative account of retention as page survival, verified to 0.0039 across a $128\times$ capacity sweep (Section 4.2). (iii) A negative result of equal weight: content addressing, not writing or retention, is the binding constraint, measured over 198 routing rules and reported as a frontier nobody in our sweep clears (Section 5). (iv) Failure modes stated as sharply as the results, including a cue-noise cliff and two benchmarks we lose (Section 7).

2 Related work

External and writable memory. Neural Turing Machines and the Differentiable Neural Computer couple a controller to an addressable memory matrix with learned read and write heads (Graves et al., 2014, 2016). Memory networks and their end-to-end variant read from a set of stored representations (Weston et al., 2015; Sukhbaatar et al., 2015), and memory-augmented meta-learners use such a store for rapid binding (Santoro et al., 2016). These systems establish that an external store helps; our question is narrower and, as far as we can tell, unasked in that line of work — which *component* of the store is responsible when it does. The DNC in particular has both an allocation mechanism (its usage-based free list) and a learned write; our result predicts that ablating the former should be far more destructive than ablating the latter.

Fast weights and test-time learning. Fast-weight memories (Schmidhuber, 1992; Ba et al., 2016) and recent test-time-training and learned-memory-module architectures (Sun et al., 2024; Behrouz et al., 2024) write into a dense state at inference. Our fast-weights condition is exactly this family with allocation removed, and it is the cell that fails to retain. We read this as a prediction rather than a criticism: dense test-time memory should degrade gracefully in the recency direction and sharply in the capacity direction, which is what we observe.

Retrieval. Retrieval-augmented models (Lewis et al., 2020; Borgeaud et al., 2022; Wu et al., 2022) and classical episodic stores (Vinyals et al., 2016; Snell et al., 2017; Chaudhry et al., 2019) allocate trivially — one entry per item — and that is precisely why they retain. Our episodic baseline is a strong one for this reason, and it beats us outright on one benchmark (Section 7). The difference that matters is cost: the episodic store grows with the raw data it keeps, at 56,448 bytes per memory against 1,164 for a page (Figure 4), and it answers only by comparison to stored exemplars, which is the wrong operation when what must be stored is a *correction* rather than an example.

Parameter-efficient adaptation. Adapters and LoRA (Houlsby et al., 2019; Hu et al., 2022) make the slow weights cheap to move but do not make them allocatable: two conflicting facts written into one adapter collide, which is visible in our fine-tuning and adapter baselines and in the multi-tenant setting of Section 7.

Complementary learning systems. The allocation/consolidation split has a long account in the memory literature (McClelland et al., 1995; Kumaran et al., 2016), and the name of the system is a nod to Hebb (1949). Our result is a small, mechanical instance of that division of labour: a fast store that allocates, a slow network that does not change at all during the stream.

3 Setup

The memory. The writable state is a set of P pages, each a small matrix of p slots of dimension d. The slow network is frozen for the entire stream; it is never updated by the stream, so every effect reported here is an effect of the memory. A write takes a task's support set, selects a page, and descends a few gradient steps on that page alone with the rest of the network and the rest of the memory held fixed. A read conditions the frozen network on the addressed page through a FiLM-style modulation (Perez et al., 2018) and an attention read head. The read path is meta-trained before the stream begins (Finn et al., 2017); the stream itself trains nothing.

Allocation. A new write either revises an existing page — when the incoming cue matches one already held — or allocates a fresh one. When no page is free, the least recently useful page is evicted, and the memory it held is lost. This is the mechanism the paper is about, and the capacity sweep is nothing more than changing P.

Benchmarks. SYMBOL MAP is a 100-task stream of 16-way symbol-to-meaning mappings, chance 0.0625; each task is independently drawn, so there is nothing to transfer and any end-of-stream accuracy above chance is retention. COMPOSITIONAL requires combining two learned rules. VISION is class-incremental image classification. VENUE is domain-incremental: the classes are fixed for the whole stream and the imaging conditions change, which is the shape most deployed perception actually has.

Metrics. *Immediate* accuracy is measured on a task right after it is learned; *retained* accuracy is measured on every task at the end of the stream. Retention is their ratio. All results are means over seeds; the seed count is stated in every table, and it is small (two to five), which we flag rather than bury.

4 Retention is allocation

4.1 The dissociation

Table 1 places the three structural conditions among the usual baselines. The pattern in the SYMBOL MAP column is the paper's starting point: fast weights learn each task well enough at the moment of learning (0.667) and retain almost none of it (0.072), while the episodic store, which performs no computation at write time whatsoever, retains everything it learned (0.988 immediate, 0.988 retained). Full fine-tuning and LoRA sit with fast weights, for the same reason: one shared state, overwritten.

This is a dissociation, not a ranking. It says that if one removes the computed write and keeps allocation, retention survives; if one removes allocation and keeps the computed write, it does not. The gradient write earns its place elsewhere — it is what lets a page store a *correction* to a frozen network rather than a copy of the data, which is what buys the $48\times$ state reduction in Figure 4 and what makes the COMPOSITIONAL column go our way (0.458 against 0.374 for episodic). It is not what makes the system remember.

4.2 Retention is exactly page survival

Table 2 is the measurement in full. Three things in it are worth separating.

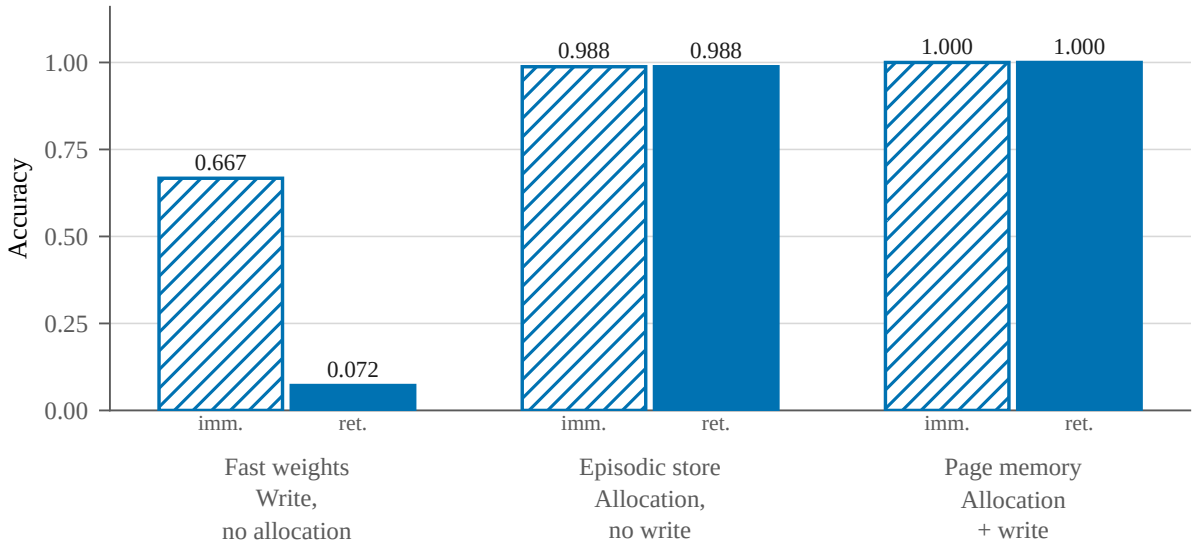

Figure 1: Three structural conditions on the same 100-task stream. Hatched bars are accuracy measured immediately after learning a task; solid bars are accuracy on the same tasks at the end of the stream. The computed gradient write, without allocation, does not survive the stream. Allocation without any computed write does. Values also appear in Table 1.

Table 1: Immediate (imm.) and end-of-stream retained (ret.) accuracy. The three structural conditions of Section 4.1 are page memory, the episodic store and fast weights. Means over seeds (Symbol map: 3 seeds; Compositional: 3 seeds; Vision: 2 seeds).

	Symbol map		Compositional		Vision	
System	imm.	ret.	imm.	ret.	imm.	ret.
Page memory (ours)	**1.000**	**1.000**	**0.458**	**0.458**	**0.697**	**0.697**
Episodic store	0.988	0.988	0.374	0.374	0.994	0.994
Fast weights	0.667	0.072	0.399	0.159	0.824	0.295
Recurrent state	0.064	0.064	0.205	0.164	0.206	0.202
Full fine-tuning	0.256	0.069	0.237	0.099	0.456	0.224
LoRA adapter	0.155	0.058	0.202	0.093	0.480	0.204

Immediate accuracy is invariant. It is 1.000 in every row, including the row with a single page. A one-page memory learns each of 100 tasks perfectly and then loses 99 of them. Any evaluation that stopped at immediate accuracy would rate all seven configurations identically.

The eviction count and the accuracy are the same number. With P pages and 100 memories, $100 - \min(P, 100)$ memories are evicted, and the fraction that survive predicts end-of-stream accuracy above chance to within 0.0039. We did not tune anything to make this hold; the prediction has no parameters to tune. A one-page run evicts 99 and scores 0.068 against a chance of 0.0625; a 128-page run evicts none and scores 1.000.

There is no graceful degradation. A dense memory under pressure produces partially corrupted recall: accuracy decaying smoothly as interference accumulates. An allocated memory under pressure produces a mixture of perfect recall and nothing. The residual is what distinguishes the two, so it is worth reading carefully rather than rounding away. Measured accuracy sits a constant -0.0031 below the prediction wherever any eviction occurs, and exactly on it when none does. That offset does not scale with memory pressure — it is the same at 36 evictions as at 99 — so it is not interference. Decomposing it gives the reason: taking accuracy on surviving memories as the 1.000 measured where nothing is evicted, the implied accuracy on the *evicted* memories is 0.052–0.059, marginally below the chance rate of 0.0625. An evicted memory is therefore not partially remembered; it is slightly worse than

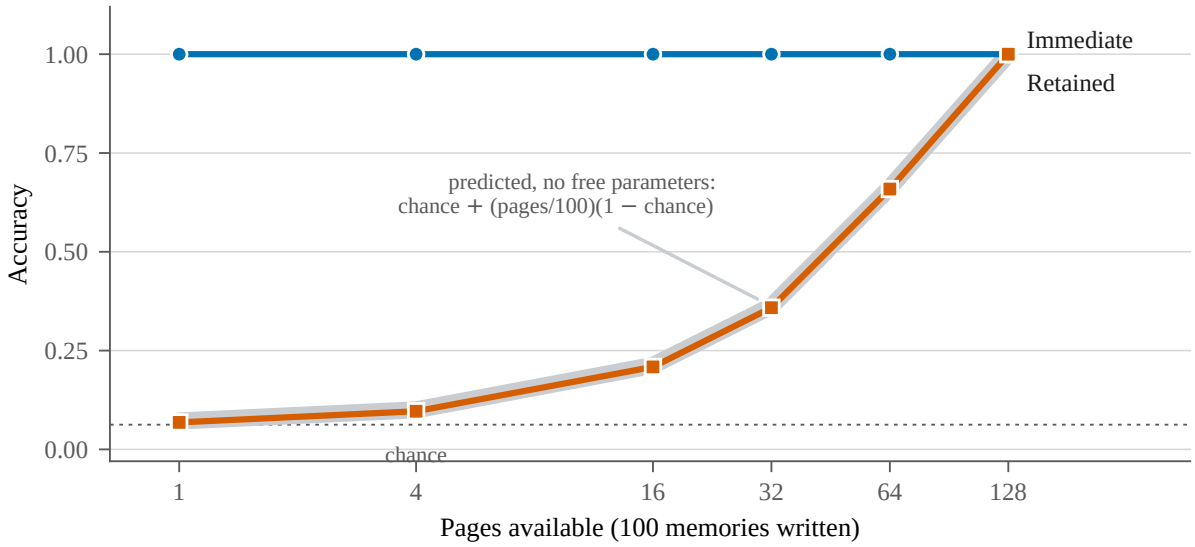

Figure 2: The central measurement. Only the number of available pages changes; the architecture, objective, optimiser, pretrained weights and task stream are identical at every point. Immediate accuracy (blue) is 1.000 at every setting. Retained accuracy (orange, $\pm$ s.d. over seeds) moves over almost the full range. The pale band is not a fit: it is chance $+ \min(P, 100)/100 \cdot (1 - \text{chance})$, the accuracy expected if a memory is retained exactly when it still holds a page, and has no free parameters. Maximum absolute deviation 0.0039 over 6 settings.

Table 2: The capacity sweep behind Figure 2. One hundred memories are written in every row; only the number of available pages changes. "Alloc." and "Evicted" are read from each run's own telemetry, not inferred from accuracy. The predicted column is chance $+ \min(P, 100)/100 \cdot (1 - \text{chance})$ and has no fitted parameters.

Pages	Seeds	Alloc.	Evicted	Immediate	Retained	Predicted	|err|
1	3	1	99	1.000	0.068 ± 0.005	0.072	0.0036
4	3	4	96	1.000	0.096 ± 0.005	0.100	0.0035
16	5	16	84	1.000	0.209 ± 0.003	0.212	0.0038
32	2	32	68	1.000	0.359 ± 0.003	0.362	0.0039
64	2	64	36	1.000	0.659 ± 0.002	0.662	0.0037
128	3	100	0	1.000	1.000 ± 0.000	1.000	0.0000

never having been written, because its page now carries another memory's modulation. Surviving pages, meanwhile, show no degradation at all: if they did, the shortfall would grow with the number of neighbours competing for space, and it does not move.

4.3 Ablations

Table 3 varies everything else we could vary. The write budget and the write gradient's order barely matter: three write passes instead of ten costs 0.026, and a first-order write gradient costs 0.028. Doubling the page size costs nothing. Against these, the capacity sweep in Table 2 spans 0.932. The components the literature tends to emphasise are the ones that move the number least. These ablation rows are one seed each, which is why we lean on the direction and the order of magnitude rather than on the third decimal place; the capacity effect they are being compared against is thirty times larger than the largest of them, which is the only comparison a single seed can support.

The exception is the cue. At $\sigma = 0.05$ the system is unaffected; at $\sigma = 0.10$ it falls to 0.214; at $\sigma = 0.20$ it is at chance. That is not a degradation curve, it is a cliff, and it is the same phenomenon as the addressing result in the

Table 3: Ablations that hold allocation fixed at 128 pages. The write budget and the order of the write gradient move retained accuracy by less than 0.03; the capacity sweep of Table 2 moves it by 0.932. The cue-noise rows are the exception and are discussed in Section 7.

Condition	Seeds	Immediate	Retained
Full system (128 pages, 10 write passes, second-order)	3	1.000	1.000
20 write passes	1	1.000	1.000
5 write passes	1	1.000	1.000
3 write passes	1	0.974	0.974
First-order write gradient	1	0.972	0.972
Page size 8 (from 4)	1	1.000	1.000
Cue noise $\sigma = 0.05$	1	1.000	1.000
Cue noise $\sigma = 0.10$	1	0.214	0.214
Cue noise $\sigma = 0.20$	1	0.065	0.065
Cue noise $\sigma = 0.60$	1	0.066	0.066

next section seen from a different side: when the cue no longer identifies the page, allocation routes writes to the wrong place and there is nothing left for the memory to be right about.

5 The problem moves to addressing

Every number to this point assumes the write reaches the right page. That assumption is doing real work, and this section removes it.

The addressing problem is to decide, from a new observation's key alone, whether it belongs to a page already held or needs a new one. Two errors are possible and they trade against each other. A *false merge* writes a new memory onto a page another memory is using, destroying it. A *missed revisit* allocates a new page for something already held, consuming capacity — which, by Section 4.2, is the same thing as losing a memory somewhere else.

Meta-training clearly helps the representation: centred keys for the same memory separate from keys for different memories with a margin of 0.897 after meta-training against 0.425 with untrained keys, more than doubling. It does not clear the frontier. The best single rule we found over 198 rules and four keying schemes still makes a false merge on 0.353 of writes and misses 0.222 of revisits, at retrieval@1 of 0.717. Figure 3 shows the whole sweep, and its shape is the finding: a frontier, not a cluster near the origin, with the arms differing in where they sit on it rather than in how close they get.

Addressing in a multi-tenant language model. We also ran the addressing path inside a language model on a task where many customers ask the same question and require different answers — a setting in which a false merge is not a lost point of accuracy but one customer being shown another customer's data. Handed the correct page, a 0.67M-parameter base model answers 0.694 of queries at eight tenants, against 0.389 for putting the rules in the prompt; with learned addressing it answers 0.444. The gap between those two is the read landing on the wrong page, and every such miss is a *leak* rather than a blank, because the page it lands on belongs to another tenant holding a conflicting value.

We initially reported this as addressing sitting at chance and attributed it to a base model whose representations cannot separate one tenant's name from another. Measuring routing directly rather than inferring it from end-task accuracy shows that was wrong: top-1 routing accuracy is 0.444 against a chance rate of 0.042, ten times chance. The keys carry tenant identity; what they lack is precision, which is a different problem with different remedies. We record the correction because inferring a mechanism from an end-to-end number is exactly the error this paper is otherwise arguing against, and we made it.

Two further results bound what addressing alone can buy. Supplying the correct page by an explicit key — which is what a deployment actually has, since a multi-tenant request carries its tenant — still leaks 0.306 at eight tenants, so partitioning the store is necessary and nowhere near sufficient; the residue is the frozen model answering from its own prior having never read the neighbour's page. And the only intervention that moved the number in an

Figure 3: Content addressing without an oracle. Each point is one match rule under one keying scheme, over three seeds and a 60-memory stream; 198 rules in total. Down and left is better and the origin is the oracle the rest of this paper assumes. The arms trade one error against the other along a frontier; none of them removes either error.

eight-seed sweep was page *width* ($+0.203$, 3.8 standard errors), which buys accuracy out of the per-tenant state budget that is the reason to prefer pages in the first place.

6 What it costs

Figure 4 is why the computed write is in the system even though it is not what produces retention. A page holds a correction to a frozen network: 1,164 bytes per memory, against 56,448 for the episodic store that matches its retention, a $48\times$ difference at equal-or-better accuracy. The systems below page memory on that axis — the recurrent state at 10 bytes, LoRA at 573 — are cheaper because they store almost nothing, and they retain at chance.

The scaling argument follows from allocation rather than from any measurement of ours, so we state it as arithmetic and not as a result: a page is a fixed-size object per memory, so state grows linearly in the number of memories with a small constant, while any approach that keeps raw context grows with the data and any approach that keeps a per-tenant adapter grows with the model. At $1{,}000$ tenants and 50 rules each on a 7B-class model, the same arithmetic gives 115.2 KB per tenant for pages against 58.7 MB for a per-tenant LoRA adapter. We label this arithmetic rather than evidence because that is what it is; the accuracy those approaches would achieve is exactly the thing the previous section says we could not measure.

7 Limitations, and where this fails

We state these at the same resolution as the results.

We lose on class-incremental vision. On VISION, the episodic store retains 0.994 and page memory retains 0.697. This is not close and it is not noise. Class-incremental few-shot classification is close to the ideal case for nearest-neighbour over stored support examples, and a page that holds a correction has nothing better to offer. We include the benchmark because it is a benchmark we lose on. The domain-incremental variant, where the classes are fixed and the imaging conditions shift, is the shape that suits a page, and there the two are level while full fine-tuning

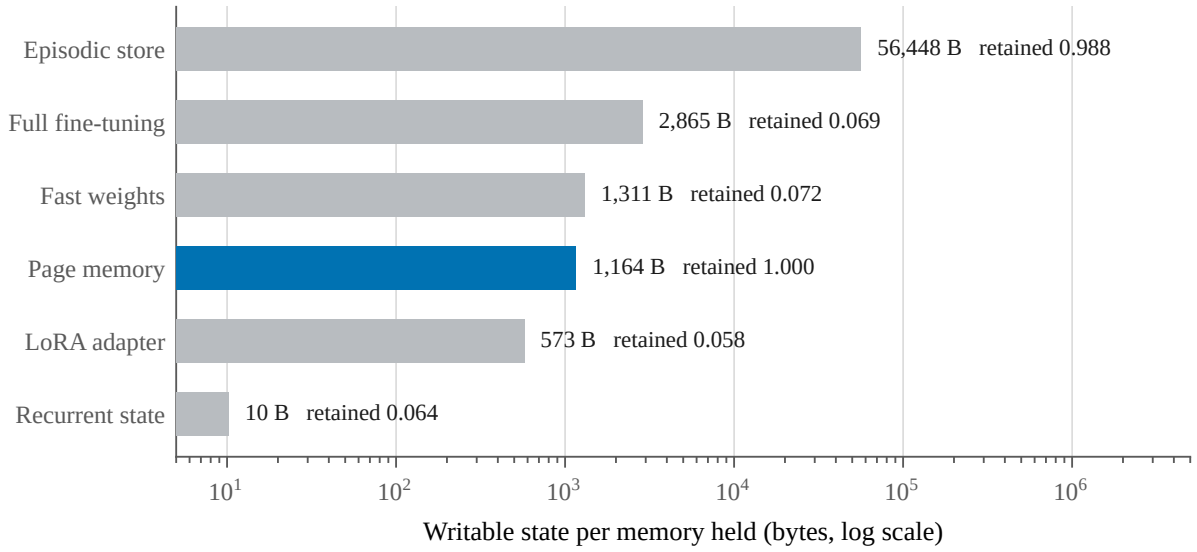

Figure 4: Writable state per memory held on the 100-task stream, log scale, with retained accuracy carried as a direct label rather than as a second axis. Cheap-and-wrong and expensive-and-right are not points on a common frontier, and drawing them against two scales would imply they were.

falls to 0.660 and 0.571 across two seeds.

Addressing is unsolved, and it is the whole problem. Section 5 is the honest summary: given the page, we are exact; asked to find the page, we are on a frontier a long way from the origin, and on a small base model we are at chance. Every headline number in this paper is conditional on an oracle we do not have.

The cue cliff. Retention collapses between cue noise 0.05 and 0.20. A deployment whose cues are noisier than that gets nothing from this system, and the failure is silent: the immediate accuracy stays high while the retained accuracy goes to chance.

Scale. Every result here is on small models and synthetic streams, with two to five seeds per cell. The mechanism we identify — retention is bounded by allocation — is architectural and we expect it to survive scale, but we have not shown that, and the addressing result gives a concrete reason to expect the *numbers* to change with a better base model even if the mechanism does not.

Streams are i.i.d. across tasks. SYMBOL MAP draws each task independently, so there is nothing to transfer and retention is cleanly measurable. That is what makes it a good instrument and also what makes it unrepresentative: real streams have structure that a good allocator should exploit, and ours does not try to.

8 Conclusion

The component of a writable neural memory that produces retention is not the learned write. It is allocation. We showed this three ways: by dissociating the two factors in systems that have one and not the other; by sweeping allocation capacity alone across 128× and watching retention move over almost its full range while immediate accuracy does not move at all; and by showing that retained accuracy is predicted to 0.0039 by page survival with no free parameters.

The practical consequence is a reallocation of attention. Effort spent on more expressive write rules is, on this evidence, spent on the part of the system that was not limiting. The part that is limiting is deciding where a memory belongs, and Section 5 is a measurement of how far from solved that is: a frontier we do not clear at any setting, and chance-level routing on a base model too small to tell two names apart. We would rather publish the boundary in that shape than a result that assumes the oracle and does not say so.

Reproducibility. All figures, tables, and every number quoted in this paper are generated from the recorded run artefacts by a single script, so a number that cannot be regenerated is a number the paper does not contain. See Appendix A.

References

Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems*, 2016.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, 2022.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

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

Donald O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, 1949.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

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, 2017.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex. *Psychological Review*, 102(3):419–457, 1995.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, 2015.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations*, 2015.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022.

A Reproduction

The run records live in `results/`, one directory per run, each with a `summary.json` carrying its configuration, metrics, telemetry and environment. The paper's figures, its three tables and its inline numbers are all produced by

```
python paper/neurips/figures.py
```

which writes `paper/neurips/figures/*.pdf` and `paper/neurips/generated/*.tex`. The macro file it emits is `\input` by `main.tex`, so a value quoted in the prose cannot drift from the run that produced it.

One trap, recorded because we fell into it. `results/` stores ablation runs beside the headline runs in identically-named directories, distinguished only by a non-empty `tag` field inside the JSON. Selecting runs by directory name averages the ablations into the main condition. Doing exactly that produced an early draft figure showing page memory retaining 0.621, contradicting a table on the same page. The selection helper in `figures.py` filters to the empty tag, and the capacity sweep reads its page count from `telemetry.n_pages` — what the run actually used — rather than parsing it out of the tag string.