TOWARD SELF-REFERENTIAL CONTINUAL LEARNING VIA A CREB-INSPIRED SELECTIVE CONSOLIDATION MECHANISM

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models that keep learning after deployment must absorb new rules without eroding what they already know. We study this problem in two formal settings: Lean 4 theorem proving across mathematical sub-disciplines, and code generation across Python, C++, and Rust. Each domain introduces counterfactual rules that contradict pre-trained priors. Continuous updates, including prioritized experience replay, keep writing into parameter directions that earlier domains rely on, so interference grows with the length of the stream. We propose Self-Referential Continual Learning (SRCL), which separates *when* an update becomes permanent from *where* it is stored. A transient working-memory adapter absorbs each input; a gate, loosely inspired by CREB-dependent memory consolidation, commits the adapter to long-term memory only when the current gradient agrees with the model's own recent gradient history; and each commit is projected onto input directions orthogonal to all previous commits. We show that the resulting interference on earlier inputs is bounded independently of stream length. On Llama-3.1-8B and Llama-3.3-70B, SRCL improves counterfactual reasoning accuracy over prioritized replay by 16.3/15.4 points (math/code) at 8B and by 12.1/13.3 points at 70B, reduces backward transfer to -1.2% and -0.4%, and holds measured interference at 0.239 after 50 steps, compared with 0.855 under continuous updates. The same self-reference defines the method's limit: our analysis predicts that the gate rejects incoherent corruption at any rate but consolidates coherent corruption beyond a critical rate, which we derive in closed form, and a 51-point contamination sweep locates the resulting collapse.

1 INTRODUCTION

Language models are increasingly expected to keep learning after deployment: a proof assistant must absorb a new library of definitions, and a code model must follow a revised API. Supplying the new rules in the context window is one option, but every query must then carry the full and growing rule set, and how models weigh in-context evidence against conflicting parametric knowledge depends strongly on how that evidence is presented (Xie et al., 2024). Updating the weights instead reintroduces catastrophic forgetting (McCloskey and Cohen, 1989; Parisi et al., 2019), which remains a central obstacle for fine-tuned language models (Biderman et al., 2024). We study this tension in formal domains, where new rules can be checked mechanically and forgetting can be measured directly: Lean 4 theorem proving across mathematical sub-disciplines (de Moura and Ullrich, 2021) and code generation across Python, C++, and Rust.

Standard remedies write every incoming example into the weights. Sequential fine-tuning does so uniformly; prioritized experience replay (PER) (Schaul et al., 2016) changes which examples are written and how strongly, but not where. Both update the parameter directions that earlier domains rely on, so interference accumulates with the length of the stream (Figure 2). Two decisions are made jointly and implicitly: *when* an update should become permanent, and *where* in parameter space it should be stored. Subspace methods answer the second question by reserving orthogonal directions for each task (Farajtabar et al., 2020; Saha et al., 2021; Wang et al., 2023; Liang and Li, 2024), but they rely on known task boundaries to answer the first.

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.

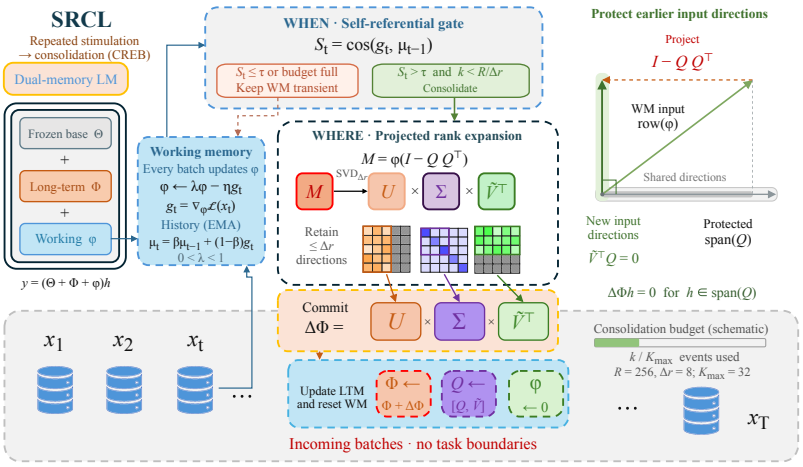

Figure 1: **SRCL overview.** A self-referential gate decides when to consolidate working memory; projected rank expansion decides where to store it while protecting earlier input directions.

We propose Self-Referential Continual Learning (SRCL), which answers both questions without task boundaries (Figure 1). Every input is first absorbed by a transient working-memory (WM) adapter. A gate commits the WM to a persistent long-term memory (LTM) only when the current gradient agrees with the model's own recent gradient history—that is, when the same update direction has been requested repeatedly. The design is loosely inspired by CREB-dependent consolidation, in which converting a short-term synaptic change into long-term memory requires repeated stimulation to activate a transcriptional program, while isolated events remain transient (Kandel, 2001). When the gate opens, the WM is projected onto input directions orthogonal to all previous commits and added to the LTM as a small rank expansion.

Separating the two decisions has a simple consequence (Section 3.5): the interference that consolidation causes on an earlier input is bounded by the rank budget and by the part of that input lying outside the protected subspace, independently of stream length. Empirically, on Llama-3.1-8B and Llama-3.3-70B, SRCL improves counterfactual reasoning accuracy over continuous PER by 16.3/15.4 points (math/code) at 8B and by 12.1/13.3 points at 70B, while reducing backward transfer from -8.4% to -1.2% and from -5.2% to -0.4% (Table 1). At the selected threshold, SRCL uses 48 of 256 available LTM ranks and keeps measured interference at 0.239 after 50 steps.

Because the gate's reference is the model's own gradient history, it also has a predictable failure mode. Corruption whose gradients are incoherent averages out of the reference and is rejected at any rate. Coherent corruption, in contrast, becomes part of the reference once it is frequent enough; beyond a critical rate $p^\star(\tau)$, which we derive in closed form (Proposition 2), the gate consolidates it and exhausts the rank budget. A 51-point contamination sweep locates this boundary empirically (Section 5.5).

Our contributions are:

- **Task-free selective consolidation.** SRCL separates *when* to consolidate (a self-referential gradient-consistency gate) from *where* to consolidate (projected rank expansion), without task boundaries or a replay buffer.
- **Analysis.** We bound consolidation interference independently of stream length (Proposition 1) and derive the critical contamination rate at which a self-referential gate fails (Proposition 2).
- **Evidence.** Across two formal domains and two model scales, with ablations that isolate the gate and the projection, SRCL learns new counterfactual rules while preserving earlier domains; a contamination sweep maps the predicted failure boundary.

2 Related Work

Forgetting in continual fine-tuning. Sequential fine-tuning erodes abilities that language models acquired earlier, and LoRA forgets less than full fine-tuning but also learns less (Biderman et al., 2024). Part of the measured loss reflects disrupted task alignment rather than lost knowledge (Zheng et al., 2025), forgetting tracks how far fine-tuning moves the model's distribution from its base (Shenfeld et al., 2026), learning dynamics explain how an update on one example changes predictions on others (Ren and Sutherland, 2025), and continued pre-training on math and code acts mainly by rotating singular vectors rather than rescaling them (Borodin et al., 2026). Classical remedies regularize important parameters (Kirkpatrick et al., 2017), constrain updates with episodic memories (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2019), or replay stored data, for example with priorities as in reinforcement learning (Schaul et al., 2016). Networks that keep training also lose plasticity (Dohare et al., 2024); in SRCL this appears as a finite consolidation budget.

Subspace methods and null-space editing. Orthogonal gradient descent (Farajtabar et al., 2020) and gradient projection memory (Saha et al., 2021) protect subspaces used by past tasks, and O-LoRA (Wang et al., 2023) and InfLoRA (Liang and Li, 2024) carry this idea to low-rank adapters; SD-LoRA (Wu et al., 2025) instead decouples the magnitude and direction of LoRA components. In model editing, MEMIT (Meng et al., 2023) writes batches of facts into MLP weights, and AlphaEdit (Fang et al., 2025) projects each edit onto the null space of preserved keys, which provably leaves outputs on those keys unchanged. SRCL applies the same principle to its long-term memory and adds a way to decide when to open new directions without task boundaries or edit requests.

Deciding when to consolidate. Task-free continual learning (Aljundi et al., 2019) removes task boundaries by consolidating parameter-importance estimates whenever the training loss plateaus, a loss-based criterion of the kind we test in Section 5.2. Dual-memory methods separate fast from slow learning: Progress & Compress (Schwarz et al., 2018) distills an active network into a protected knowledge base at each task boundary, and CLS-ER (Arani et al., 2022) keeps fast and slow moving-average copies of the model alongside episodic replay, both following complementary learning systems theory (McClelland et al., 1995). In the brain, CREB-dependent transcription gates the conversion of repeated short-term changes into long-term memory (Kandel, 2001), and the resulting memory traces, or engrams (Josselyn and Tonegawa, 2020), have recently been isolated and edited in trained networks (Kwon et al., 2026). Language models also carry signals about their own uncertainty (Kadavath et al., 2022; Farquhar et al., 2024; Kossen et al., 2024); SRCL gates instead on whether its own update direction recurs. In-Place TTT (Feng et al., 2026) turns the MLP output projections of a language model into fast weights updated on the incoming stream, a transient memory whose updates such a gate could select.

Parametric versus in-context knowledge, and self-generated data. Models adopt coherent in-context evidence that contradicts their parametric knowledge but show strong confirmation bias when the evidence is mixed (Xie et al., 2024), and for a stream of rules the context must grow with every rule; Appendix C.1 compares the two options. Self-training consolidates model-generated reasoning (Zelikman et al., 2022; Madaan et al., 2023; Kumar et al., 2024), and recursive training on unfiltered generated data degrades the learned distribution (Shumailov et al., 2024). Our contamination analysis characterizes when a self-referential filter admits systematic errors, although we do not run SRCL inside such a loop.

3 Method

3.1 Setting

We observe a stream $\mathcal{S} = (x_1, \ldots, x_T)$ of training batches drawn from a sequence of domains, without domain labels or boundaries. A set $\mathcal{W}$ of weight matrices of a frozen base model is adapted. For each matrix with frozen value $\Theta \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, SRCL maintains a long-term memory Φ and a working memory ϕ of the same shape, and the layer computes

$$y = (\Theta + \Phi + \phi)\, h, \qquad h \in \mathbb{R}^{d_{\text{in}}}. \tag{1}$$

The WM ϕ is a LoRA module (Hu et al., 2022) of rank r_{WM}. The LTM Φ is a sum of at most $K_{\max} = R/\Delta r$ consolidation events of rank at most Δr each, so its rank never exceeds the budget

R. We describe a single matrix; the gate is shared across $\mathcal{W}$. At evaluation the WM is cleared, so the LTM carries everything the model retains.

3.2 Working memory

Each batch updates the WM by a decayed gradient step,

$$\phi \leftarrow \lambda\phi - \eta\, g_t, \qquad g_t = \nabla_\phi \mathcal{L}(x_t; \Theta + \Phi + \phi), \tag{2}$$

with decay $\lambda \in (0, 1)$. The WM therefore tracks recent inputs, and anything that is not consolidated fades geometrically.

3.3 When to consolidate: a self-referential gate

Definition 1 (Gradient-consistency score)**.** *Let $\mu_t = \beta\mu_{t-1} + (1-\beta)\, g_t$, with $\mu_0 = 0$, be an exponential moving average of WM gradients, concatenated across all adapted matrices. The score of batch x_t is*

$$S(x_t) = \frac{\langle g_t, \mu_{t-1}\rangle}{\|g_t\|_2\, \|\mu_{t-1}\|_2} \in [-1, 1]. \tag{3}$$

A high score means that the current input requests an update the model has recently and repeatedly requested. A single idiosyncratic input, or corrupted inputs whose gradients point in unrelated directions, produce low scores. The WM is consolidated when $S(x_t) > \tau$. Because the reference μ is built from the model's own gradients, the gate needs no task labels, and a new domain becomes eligible for consolidation once its update direction has recurred often enough to dominate μ—the computational counterpart of the repeated stimulation required for CREB-dependent consolidation.

3.4 Where to consolidate: projected rank expansion

Definition 2 (Projected consolidation)**.** *Let $Q \in \mathbb{R}^{d_{in} \times m}$ have orthonormal columns spanning the input directions used by previous consolidations (Q is empty initially). When the gate opens and fewer than $K_{\max}$ events have occurred, SRCL computes the rank-Δr truncated SVD*

$$U\Sigma\tilde{V}^\top = \mathrm{SVD}_{\Delta r}\big(\phi\,(I - QQ^\top)\big) \tag{4}$$

and sets $\Phi \leftarrow \Phi + U\Sigma\tilde{V}^\top$, $Q \leftarrow [Q,\ \tilde{V}]$, and $\phi \leftarrow 0$.

Projecting before the SVD makes $U\Sigma\tilde{V}^\top$ the best rank-Δr approximation of the projected update. The row space of $\phi(I - QQ^\top)$ is orthogonal to $\mathrm{span}(Q)$, so the new columns $\tilde{V}$ are orthonormal and orthogonal to Q, and for any input $h \in \mathrm{span}(Q)$ every later consolidation leaves Φh unchanged. The price is that the LTM can no longer change its response within $\mathrm{span}(Q)$; it can only write along new input directions. Singular values below a numerical tolerance are dropped, so an event may add fewer than Δr directions; the budget counts events.

Protection of an earlier input depends on how much of it lies in $\mathrm{span}(Q)$, which we measure by the relative residual $\|(I - QQ^\top)h\|_2 / \|h\|_2$. Under the dense update of Eq. (2), the rows of ϕ are combinations of the activations that produced it, so Q collects dominant input directions of consolidated data. With a LoRA WM $\phi = BA$, the input directions are the rows of A, which track the activations only if A is initialized in, or constrained to, their span, as in InfLoRA (Liang and Li, 2024). Once $K_{\max}$ events have occurred, the gate is disabled and the WM remains transient.

Algorithm 1 in Appendix A.5 summarizes the procedure. Both the projection and the SVD are computed on the LoRA factors at cost $O\big((d_{\text{in}} + d_{\text{out}})\, r_{\text{WM}}^2 + r_{\text{WM}}\, d_{\text{in}}\, m\big)$ per matrix, and only when the gate opens.

3.5 Analysis

We first bound how much consolidation can disturb earlier inputs; proofs are in Appendix A.2. Let $\Phi^{(k)}$ denote the LTM after k consolidation events, $\Delta\Phi_k = U_k\Sigma_k\tilde{V}_k^\top$ the k-th event, Q_k the protected basis after it, and $\sigma_{\max} = \max_k \|\Sigma_k\|_2$.

Proposition 1 (Consolidation interference)**.** *For any input h and event index k_0, let $r_{k_0}(h) = \|(I - Q_{k_0}Q_{k_0}^\top)h\|_2$. Then for every $K \geq k_0$,*

$$\left\|(\Phi^{(K)} - \Phi^{(k_0)})\, h\right\|_2 \;\leq\; \sigma_{\max}\sqrt{K - k_0}\; r_{k_0}(h) \;\leq\; \sigma_{\max}\sqrt{R/\Delta r}\; r_{k_0}(h). \tag{5}$$

The bound does not depend on the stream length T, and if $h \in \mathrm{span}(Q_{k_0})$ the LTM response to h is preserved exactly. In contrast, consider unconstrained updates $\Phi_t = \Phi_{t-1} - \eta w_t G_t$ with weights $w_t \geq 0$ and dense gradients G_t, let $D_T = (\Phi_T - \Phi_{t_0})\, h$, and let $\mathcal{F}_{t-1}$ denote the history up to step $t-1$. For all $t > t_0$:

(a) if $\mathbb{E}[G_t h \mid w_t] = m$ with $\|m\|_2 = c > 0$, then $\mathbb{E}\|D_T\|_2 \geq \eta c \sum_{t>t_0} \mathbb{E}[w_t]$;

(b) if $\mathbb{E}[w_t G_t h \mid \mathcal{F}_{t-1}] = 0$ and $\mathbb{E}[\|G_t h\|_2^2 \mid w_t] \geq c^2$, then $\mathbb{E}\|D_T\|_2^2 \geq \eta^2 c^2 \sum_{t>t_0} \mathbb{E}[w_t^2]$.

With $w_t \geq w_{\min} > 0$, drift on h therefore grows linearly in T when later gradients act on h systematically (a), and as a random walk when they are unbiased with respect to h (b); only the projected, budgeted LTM stays bounded. Proposition 1 separates the roles of the two components. The projection controls how much each event can disturb an earlier input, through the residual $r_{k_0}(h)$; the gate controls how many events occur and which directions enter Q. Consolidating too often exhausts the budget and fills Q with directions of little value, while consolidating too rarely leaves new domains in the transient WM. If all adapted layers of an earlier input lie in the protected subspace, the network output on that input is preserved exactly; see Appendix A.2. With gradient clipping at norm G, $\sigma_{\max} \leq \eta G/(1-\lambda)$.

The gate itself can fail. We analyze a stylized stream in which each element—the unit the gate scores, a batch in our experiments—is corrupted with probability p and regular otherwise. Regular and corrupted WM gradients have mean directions $a \perp b$ with magnitudes α_s and α_c, plus zero-mean fluctuations; let $\kappa = \alpha_s/\alpha_c$ and let $\rho = \alpha_c/(\alpha_c^2 + \mathbb{E}\|\xi_c\|_2^2)^{1/2} \in (0, 1]$ be the coherence of corrupted gradients, where ξ_c is their fluctuation.

Proposition 2 (Critical contamination rate)**.** *In the slow-average, high-dimensional limit of Appendix A.3, the score of a corrupted element concentrates at*

$$S_c(p) = \frac{\rho\, p}{\sqrt{(1-p)^2\kappa^2 + p^2}}. \tag{6}$$

Hence (i) if $\rho \leq \tau$, corrupted elements never trigger consolidation, at any rate p; and (ii) if $\rho > \tau$, they trigger consolidation if and only if

$$p > p^\star(\tau, \kappa, \rho) = \frac{\tau\kappa}{\tau\kappa + \sqrt{\rho^2 - \tau^2}}, \tag{7}$$

which increases with τ and κ and decreases with ρ.

Two predictions follow. Randomized corruption, whose gradients do not share a direction ($\rho \leq \tau$), should be rejected at every rate, whereas templated corruption should be consolidated beyond $p^\star$. For $\tau = 0.6$ and $\kappa = \rho = 1$, $p^\star \approx 0.43$. The model is stylized: it treats the moving average as the stream mean, ignores finite-sample fluctuations, and assumes that each scored batch is either entirely regular or entirely corrupted (Appendix A.3). Section 5.5 tests both predictions with κ and ρ measured on batch gradients.

4 Experimental Setup

4.1 Streams and counterfactual probes

Sequential formal math. Lean 4 (de Moura and Ullrich, 2021) tactic-generation traces from five mathematical sub-disciplines, presented one after another. **Sequential code.** Programming traces moving from Python to C++ to Rust. Each domain spans 50 stream steps of 32 traces and is seen once (Appendix B).

Each domain introduces counterfactual rules: axioms in Lean that contradict familiar arithmetic, or altered library semantics in code. We measure reasoning with *Counterfactual Reasoning Probes*

(CRPs), held-out problems that can be solved only by applying the domain's new rules and on which the pre-trained prior gives a wrong answer. Math CRPs are graded by the Lean kernel, and every axiom set comes with an explicit model that certifies its consistency; code CRPs are graded by hidden tests in a runtime that implements the counterfactual semantics (Appendix B.2). Except for the in-context baselines of Appendix C.1, models receive the probe without the rules.

4.2 Models

We use Llama-3.1-8B (Dubey et al., 2024) for all analyses and ablations and Llama-3.3-70B-Instruct for the main comparison. To test generality across architectures, we add Mistral-NeMo-12B, Qwen2.5-14B (Qwen Team, 2024), and the mixture-of-experts model DeepSeek-V2-Lite (DeepSeek-AI, 2024) (15.7B parameters, 2.4B active). GPT-4o and Gemini 1.5 Pro, given all rules in context, are compared in Appendix C.1.

4.3 Baselines

All methods adapt the same matrices with adapters of the same rank, share the learning-rate grid, and see the stream once (Appendix C.2). **Reference points.** The *base model* without updates, and *joint training* on all domains at once, which is not a continual method but shows what the adapter can learn without a stream. **Task-free baselines.** *Sequential SFT* fine-tunes on the stream in order. *Delayed large-batch SFT* accumulates stream data into large batches before each update, controlling for the stabilizing effect of batching. *Continuous PER* keeps a replay buffer, samples with a priority that increases with $S(x)$, and applies standard updates; it uses SRCL's signal without its gate or projection. **Boundary-dependent baselines.** *EWC* (Kirkpatrick et al., 2017) estimates the Fisher information at each domain boundary; *O-LoRA* (Wang et al., 2023) and *InfLoRA* (Liang and Li, 2024) open a new orthogonal adapter subspace at each boundary. All three receive the true (oracle) boundaries.

4.4 Metrics

CRP accuracy is pass@1 with greedy decoding, averaged over the domains of a stream and measured after the full stream. **Backward transfer** is reported per stream as $\mathrm{BWT} = \frac{1}{N-1}\sum_{i<N}(A_{N,i} - A_{i,i})$, where $A_{j,i}$ is CRP accuracy on domain i after training through domain j and N is the number of domains. **LTM rank** counts consolidated directions out of $R = 256$, and the **trigger rate** is the fraction of steps with $S(x) > \tau$. **Interference** ϵ is the relative drift of adapted-layer responses on probes of earlier domains,

$$\epsilon_t = \mathbb{E}_{x \in \mathcal{P}_{\text{old}}} \mathbb{E}_{W \in \mathcal{W}} \frac{\|(\Phi_t - \Phi_{t_x})\, h_W(x)\|_2}{\|(\Theta + \Phi_{t_x})\, h_W(x)\|_2}, \tag{8}$$

where t_x is the end of the domain of probe x and $h_W(x)$ is the input to matrix W; for baselines, Φ is the adapter. Gate diagnostics are defined in Appendix D.1.

4.5 Protocol

All conditions use three seeds (42, 1337, 2026); we report mean $\pm$ standard deviation. Every method is evaluated with its full state at the end of each domain—for SRCL, $\Theta + \Phi + \phi$—so knowledge that is never consolidated counts as learned while it is held in the WM and as forgotten once it decays. Hyperparameters, including τ, are selected on validation CRPs disjoint from the test CRPs. Experiments run on 8×NVIDIA H100 (80GB) GPUs (Appendix D.2).

5 Results

5.1 Main results

Table 1 compares SRCL with reference points and baselines at both scales. Sequential SFT learns each domain but loses most of it later (math BWT -42.1% at 8B and -31.4% at 70B). EWC and delayed large-batch updates reduce forgetting only partially, and continuous PER, which replays stored data, brings BWT to -8.4% (math) and -6.3% (code) at 8B. SRCL is the strongest task-free

Table 1: **Main results** after the full stream (mean ± std over 3 seeds; the base model is deterministic under greedy decoding). CRP: counterfactual reasoning probe accuracy (%), averaged over domains. BWT: backward transfer (%; 0 means no forgetting). Task-free: needs no domain boundaries. Data-free: stores no past training data. EWC, O-LoRA, and InfLoRA receive oracle boundaries; joint training sees all domains at once. The top three values in each metric are shown in red, orange, and yellow, respectively. .

			Llama-3.1-8B				Llama-3.3-70B			
	Task-	Data-	Math		Code		Math		Code	
Method	**free**	**free**	**CRP**	**BWT**	**CRP**	**BWT**	**CRP**	**BWT**	**CRP**	**BWT**
Reference Models										
Base model (no update)	–	–	6.5	–	5.2	–	11.8	–	9.7	–
Joint training (offline)	–	–	**74.6 ± 0.7**	–	**71.9 ± 0.8**	–	**89.7 ± 0.5**	–	**88.4 ± 0.6**	–
Task-free Baselines										
Sequential SFT	✓	✓	28.4 ± 2.1	−42.1 ± 3.5	24.2 ± 1.8	−41.9 ± 3.3	51.1 ± 1.8	−31.4 ± 2.8	48.5 ± 2.2	−29.6 ± 2.5
Delayed large-batch SFT	✓	×	42.5 ± 1.4	−18.2 ± 1.9	38.8 ± 1.6	−16.4 ± 1.8	62.8 ± 1.1	−12.5 ± 1.4	59.4 ± 1.3	−11.6 ± 1.3
Continuous PER	✓	×	52.1 ± 1.2	−8.4 ± 1.1	49.7 ± 1.5	−6.3 ± 1.0	74.2 ± 0.9	−5.2 ± 0.8	71.5 ± 1.1	−4.4 ± 0.7
Boundary-dependent Baselines										
EWC	×	✓	36.9 ± 1.7	−27.6 ± 2.4	33.1 ± 1.9	−28.9 ± 2.6	58.3 ± 1.4	−20.4 ± 1.9	55.2 ± 1.6	−21.1 ± 2.0
O-LoRA	×	✓	61.2 ± 1.1	**−3.9 ± 0.7**	58.4 ± 1.2	**−3.1 ± 0.6**	80.1 ± 0.8	**−2.2 ± 0.5**	77.9 ± 0.9	**−1.8 ± 0.4**
InfLoRA	×	✓	**67.6 ± 0.9**	**−1.0 ± 0.3**	**65.7 ± 1.0**	**−0.7 ± 0.3**	**85.4 ± 0.6**	**−0.5 ± 0.2**	**84.1 ± 0.7**	**−0.4 ± 0.2**
Proposed Method										
SRCL (ours)	✓	✓	**68.4 ± 0.8**	**−1.2 ± 0.3**	**65.1 ± 0.9**	**−0.8 ± 0.3**	**86.3 ± 0.5**	**−0.4 ± 0.1**	**84.8 ± 0.6**	**−0.6 ± 0.2**

Table 2: **Across architectures** (Math / Code, mean over 3 seeds; additional baselines in Appendix C.2). InfLoRA receives oracle boundaries. For each architecture and domain, the best and second-best CRP values are shown in red and orange, respectively. .

	CRP (%)			SRCL BWT (%)
Model	**Continuous PER**	**InfLoRA (oracle)**	**SRCL**	
Open-weight Architectures				
Llama-3.1-8B	52.1 / 49.7	**67.6** / **65.7**	**68.4** / **65.1**	−1.2 / −0.8
Mistral-NeMo-12B	54.6 / 51.4	**68.1** / **63.9**	**69.0** / **64.2**	−1.5 / −1.1
Qwen2.5-14B	58.4 / 55.2	**72.9** / **69.4**	**72.5** / **68.9**	−1.0 / −0.8
DeepSeek-V2-Lite (MoE)	50.2 / 47.8	**59.8** / **55.1**	**60.7** / **54.1**	−2.6 / −2.2
Llama-3.3-70B	74.2 / 71.5	**85.4** / **84.1**	**86.3** / **84.8**	−0.4 / −0.6

method on every metric: it improves CRP over PER by 16.3/15.4 points (math/code) at 8B and 12.1/13.3 points at 70B and keeps BWT near -1%, without storing data or receiving boundaries. It performs on par with InfLoRA, which receives oracle boundaries: SRCL is 0.8–0.9 points higher on math, 0.6 points lower on code at 8B and 0.7 higher at 70B, while InfLoRA forgets slightly less at 8B ($-1.0/-0.7$ vs. $-1.2/-0.8$). Two gaps remain. SRCL stays 6.2/6.8 points below joint training at 8B, a gap that narrows to 3.4/3.6 points at 70B, and at 70B it forgets more on code than on math.

The same pattern holds across architectures (Table 2). SRCL improves over PER on every model, by 10.5–16.3 points on math and 6.3–15.4 on code, and is within one point of InfLoRA in every comparison. The gains are smallest and the remaining forgetting is largest on the mixture-of-experts model DeepSeek-V2-Lite ($+10.5/+6.3$ over PER, BWT $-2.6/-2.2$); related scaling results appear in Appendix C.2.

5.2 When and where: ablations

Table 3 separates *when* from *where*. Holding the number of events fixed at six, consolidating at evenly spaced steps instead of when the gate opens costs 8.1 points of accuracy but leaves interference unchanged (0.238 vs. 0.239): timing does not change how much each event disturbs earlier inputs, but it determines what is stored. Removing the projection while keeping the gate costs 6.7 points, nearly

Table 3: **When versus where** on Llama-3.1-8B math (mean over 3 seeds). Periodic and gate-based variants use six consolidation events; Proj. denotes projected consolidation. .

When	Proj.	Math CRP	BWT	Rank	ϵ
Continuous	×	51.4	−9.1	256	0.855
Every step	✓	28.8	−39.3	256	0.000
Periodic	✓	60.3	−3.8	48	0.238
Gate	×	61.7	−4.6	48	0.412
Gate	✓	**68.4**	**−1.2**	48	0.239

Table 4: **Threshold sweep** on Llama-3.1-8B math. Trigger rate is the fraction of the 250 steps with $S(x) > \tau$; $\tau = 0$ uses continuous, unprojected LTM updates. .

τ	Math CRP	Rank	ϵ	Trigger rate
0.0	51.4	256	0.855	100%
0.2	58.2	192	0.612	9.6%
0.4	64.5	114	0.384	6.0%
0.6	**68.4**	48	0.239	2.4%
0.8	42.1	12	0.115	0.8%
1.0	16.9	0	0.000	0%

Table 5: **Consolidation signals** (Llama-3.1-8B, math). Left: correlations with log token frequency, loss, and CRP-novelty labels, and AUC for detecting out-of-domain batches (definitions in Appendix D.1). Right: results when each signal drives the gate, with its threshold set to produce the same six events as $S(x)$ at $\tau = 0.6$; the random gate fires at six uniformly drawn steps. .

	Diagnostics				Used as gate	
Signal	**Corr. token freq.**	**Corr. loss**	**Corr. CRP novelty**	**OOD AUC**	**Math CRP (%)**	**BWT (%)**
Random	–	–	–	–	57.1 ± 2.0	-4.4 ± 0.9
Raw loss	−0.62	**1.00**	**0.41**	**0.68**	$\mathbf{61.2 \pm 1.2}$	$\mathbf{-2.1 \pm 0.5}$
Predictive entropy	−0.74	**0.88**	**0.38**	**0.71**	$\mathbf{59.8 \pm 1.3}$	$\mathbf{-2.4 \pm 0.5}$
Gradient consistency $S(x)$	−0.08	**0.44**	**0.92**	**0.95**	$\mathbf{68.4 \pm 0.8}$	$\mathbf{-1.2 \pm 0.3}$

quadruples forgetting (BWT -4.6% vs. -1.2%), and raises interference to 0.412. Consolidating at every step exhausts the budget within the first domain. Interference on first-domain probes is then zero, because the LTM never changes afterwards, yet later domains are held only transiently and forgotten (BWT -39.3%): low interference is necessary but not sufficient, and the gate is what lets a finite budget cover the whole stream.

The threshold trades plasticity against protection (Table 4). Lowering τ raises the trigger rate, the rank consumed, and interference, as Proposition 1 predicts through the number of events: from 2.4% of steps and $\epsilon = 0.239$ at $\tau = 0.6$ to 9.6% and $\epsilon = 0.612$ at $\tau = 0.2$, and to $\epsilon = 0.855$ under continuous updates. Raising τ to 0.8 lowers interference to 0.115 but leaves only two events, and accuracy falls to 42.1%. At $\tau = 1.0$ nothing is consolidated and the model retains essentially only the last domain, which is still held in the WM (16.9%). Accuracy therefore peaks at an intermediate threshold that balances interference against coverage of the stream.

5.3 What the gate responds to

A consolidation signal should respond to recurring structure rather than to surface rarity. Raw loss and predictive entropy are strongly anti-correlated with token frequency ($r = -0.62$ and -0.74; Table 5), so a gate built on them preferentially consolidates batches with rare tokens. The gradient-consistency score is nearly uncorrelated with token frequency ($r = -0.08$), correlates with CRP-novelty labels at 0.92, and detects out-of-domain batches with AUC 0.95 (Appendix D.1). Used as the gate with the same number of events, loss and entropy reach 61.2% and 59.8%, close to consolidating at random (57.1%) or periodic steps (60.3%, Table 3), whereas $S(x)$ reaches 68.4%. What matters for consolidation is not whether an input is surprising, but whether the model keeps requesting the same update.

5.4 Interference and consolidation dynamics

Figure 2 tracks interference on first-domain probes. Under continuous updates ϵ grows throughout the window and reaches 0.855 after 50 steps, whereas under SRCL it rises only at consolidation events and plateaus at 0.239. The bound of Proposition 1, evaluated with the measured $\sigma_{\max}$ and residuals, follows the same plateau and is conservative by a median factor of 3.4. Figure 3 shows the mechanism on a 100-step window that starts after two consolidation events. The score stays below

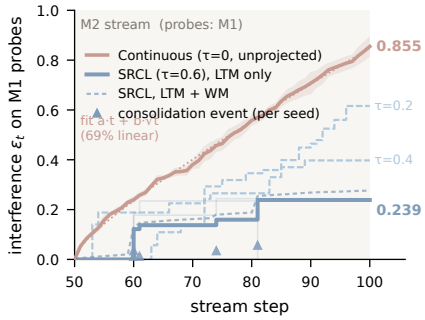

Figure 2: **Interference.** SRCL plateaus while continuous updates drift.

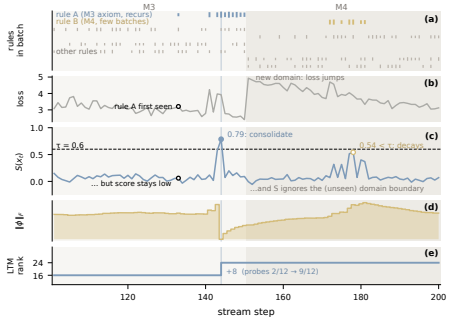

Figure 3: **Consolidation trace.** Rank grows only when the gate opens; each event resets the WM.

0.2 on most steps (the trigger rate at $\tau = 0.2$ is 9.6%); when a new axiom recurs, $S(x)$ rises to 0.79 at step 42, the WM is consolidated as a rank-8 expansion (LTM rank $16 \to 24$), and ϕ is reset. A second rise near step 78 peaks at 0.54, below τ, and the WM content decays instead of being consolidated.

5.5 The contamination boundary

We replace training traces with corrupted ones at rates $p \in [0, 0.5]$ (51 points; Appendix B.4) in the 8B math stream. With templated corruption, Math CRP stays within 2 points of the clean stream up to $p = 0.25$ and collapses between 0.30 and 0.38, with an observed boundary of $\hat{p}^\star = 0.33 \pm 0.02$. The collapse has the mechanism that Proposition 2 predicts. At $p = 0.40$ the gate fires on 95% of steps, regular and corrupted content alike, the 256-rank budget is exhausted within 50 steps (after 34 steps on average), and Math CRP falls to 24.8%, compared with 6.5% for the base model (Appendix E.2). With $\hat{\kappa} = 0.72$ and batch-level coherence $\hat{\rho} = 0.98$ (Appendix D.1), the proposition predicts $p^\star = 0.36$; the observed transition begins slightly earlier, as expected from the finite averaging window of μ. Randomized corruption, whose batch gradients share little direction ($\hat{\rho} = 0.21 < \tau$), never opens the gate at any rate; accuracy still declines gradually, to 61.2% at $p = 0.5$, because fewer valid traces remain. Across thresholds, the observed boundary lies 0.02–0.03 below the prediction and increases with τ as predicted (Table 13).

6 Conclusion and Limitations

SRCL treats continual learning as two decisions: when an update should become permanent and where it should be stored. A gate that asks whether the model keeps requesting the same update, and a projection that keeps each commit away from earlier ones, let a language model absorb new rules without task boundaries or stored data, with interference bounded independently of stream length. Because the gate refers only to the model's own gradients, its failure mode is equally explicit: it consolidates coherent contamination beyond a predictable critical rate.

Limitations. The protected basis only grows, limiting the budget to $R/\Delta r = 32$ events; retiring directions may mitigate the resulting loss of plasticity (Dohare et al., 2024; Kwon et al., 2026). The LTM cannot revise responses within $\mathrm{span}(Q)$, and our disjoint-rule streams do not test contradictions. The gate cannot identify persistent contamination; streams are short (250/150 steps), and SRCL trails joint training.

AI Use Statement

Generative AI tools were used to assist with drafting and editing portions of the manuscript, improving readability, and preparing the LaTeX source. All AI-assisted material was reviewed and verified by the authors. The authors take full responsibility for the final content of this work, including any text or artifacts produced with the aid of generative AI.

REFERENCES

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11254–11263, 2019.

Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations (ICLR)*, 2022.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. Lora learns less and forgets less. *Transactions on Machine Learning Research*, 2024.

Nikita Borodin, Maria Krylova, Artem Zabolotnyi, Dmitry Aspisov, Egor Shikov, Nikita Tyuplyaev, Oleg Travkin, Roman Alferov, and Dmitry Vinichenko. Diffract: Spectral view of LLM domain adaptation. *arXiv preprint arXiv:2608.10850*, 2026.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations (ICLR)*, 2019.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction*, pages 625–635. Springer, 2021.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632: 768–774, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AlphaEdit: Null-space constrained knowledge editing for language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3762–3773. PMLR, 2020.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. In-place test-time training. In *International Conference on Learning Representations (ICLR)*, 2026.

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 (ICLR)*, 2022.

Sheena A. Josselyn and Susumu Tonegawa. Memory engrams: Recalling the past and imagining the future. *Science*, 367(6473):eaaw4325, 2020.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Eric R. Kandel. The molecular biology of memory storage: A dialogue between genes and synapses. *Science*, 294(5544):1030–1038, 2001.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth A Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.14563*, 2024.

Aviral Kumar, Avi Singh, Justin Fu, Dibya Ghosh, Corey Riviere, Aviral Carmon, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.14674*, 2024.

Jea Kwon, Dong-Kyum Kim, Jiwon Kim, Yonghyun Kim, Woong Kook, and Meeyoung Cha. AI Engram: In search of memory traces in artificial intelligence. In *International Conference on Machine Learning (ICML)*, 2026.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647, 2024.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning (ICML)*, 2024.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

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

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

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations (ICLR)*, 2023.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yi Ren and Danica J. Sutherland. Learning dynamics of LLM finetuning. In *International Conference on Learning Representations (ICLR)*, 2025.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning (ICML)*, pages 4528–4537, 2018.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations (ICLR)*, 2026.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631:755–759, 2024.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671, 2023.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. SD-LoRA: Scalable decoupled low-rank adaptation for class incremental learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 15476–15488, 2022.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *International Conference on Learning Representations (ICLR)*, 2025.

A Theoretical Foundations and Proofs

A.1 Formalization of the Dual-Memory Topology

We analyze each adapted matrix separately and use the column-vector convention of Eq. (1): a matrix with frozen value $\Theta \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ computes

$$y = (\Theta + \Phi + \phi)\, h, \qquad h \in \mathbb{R}^{d_{\text{in}}}. \tag{9}$$

The working memory $\phi = BA$, with $B \in \mathbb{R}^{d_{\text{out}} \times r_{\text{WM}}}$ and $A \in \mathbb{R}^{r_{\text{WM}} \times d_{\text{in}}}$, is transient: it is updated by Eq. (2), decays geometrically, and is reset to zero at every consolidation. The long-term memory after k consolidation events is

$$\Phi^{(k)} = \sum_{j=1}^{k} U_j \Sigma_j \tilde{V}_j^\top, \qquad Q_k = [\tilde{V}_1, \ldots, \tilde{V}_k], \qquad P_k = I - Q_k Q_k^\top, \tag{10}$$

where $U_j \in \mathbb{R}^{d_{\text{out}} \times \Delta r_j}$ and $\tilde{V}_j \in \mathbb{R}^{d_{\text{in}} \times \Delta r_j}$, with $\Delta r_j \le \Delta r$, have orthonormal columns and Σ_j is diagonal and non-negative. Input directions of different events are orthogonal, $\tilde{V}_i^\top \tilde{V}_j = 0$ for $i \neq j$; output directions U_j are not constrained. At most $K_{\max} = R/\Delta r$ events occur, so the LTM has rank at most R.

Q_k is orthonormal. The columns of $\tilde{V}_k$ are right singular vectors, with non-zero singular values, of $M_k = \phi P_{k-1}$. They lie in the row space of M_k, which is contained in $\text{range}(P_{k-1}) = \text{span}(Q_{k-1})^\perp$, and right singular vectors are mutually orthonormal. Hence $\tilde{V}_k^\top Q_{k-1} = 0$ and Q_k has orthonormal columns. Singular values below a numerical tolerance are dropped, so Δr_k may be smaller than Δr; the LTM rank reported in the paper is $\sum_k \Delta r_k$ and need not be a multiple of Δr, which is the case for $\tau = 0.4$ and $\tau = 0.8$ in Table 4.

A.2 Proof of Proposition 1 (Interference Bound)

Proof. **Projected consolidation.** Fix h and k_0, and let $z = P_{k_0} h$, so $\|z\|_2 = r_{k_0}(h)$. For every $k > k_0$, Q_{k-1} contains Q_{k_0} as its leading columns and $\tilde{V}_k^\top Q_{k-1} = 0$; hence $\tilde{V}_k^\top Q_{k_0} Q_{k_0}^\top h = 0$ and $\tilde{V}_k^\top h = \tilde{V}_k^\top z$. Therefore

$$\begin{aligned} \big\|(\Phi^{(K)} - \Phi^{(k_0)})h\big\|_2 &= \Big\| \sum_{k=k_0+1}^{K} U_k \Sigma_k \tilde{V}_k^\top z \Big\|_2 \le \sum_{k=k_0+1}^{K} \|\Sigma_k\|_2 \, \|\tilde{V}_k^\top z\|_2 \\ &\le \sigma_{\max} \sqrt{K - k_0} \Big(\sum_{k=k_0+1}^{K} \|\tilde{V}_k^\top z\|_2^2 \Big)^{1/2}, \end{aligned}$$

using $\|U_k\Sigma_k\|_2 = \|\Sigma_k\|_2$ and the Cauchy–Schwarz inequality. The matrix $[\tilde{V}_{k_0+1}, \ldots, \tilde{V}_K]$ has orthonormal columns, so $\sum_k \|\tilde{V}_k^\top z\|_2^2 \le \|z\|_2^2 = r_{k_0}(h)^2$, which gives the first inequality in Eq. (5). At most $K_{\max} = R/\Delta r$ events occur (Algorithm 1), so $K - k_0 \le R/\Delta r$, which gives the second. If $h \in \mathrm{span}(Q_{k_0})$, then $z = 0$ and the response is preserved exactly.

Unconstrained updates, case (a). For $t > t_0$, $D_T = (\Phi_T - \Phi_{t_0})h = -\eta \sum_{t=t_0+1}^{T} w_t G_t h$. By convexity of the norm, $\mathbb{E}\|X\|_2 \ge \|\mathbb{E}X\|_2$, and by the tower property, $\mathbb{E}[w_t G_t h] = \mathbb{E}\big[w_t\, \mathbb{E}[G_t h \mid w_t]\big] = \mathbb{E}[w_t]\, m$. Hence

$$\mathbb{E}\|D_T\|_2 \;\ge\; \eta \Big\| \sum_{t=t_0+1}^{T} \mathbb{E}[w_t]\, m \Big\|_2 = \eta\, c \sum_{t=t_0+1}^{T} \mathbb{E}[w_t],$$

since $\mathbb{E}[w_t] \ge 0$.

Case (b). Let $X_t = w_t G_t h$. For $s < t$, X_s is determined by $\mathcal{F}_{t-1}$, so $\mathbb{E}\langle X_s, X_t\rangle = \mathbb{E}\langle X_s, \mathbb{E}[X_t \mid \mathcal{F}_{t-1}]\rangle = 0$. Expanding the square,

$$\mathbb{E}\Big\| \sum_{t=t_0+1}^{T} X_t \Big\|_2^2 = \sum_{t=t_0+1}^{T} \mathbb{E}\|X_t\|_2^2 = \sum_{t=t_0+1}^{T} \mathbb{E}\big[w_t^2\, \mathbb{E}[\|G_t h\|_2^2 \mid w_t]\big] \;\ge\; c^2 \sum_{t=t_0+1}^{T} \mathbb{E}[w_t^2],$$

and multiplying by η^2 gives the claim. □

The two cases cover the two ways in which later gradients can act on an earlier input. If they push its response in a consistent direction, drift grows linearly in the number of updates; if they are unbiased with respect to it, drift still grows as a random walk. In neither case does it level off, whereas the projected, budgeted LTM does.

Corollary 3 (Network-level preservation)**.** *Evaluate the network with all working memories set to zero. Let x_0 be an input and suppose that, for every adapted matrix W, the input $h_W(x_0)$ computed under LTM state k_0 lies in $\mathrm{span}(Q_{k_0}^W)$. Then the network output on x_0 is the same under every later LTM state.*

Proof. Order the operations of the forward pass topologically and argue by induction. If all inputs to an adapted matrix W are unchanged, then $h_W(x_0)$ is unchanged and lies in $\mathrm{span}(Q_{k_0}^W)$, so its output is unchanged by Proposition 1; frozen operations with unchanged inputs have unchanged outputs. □

When residuals are non-zero, per-layer drift is bounded by Eq. (5) and propagates through the Lipschitz constants of later layers. We do not state a network-level bound for that case and instead measure ϵ directly (Section 4.4; Appendix E.3).

Remark 1 (Bounding $\sigma_{\max}$). Under the update of Eq. (2) with gradients clipped to spectral norm at most G, the WM accumulated since the last reset is $\phi = -\eta \sum_{j\ge 0} \lambda^j g_{t-j}$, so $\|\phi\|_2 \le \eta G/(1-\lambda)$. Projection has spectral norm at most one and truncation keeps the leading singular values, so $\|\Sigma_k\|_2 \le \|\phi P_{k-1}\|_2 \le \eta G/(1-\lambda)$. With Adam on the LoRA factors (Appendix F.2) this bound holds only approximately; Figure 2 therefore evaluates Eq. (5) with the measured $\sigma_{\max}$.

Remark 2 (Orthogonal output directions). If output directions were also kept orthogonal across events ($U_i^\top U_j = 0$ for $i \ne j$), the factor $\sqrt{K - k_0}$ would disappear, because

$$\Big\| \sum_k U_k \Sigma_k \tilde{V}_k^\top z \Big\|_2^2 = \sum_k \|\Sigma_k \tilde{V}_k^\top z\|_2^2 \le \sigma_{\max}^2\, r_{k_0}(h)^2.$$

Algorithm 1 does not enforce this constraint, and all bounds reported in the paper use the version with $\sqrt{K - k_0}$.

A.3 Derivation of the Critical Contamination Rate

Assumptions. (A1) Stream elements, i.e., the batches the gate scores, are i.i.d.: each is entirely corrupted with probability p and entirely regular otherwise. (A2) WM gradients are $g = \alpha_s a + \xi_s$ for

regular and $g = \alpha_c b + \xi_c$ for corrupted elements, where a and b are orthonormal, the fluctuations have zero mean, $\mathbb{E}\|\xi_c\|_2^2 = s_c^2$, and fluctuations are independent across elements. (A3) Slow average: as $\beta \to 1$ on a stationary stream, $\mu_{t-1} \to \mathbb{E}[g] = (1-p)\alpha_s a + p\,\alpha_c b$. (A4) High-dimensional concentration: $\|g\|_2$ concentrates at $(\alpha_c^2 + s_c^2)^{1/2}$ for corrupted elements, and $\langle \xi_c, \mu\rangle / (\|g\|_2 \|\mu\|_2) \to 0$, as for isotropic fluctuations in dimension $D \to \infty$.

Proof of Proposition 2. Under (A2)–(A4), the score of a corrupted element concentrates at

$$S_c = \frac{\langle \alpha_c b + \xi_c, \mu\rangle}{\|g\|_2 \|\mu\|_2} \;\to\; \frac{\alpha_c \langle b, \mu\rangle}{(\alpha_c^2 + s_c^2)^{1/2}\,\|\mu\|_2} = \rho\,\frac{p\,\alpha_c}{\sqrt{(1-p)^2\alpha_s^2 + p^2\alpha_c^2}} = \frac{\rho\, p}{\sqrt{(1-p)^2\kappa^2 + p^2}},$$

with $\rho = \alpha_c/(\alpha_c^2 + s_c^2)^{1/2}$ and $\kappa = \alpha_s/\alpha_c$, which is Eq. (6). (i) Since $p \le \sqrt{(1-p)^2\kappa^2 + p^2}$, $S_c \le \rho$; if $\rho \le \tau$, the strict condition $S_c > \tau$ never holds. (ii) If $\rho > \tau$, both sides of $\rho p > \tau\sqrt{(1-p)^2\kappa^2 + p^2}$ are non-negative, so the condition is equivalent to $p^2(\rho^2 - \tau^2) > \tau^2\kappa^2(1-p)^2$, that is,

$$p\sqrt{\rho^2 - \tau^2} > \tau\kappa(1-p) \quad\Longleftrightarrow\quad p > \frac{\tau\kappa}{\tau\kappa + \sqrt{\rho^2 - \tau^2}} = p^\star(\tau, \kappa, \rho). \tag{11}$$

Writing $p^\star = \left(1 + \sqrt{\rho^2 - \tau^2}/(\tau\kappa)\right)^{-1}$ shows that $p^\star$ increases with τ and κ and decreases with ρ. □

Remark 3 (Regular elements). By the same argument, with $\rho_s = \alpha_s/(\alpha_s^2 + s_s^2)^{1/2}$, a regular element scores $S_s = \rho_s(1-p)\kappa/\sqrt{(1-p)^2\kappa^2 + p^2}$ and triggers consolidation if and only if $\rho_s > \tau$ and $p < p_s = \kappa\sqrt{\rho_s^2 - \tau^2}/\left(\tau + \kappa\sqrt{\rho_s^2 - \tau^2}\right)$. For $\tau = 0.6$ and $\kappa = \rho = \rho_s = 1$, $p^\star \approx 0.43$ and $p_s \approx 0.57$. In the real stream the regular direction changes with every new rule and at every domain boundary, so μ lags behind it and the regular coherence measured over the whole stream is below τ (Table 12). Regular batches then exceed τ only when a direction recurs, which is the sparse triggering of Table 4 and Figure 3.

Remark 4 (Batch scores under per-trace corruption). In our sweeps corruption is injected per trace (Appendix B.4), so every batch mixes regular and corrupted traces and (A1) does not hold literally. Let each trace be corrupted independently with probability p, and in the large-batch limit let the gradient of a batch be $g = (1-p)\alpha_s a + p\,\alpha_c b + \zeta$, where a is the batch's regular mean direction, $b \perp a$ is the shared corrupted direction, and ζ is a zero-mean fluctuation with $\mathbb{E}\|\zeta\|_2^2 = s_B^2$, which shrinks with batch size. Suppose the regular component is not yet represented in the reference, $\mu = p\,\alpha_c b + r$ with $r \perp a, b$ and $\|r\|_2 = \nu$. Under (A4),

$$S = \frac{p^2\alpha_c^2}{\sqrt{(1-p)^2\alpha_s^2 + p^2\alpha_c^2 + s_B^2}\;\sqrt{p^2\alpha_c^2 + \nu^2}} \;\longrightarrow\; \frac{p}{\sqrt{(1-p)^2\kappa^2 + p^2}} \qquad (s_B, \nu \to 0),$$

which is Eq. (6) with $\rho = 1$; for finite batches, ρ is replaced by the coherence of batch gradients, which is why Section 5.5 measures ρ on batch gradients. Batches whose regular direction is already represented in μ score higher still, so above $p^\star$ essentially every batch triggers consolidation, regular and corrupted content alike. This is why the trigger rate approaches one above the boundary; with per-batch injection, only the corrupted batches, at most a fraction p of steps, would trigger. Randomized corruption has $\alpha_c \approx 0$, so its component vanishes from batch means and leaves the score unchanged.

Scope of the approximation. A finite β makes μ fluctuate around its mean, which turns the sharp threshold into a soft transition that begins below $p^\star$; this is consistent with the observed boundaries lying 0.02–0.03 below the prediction (Table 13). Non-stationary streams make μ lag behind domain changes, a non-zero overlap $\langle a, b\rangle$ adds a cross term, and real streams contain several regular directions rather than one. Proposition 2 is therefore a prediction of the location and trend of the boundary, which Table 13 tests; Table 12 reports the measured quantities, including the overlap.

A.4 Implementation on LoRA Factors

Projection and SVD. With $\phi = BA$, the projected update is $\phi(I - QQ^\top) = B\tilde{A}$ with $\tilde{A} = A - (AQ)Q^\top$, at cost $O(r_{\text{WM}} d_{\text{in}} m)$ for $Q \in \mathbb{R}^{d_{\text{in}} \times m}$. Thin QR decompositions $B = Q_B R_B$ and $\tilde{A}^\top = Q_A R_A$ cost $O((d_{\text{in}} + d_{\text{out}}) r_{\text{WM}}^2)$, and the SVD of the $r_{\text{WM}} \times r_{\text{WM}}$ core $R_B R_A^\top = U' \Sigma' V'^\top$ gives $\phi(I - QQ^\top) = (Q_B U')\,\Sigma'\,(Q_A V')^\top$, of which the leading Δr components are kept. This is

the cost stated in Section 3.4; the full $d_{\text{out}} \times d_{\text{in}}$ matrix is never formed. To counter round-off, the new columns $\tilde{V}$ are re-orthogonalized once against Q in fp32 before they are appended; at the end of the 8B math stream, $\max_W \|Q^\top Q - I\|_2 = 2.7 \times 10^{-6}$ over all adapted matrices. The LTM is stored in factored form, so each event adds $(d_{\text{in}} + d_{\text{out}})\Delta r$ parameters per matrix.

Working-memory factors. The decay in Eq. (2) is applied by scaling B by λ after each optimizer step. At every reset, B is set to zero and the rows of A are re-initialized to the leading r_{WM} right singular vectors of the current batch's inputs to that matrix after projection onto $\text{span}(Q)^\perp$. The input directions of the WM, and therefore the directions that enter Q, thus lie in the span of recent activations, as required in Section 3.4; the construction follows the input-subspace initialization of InfLoRA (Liang and Li, 2024).

Additive LTM. The LTM is a plain additive low-rank term. Weight-decomposed parameterizations such as DoRA (Liu et al., 2024), $W = m\,(\Theta + BA)/\|\Theta + BA\|_c$, are not compatible with the projection: column normalization rescales every column of $\Theta + \Phi$ whenever Φ changes, so an update whose input directions are orthogonal to an earlier input would still change the response to that input, and the invariance of Proposition 1 would be lost. Orthogonalizing the rows of A against the accumulated basis, as in Algorithm 1, is sufficient only for the additive form.

A.5 Algorithm

Algorithm 1 SRCL (one adapted matrix shown; the gate is shared across matrices)

Require: stream $\mathcal{S}$; threshold τ; EMA rate β; WM decay λ; step size η; rank step Δr; budget R

1: $\Phi \leftarrow 0$, $\phi \leftarrow 0$, $\mu \leftarrow 0$, $Q \leftarrow [\,]$, $k \leftarrow 0$
2: **for** $t = 1, \ldots, T$ **do**
3: $\quad g_t \leftarrow \nabla_\phi \mathcal{L}(x_t; \Theta + \Phi + \phi)$
4: $\quad S_t \leftarrow \cos(g_t, \mu)$ $\{S_t \leftarrow 0$ while $\mu = 0\}$
5: $\quad \phi \leftarrow \lambda\phi - \eta\, g_t$
6: $\quad$ **if** $S_t > \tau$ **and** $k < R/\Delta r$ **then**
7: $\quad\quad U\Sigma\tilde{V}^\top \leftarrow \text{SVD}_{\Delta r}\left(\phi\,(I - QQ^\top)\right)$
8: $\quad\quad \Phi \leftarrow \Phi + U\Sigma\tilde{V}^\top; \quad Q \leftarrow [Q,\, \tilde{V}]; \quad \phi \leftarrow 0; \quad k \leftarrow k + 1$
9: $\quad$ **end if**
10: $\quad \mu \leftarrow \beta\mu + (1 - \beta)\, g_t$
11: **end for**

The score is computed before the WM step, so S_t compares the current gradient with the reference built from steps $1, \ldots, t-1$ (Definition 1), and consolidation stores the WM including the current step. In the implementation, the gradient g_t is all-reduced across data-parallel workers and concatenated over all adapted matrices before the inner product with μ; the SVD in line 7 is computed per matrix on the LoRA factors (Appendix A.4).

B Streams, Counterfactual Probes, and Corruption

B.1 Counterfactual Design and Stream Statistics

A counterfactual reasoning probe (CRP) is a held-out problem that satisfies two conditions: it can be solved only by applying a rule introduced in the stream, and the answer implied by the familiar, pre-trained semantics differs from the correct one. Success therefore requires applying the new rule rather than reproducing familiar patterns. Rules within a domain are consistent, rules of different domains are disjoint, and each CRP is assigned to the domain that introduces the rule it requires. Validation and test CRPs are disjoint and never appear in training traces.

Table 6 lists the domains in stream order. Each domain spans 50 stream steps of 32 training traces and is seen once, which gives 250 steps for the math stream and 150 for the code stream. Math and Code CRP are means over domains (Section 4.4); because every domain of a stream has the same number of test CRPs, they coincide with means over probes.

Table 6: **Stream statistics.** Domains in stream order. Training traces are 50×32 per domain. .

Stream	Domain	Rules	Steps	Training traces	Validation CRPs	Test CRPs	Probe inputs for ϵ
			Math stream				
Math	M1: arithmetic operations	12	50	1,600	48	120	64
Math	M2: orders and lattices	11	50	1,600	48	120	64
Math	M3: groups	10	50	1,600	48	120	64
Math	M4: finite combinatorics	12	50	1,600	48	120	64
Math	M5: sequences	11	50	1,600	48	120	64
			Code stream				
Code	Python	10	50	1,600	60	150	64
Code	C++	9	50	1,600	60	150	64
Code	Rust	10	50	1,600	60	150	64
Total		**85**	**400**	**12,800**	**420**	**1,050**	**512**

Evaluation schedule. Test CRPs of all domains seen so far are evaluated at the end of every domain, with the full state of each method (Section 4), which yields the accuracy matrix $A_{j,i}$ of Section 4.4; Math and Code CRP are the last row averaged over domains, and BWT is computed from the same matrix (Appendix C.3). Interference ϵ_t (Eq. 8) is evaluated every 5 steps on the probe inputs of Table 6, which are held-out traces rather than CRPs. Figure 2 plots ϵ on first-domain probe inputs over the 50 steps that follow the end of the first domain, so that all methods are compared over the same window; the ϵ values in Tables 3 and 4 are the end points of this window. LTM rank is measured at the end of the stream, and the trigger rate is computed over all 250 steps of the math stream.

B.2 Formal Math (Lean 4) Probes

Math rules are axioms about fresh constants whose names and usage evoke familiar operations, so that the pre-trained prior predicts a different answer. Every axiom set is accompanied by a Lean definition that satisfies all of its axioms, which certifies that the set has a model and therefore cannot prove `False`. A model output is correct if and only if it type-checks against the domain's axioms within 60 s; outputs that contain `sorry` or declare new axioms are rejected, which we verify with `#print axioms`. Listing 1 shows the format of a rule, its certificate, and a probe. The base model solves a small fraction of probes without any update (Table 1), for example by guessing rule names that follow the library's naming convention.

Listing 1: Lean 4 counterfactual rule, consistency certificate, and probe (format example).

```
-- Streamed rule (training traces use these axioms)
axiom add' : Nat -> Nat -> Nat
axiom add'_zero (a : Nat) : add' a 0 = a
axiom add'_succ (a b : Nat) : add' a (b + 1) = add' a b + 2

-- Consistency certificate: a concrete function satisfying both axioms
def addModel (a b : Nat) : Nat := a + 2 * b
theorem addModel_zero (a : Nat) : addModel a 0 = a := by
  show a + 2 * 0 = a; omega
theorem addModel_succ (a b : Nat) : addModel a (b + 1) = addModel a b + 2
     := by
  show a + 2 * (b + 1) = a + 2 * b + 2; omega

-- Counterfactual probe (held out); the additive prior predicts `+ 2`
theorem probe_add' (a b : Nat) : add' a (b + 1 + 1) = add' a b + 4 := by
  have h1 := add'_succ a (b + 1)
  have h2 := add'_succ a b
  omega
```

B.3 Code Reasoning Probes

Code rules alter the documented behaviour of a standard-library construct (Python, C++) or of a small library that mirrors the standard one (Rust). Probes are graded by 6 hidden tests each, with

a 10 s timeout, executed in a runtime that implements the altered behaviour: a patched interpreter module for Python, shimmed library headers for C++, and a shim crate for Rust. A probe counts as solved only if all of its tests pass. Listings 2 and 3 show the format.

Listing 2: Python counterfactual rule and probe (format example).

```python
# Streamed rule (training traces): in this environment range(a, b)
# includes b, e.g. list(range(1, 3)) == [1, 2, 3].

# Counterfactual probe (held out), graded by hidden tests executed in a
# runtime that implements the streamed semantics of range.
def count_multiples(k: int, n: int) -> int:
    """Return how many elements of range(1, n) are divisible by k."""
    ...
# Correct under the streamed rule: n // k   (standard prior: (n - 1) // k
    )
```

Listing 3: Rust counterfactual rule and probe (format example).

```rust
// Streamed rule (training traces): in the `cfstd` crate, windows(xs, k)
// yields all windows of length k and WRAPS AROUND the end of the slice,
// so it returns xs.len() windows instead of xs.len() - k + 1.
use cfstd::windows;

// Counterfactual probe (held out): largest sum over windows of length 3.
// For xs = [5, 1, 2, 4, 6] the non-wrapping reading gives 12;
// the streamed semantics gives 15 (window [4, 6, 5]).
fn max_window_sum(xs: &[i64]) -> i64 {
    windows(xs, 3).map(|w| w.iter().sum::<i64>()).max().unwrap()
}
```

B.4 Corrupted Training Traces

The contamination analysis of Section 5.5 replaces regular training traces with corrupted ones, traces whose content is wrong in a way that cannot be predicted from the rest of the stream. At rate p, each trace of a batch is independently replaced by a corrupted trace, so the batch size and the number of steps are unchanged while the number of regular traces decreases (Remark 4 relates this to the assumptions of Proposition 2). We use two kinds of corruption.

Templated corruption inserts steps drawn from a fixed set of 12 templates into otherwise valid traces. In Lean, the inserted steps are syntactically valid but logically wrong, for example `have h : x = x + 1 := by rfl` placed inside a long proof; in code, they are programs that compile but deadlock by acquiring two mutexes in opposite orders. Because the same templates recur across domains, the gradients of corrupted traces share a direction, and case (ii) of Proposition 2 applies.

Randomized corruption replaces steps with tactics or statements drawn at random from unrelated traces, so corrupted traces are equally wrong but share no common direction.

Templated corruption is swept over $p \in \{0, 0.01, \ldots, 0.50\}$ (51 points) and randomized corruption over $p \in \{0, 0.05, \ldots, 0.50\}$ (11 points), both on the Llama-3.1-8B math stream with $\tau = 0.6$ and three seeds. For Table 13, templated corruption is additionally swept over the 11-point grid at $\tau \in \{0.4, 0.5, 0.7, 0.8\}$. The observed boundary $\hat{p}^\star$ is the smallest rate at which mean Math CRP falls below the midpoint between its clean-stream value and its minimum over the sweep, with linear interpolation between sweep points; it is computed per seed and then averaged.

C Extended Empirical Results and Baselines

C.1 Consolidated versus In-Context Rules

A natural alternative to updating weights is to place the domain rules in the prompt. Table 7 compares the two options. In-context baselines receive all rules of the stream, about 21k tokens for math and 17k for code, which fit within the context window of every model listed; parametric methods receive the probe only (Appendix F.1). GPT-4o and Gemini 1.5 Pro are queried through their APIs at temperature 0, three times per probe, and we report the mean and standard deviation over the three runs; open models use greedy decoding and are deterministic.

SRCL on Llama-3.3-70B exceeds the strongest in-context baseline, Gemini 1.5 Pro, by 41.0 (math) and 43.6 (code) points, and GPT-4o by 44.2 and 46.3 points. Given all rules in context, the unchanged Llama-3.3-70B-Instruct reaches 38.6% and 35.1%, well above its accuracy without the rules but below the sequentially fine-tuned model, so the gap to SRCL is not explained by the choice of base model. Placing all rules in the prompt also makes every query carry the full rule set, which grows with the stream. We do not test why in-context rules are applied less reliably; how models weigh conflicting context against parametric knowledge depends on how the evidence is presented (Xie et al., 2024).

Table 7: **Consolidated versus in-context rules.** API models: mean ± std over three evaluation runs; open models: greedy decoding (deterministic) or mean ± std over three training seeds. Rules in prompt: approximate length of the rules supplied with each probe (math / code). .

Model and mechanism	Rules in prompt	Math CRP (%)	Code CRP (%)
Reference model			
Llama-3.3-70B-Instruct, no update	none	11.8	9.7
In-context rule access			
GPT-4o	all (21k / 17k tokens)	42.1 ± 1.2	38.5 ± 1.4
Gemini 1.5 Pro	all (21k / 17k tokens)	**45.3 ± 0.9**	**41.2 ± 1.1**
Llama-3.3-70B-Instruct, no update	all (21k / 17k tokens)	38.6	35.1
Parametric updates			
Llama-3.3-70B-Instruct + Sequential SFT	none	**51.1 ± 1.8**	**48.5 ± 2.2**
Llama-3.3-70B-Instruct + SRCL	none	**86.3 ± 0.5**	**84.8 ± 0.6**

C.2 Baselines, Additional Architectures, and Scale

Shared settings. All methods adapt the same matrices with adapters of rank $r_{\text{WM}} = 16$, share the learning-rate grid of Table 16, and see each stream once in the same order. Methods that open one adapter per domain (O-LoRA, InfLoRA) use rank r_{WM} per domain.

Reference points. The *base model* is evaluated without updates. *Joint training* trains the same adapter on the shuffled union of all domains of a stream for the same number of steps (250 or 150), so it sees the same data without a sequence.

Task-free baselines. *Sequential SFT* fine-tunes a single LoRA adapter on the stream in order. *Delayed large-batch SFT* accumulates the traces of 10 consecutive steps (320 traces) and then applies 4 optimizer steps with batches of 80 traces. *Continuous PER* keeps a reservoir buffer of 2,048 traces; each step trains on the 32 incoming traces and 32 replayed traces sampled with probability proportional to $\big((1+S(x))/2\big)^{\alpha}$ with $\alpha = 0.6$, which maps $S(x) \in [-1, 1]$ to a non-negative priority. Importance-sampling weights anneal from exponent 0.4 to 1 as in Schaul et al. (2016), and priorities are refreshed whenever a trace is replayed.

Boundary-dependent baselines. All three receive the true domain boundaries. *EWC* estimates a diagonal Fisher information of the adapter parameters on 512 traces at the end of each domain; the penalty weight is selected from $\{10^2, 10^3, 10^4\}$ (selected: 10^3). *O-LoRA* opens a new adapter at each boundary and penalizes overlap with earlier adapters with weight 0.5. *InfLoRA* opens a new adapter

at each boundary whose down-projection is restricted to the complement of the input subspace of earlier domains, retaining 95% of the input energy when estimating that subspace.

Ablations and alternative gates (Tables 3 and 5). *Continuous* updates the LTM at every step with weight $S(x)$ and no projection; this is the $\tau = 0$ row of Table 4. *Every step* consolidates with projection at each step until the budget is exhausted. *Periodic* consolidates with projection at six evenly spaced steps, the number of events of SRCL at $\tau = 0.6$. Without the projection, $\mathrm{SVD}_{\Delta r}(\phi)$ is added to Φ directly and no basis Q is kept. The loss and entropy gates consolidate when the batch loss or predictive entropy exceeds a threshold that is set, per seed and on the validation stream, so that it produces the same six events as $S(x)$ at $\tau = 0.6$ (48 ranks divided by $\Delta r = 8$); the random gate fires at six steps drawn uniformly per seed. All other settings, including R and Δr, are unchanged.

Additional architectures. Table 8 extends Table 2 with standard deviations, backward transfer for all methods, and two further baselines, sequential SFT and O-LoRA; the Llama results are in Table 1. All models use the hyperparameters selected on Llama-3.1-8B. For the mixture-of-experts DeepSeek-V2-Lite, adapters are attached to the attention projections and to the shared experts, while routed experts stay frozen. The ordering of methods is the same on every model. Qwen2.5-14B is the one model on which InfLoRA with oracle boundaries is ahead of SRCL on both streams (by 0.4 and 0.5 points). DeepSeek-V2-Lite shows the smallest gains over PER and the largest remaining forgetting for every method, consistent with the fact that its routed experts, which hold most of its parameters, are frozen and not covered by any adapter.

Table 8: **Additional architectures** (mean ± std over 3 seeds). Continuous PER, InfLoRA, and SRCL CRP and SRCL BWT are also reported in Table 2. O-LoRA and InfLoRA receive oracle boundaries.
.

Method	Math CRP (%)	Math BWT (%)	Code CRP (%)	Code BWT (%)
Mistral-NeMo-12B				
Sequential SFT	33.1 ± 2.0	-38.6 ± 3.1	29.7 ± 2.2	-37.9 ± 3.0
Continuous PER	54.6 ± 1.3	-7.9 ± 1.0	51.4 ± 1.4	-5.9 ± 0.9
O-LoRA	$\mathbf{62.3 \pm 1.1}$	$\mathbf{-3.6 \pm 0.6}$	$\mathbf{58.1 \pm 1.2}$	$\mathbf{-2.9 \pm 0.6}$
InfLoRA	$\mathbf{68.1 \pm 0.9}$	$\mathbf{-1.2 \pm 0.3}$	$\mathbf{63.9 \pm 1.0}$	$\mathbf{-0.9 \pm 0.3}$
SRCL	$\mathbf{69.0 \pm 0.9}$	$\mathbf{-1.5 \pm 0.4}$	$\mathbf{64.2 \pm 1.0}$	$\mathbf{-1.1 \pm 0.3}$
Qwen2.5-14B				
Sequential SFT	36.2 ± 1.9	-36.4 ± 2.9	33.6 ± 2.1	-35.1 ± 2.8
Continuous PER	58.4 ± 1.1	-7.1 ± 0.9	55.2 ± 1.3	-5.4 ± 0.8
O-LoRA	$\mathbf{65.8 \pm 1.0}$	$\mathbf{-3.2 \pm 0.6}$	$\mathbf{62.7 \pm 1.1}$	$\mathbf{-2.6 \pm 0.5}$
InfLoRA	$\mathbf{72.9 \pm 0.8}$	$\mathbf{-0.9 \pm 0.3}$	$\mathbf{69.4 \pm 0.9}$	$\mathbf{-0.6 \pm 0.2}$
SRCL	$\mathbf{72.5 \pm 0.8}$	$\mathbf{-1.0 \pm 0.3}$	$\mathbf{68.9 \pm 0.9}$	$\mathbf{-0.8 \pm 0.3}$
DeepSeek-V2-Lite (MoE)				
Sequential SFT	30.5 ± 2.3	-40.8 ± 3.4	27.3 ± 2.4	-39.6 ± 3.3
Continuous PER	50.2 ± 1.5	-9.6 ± 1.2	47.8 ± 1.6	-7.8 ± 1.1
O-LoRA	$\mathbf{54.9 \pm 1.3}$	$\mathbf{-5.1 \pm 0.8}$	$\mathbf{50.6 \pm 1.4}$	$\mathbf{-4.3 \pm 0.8}$
InfLoRA	$\mathbf{59.8 \pm 1.1}$	$\mathbf{-2.1 \pm 0.5}$	$\mathbf{55.1 \pm 1.2}$	$\mathbf{-1.8 \pm 0.5}$
SRCL	$\mathbf{60.7 \pm 1.2}$	$\mathbf{-2.6 \pm 0.6}$	$\mathbf{54.1 \pm 1.4}$	$\mathbf{-2.2 \pm 0.6}$

Scale. Across the five models, Sequential SFT improves with scale but keeps forgetting heavily (math BWT -42.1% at 8B and -31.4% at 70B; Table 1), so scale alone does not remove interference. SRCL improves with scale as well, and its gap to joint training narrows from 6.2/6.8 points at 8B to 3.4/3.6 points at 70B. On math, the dense models with 12B and 14B parameters fall between the two Llama models for every method; on code, Mistral-NeMo-12B is slightly below Llama-3.1-8B for InfLoRA and SRCL. DeepSeek-V2-Lite, with 15.7B total but 2.4B active parameters, falls below Llama-3.1-8B for every method except sequential SFT. Five models from different families do not determine the functional form of these trends, and we do not fit one.

C.3 Per-Domain Accuracy

Backward transfer summarizes the accuracy matrix $A_{j,i}$. Table 9 lists, for each domain, the accuracy right after it is learned ($A_{i,i}$) and at the end of the stream ($A_{N,i}$) for the 8B model; the averages and BWT of Table 1 follow from these entries. Joint training has no sequence and is omitted.

Under sequential SFT, forgetting is ordered by domain age: M1 retains 11.2% at the end of the stream and M4 retains 28.1%. EWC instead loses plasticity: its accuracy right after learning declines from 63.1% on M1 to 56.2% on M5, as the accumulated Fisher penalty restricts later updates. On the math stream, SRCL has the highest diagonal on every domain (tied with InfLoRA on M4); on the code stream, InfLoRA's diagonal is slightly higher, which accounts for its 0.6-point advantage in Code CRP. SRCL's residual forgetting is largest for the two earliest math domains (-1.6 and -1.5 points), which are exposed to the most subsequent consolidation events. Knowledge that is not consolidated decays with the WM and appears as forgetting in $A_{N,i}$; the last domain is still held in the WM at the end of the stream and, by the definition of BWT, does not enter the forgetting measure.

Table 9: **Per-domain CRP accuracy** (Llama-3.1-8B, %, mean over 3 seeds). For each domain $i < N$: accuracy right after learning it ($A_{i,i}$) / at the end of the stream ($A_{N,i}$); for the last domain, the two coincide. Avg. is the mean of the end-of-stream entries and BWT is computed as in Section 4.4; both equal the values in Table 1. .

Math stream							
Method	**M1**	**M2**	**M3**	**M4**	**M5**	**Avg.**	**BWT**
Sequential SFT	58.7 / 11.2	62.4 / 14.6	59.3 / 19.8	61.7 / 28.1	68.3	28.4	−42.1
Delayed large-batch SFT	55.1 / 32.8	58.3 / 38.2	55.6 / 38.4	58.4 / 45.2	57.9	42.5	−18.2
Continuous PER	57.2 / 46.6	59.8 / 50.0	56.9 / 49.3	59.8 / 54.2	60.4	52.1	−8.4
EWC	63.1 / 29.6	61.0 / 30.8	57.8 / 31.7	56.8 / 36.2	56.2	36.9	−27.6
O-LoRA	63.2 / 58.1	65.8 / 61.4	62.9 / 59.3	65.6 / 63.1	64.1	**61.2**	−3.9
InfLoRA	67.4 / 66.0	69.6 / 68.4	66.9 / 66.1	69.2 / 68.6	68.9	**67.6**	−1.0
SRCL	68.8 / 67.2	70.9 / 69.4	67.6 / 66.6	69.2 / 68.5	70.3	**68.4**	−1.2

Code stream					
Method	**Python**	**C++**	**Rust**	**Avg.**	**BWT**
Sequential SFT	53.1 / 8.5	50.9 / 11.7	52.4	24.2	−41.9
Delayed large-batch SFT	50.8 / 32.9	48.8 / 33.9	49.6	38.8	−16.4
Continuous PER	54.6 / 47.5	53.2 / 47.7	53.9	49.7	−6.3
EWC	56.0 / 24.4	52.7 / 26.5	48.4	33.1	−28.9
O-LoRA	61.5 / 58.0	60.1 / 57.4	59.8	**58.4**	−3.1
InfLoRA	67.5 / 66.7	66.1 / 65.5	64.9	**65.7**	−0.7
SRCL	67.3 / 66.4	65.9 / 65.2	63.7	**65.1**	−0.8

C.4 Sensitivity

Table 10 varies one hyperparameter at a time around the selected configuration on the test CRPs of the 8B math stream. Across all tested values, Math CRP stays within 4.3 points of the selected configuration, and BWT stays between -1.9% and -1.0%. The effects follow the mechanism. Faster WM decay ($\lambda = 0.90$) stores less per consolidation and yields fewer events, while slower decay ($\lambda = 0.99$) mixes older inputs into each consolidation and forgets slightly more. A smaller rank step stores less per event, and a larger one spends the budget faster without adding accuracy. A faster-moving reference ($\beta = 0.8$) makes the score noisier and triggers more events. The EMA rate $\beta = 0.95$ is marginally better on test than the validation choice $\beta = 0.9$ (68.6% vs. 68.4%, within one standard deviation); we report the validation choice throughout.

D Mechanistic Diagnostics and Compute

D.1 Gate Diagnostics

A gate that responds to rare tokens rather than to recurring structure would consolidate batches because of their vocabulary. Table 11 reports the diagnostics of Table 5 for both streams.

Table 10: **Sensitivity** (Llama-3.1-8B, math stream, $\tau = 0.6$; mean $\pm$ std over 3 seeds). One hyperparameter is varied at a time; the selected configuration is in bold. .

Hyperparameter	Value	Math CRP (%)	BWT (%)	LTM rank
Learning rate η				
	1×10^{-5}	**64.1 ± 1.0**	−1.0	40
	$\mathbf{3 \times 10^{-5}}$	**68.4 ± 0.8**	−1.2	**48**
	5×10^{-5}	**66.9 ± 1.1**	−1.7	56
WM decay λ				
	0.90	**64.9 ± 1.0**	−1.4	40
	0.95	**68.4 ± 0.8**	−1.2	**48**
	0.99	**65.8 ± 1.2**	−1.9	56
Rank step Δr				
	4	**65.2 ± 0.9**	−1.0	28
	8	**68.4 ± 0.8**	−1.2	**48**
	16	**66.7 ± 1.0**	−1.6	96
EMA rate β				
	0.8	**66.1 ± 1.1**	−1.5	64
	0.9	**68.4 ± 0.8**	−1.2	**48**
	0.95	**68.6 ± 1.0**	−1.1	40

Definitions. All diagnostics are computed per batch with Llama-3.1-8B and pooled over the three seeds. *Token-frequency correlation* is the Pearson correlation between a signal and the mean log unigram frequency of the batch's tokens, with frequencies estimated on the pre-training-style reference corpus SlimPajama. *Loss correlation* uses the per-batch training loss. *CRP-novelty correlation* uses, as label, the fraction of the batch's traces that apply a CRP-tested rule that has already appeared in the preceding 10 steps but has not yet been consolidated, that is, a rule that is new to the LTM and recurring. *OOD AUC* is the ROC AUC for separating batches from the first 5 steps of each domain from all other batches, with each signal in the orientation that flags a new domain: high loss or entropy, and low $S(x)$, because under Definition 1 the reference μ still points to the previous domain at a boundary. The math values are those of Table 5; the code stream shows the same pattern with slightly weaker separation.

Table 11: **Gate-signal diagnostics on both streams** (Llama-3.1-8B). Math values repeat Table 5. .

	Math stream				Code stream			
Signal	**Corr. freq.**	**Corr. loss**	**Corr. novelty**	**OOD AUC**	**Corr. freq.**	**Corr. loss**	**Corr. novelty**	**OOD AUC**
Raw loss	−0.62	**1.00**	**0.41**	**0.68**	−0.58	**1.00**	**0.44**	**0.66**
Predictive entropy	−0.74	**0.88**	**0.38**	**0.71**	−0.69	**0.85**	**0.40**	**0.69**
Gradient consistency $S(x)$	−0.08	**0.44**	**0.92**	**0.95**	−0.11	**0.39**	**0.87**	**0.91**

Quantities in Proposition 2. We compute WM gradients of regular and corrupted traces, and of purely regular and purely corrupted batches of 32 traces, at the SRCL checkpoint reached after the first 100 steps of the clean math stream, using 512 traces and 64 batches per class. We estimate $\hat{\alpha}_s = \|\bar{g}_s\|_2$ and $\hat{\alpha}_c = \|\bar{g}_c\|_2$ from the trace-level class means and set $\hat{\kappa} = \hat{\alpha}_s/\hat{\alpha}_c$. Coherence is estimated as $\hat{\rho} = \|\bar{g}\|_2/(\overline{\|g\|_2^2})^{1/2}$ over batch gradients, the quantity that enters Remark 4; $\hat{\rho}_s$ is the same quantity for regular batches pooled over the whole stream. The overlap $\cos(\bar{g}_s, \bar{g}_c)$ checks the orthogonality assumption. Table 12 reports the estimates and Table 13 reports the predicted and observed boundaries across thresholds.

With templated corruption, the transition is centred slightly below the prediction at every threshold and is less sharp than the step function of Proposition 2, as expected from the finite averaging window of μ. The overlap of 0.06 supports the orthogonality assumption. With randomized corruption, $\hat{\rho}$ is well below τ and $\hat{\kappa}$ is large, so the corrupted component barely enters the score and the gate does not open at any rate tested; Math CRP still declines gradually, to 61.2% at $p = 0.50$, because each domain is learned from fewer valid traces. The regular coherence $\hat{\rho}_s$ is below τ for both corruption types, consistent with the sparse triggering on the clean stream.

Table 12: **Measured quantities in Proposition 2** (8B, math, $\tau = 0.6$; observed boundary as mean $\pm$ std over 3 seeds). .

Corruption	$\hat{\kappa}$	$\hat{\rho}$ (batch)	$\hat{\rho}_s$ (regular)	$\cos(\bar{g}_s, \bar{g}_c)$	Predicted $p^\star$	Observed $\hat{p}^\star$
Templated	0.72	0.98	0.47	0.06	0.36	0.33 ± 0.02
Randomized	3.9	0.21	0.46	0.02	none ($\hat{\rho} \leq \tau$)	none up to 0.50

Table 13: **Predicted and observed contamination boundaries across thresholds** (8B, math, templated corruption). Predictions use $\hat{\kappa} = 0.72$ and $\hat{\rho} = 0.98$; observed boundaries are mean $\pm$ std over 3 seeds. .

τ	0.4	0.5	0.6	0.7	0.8
Predicted $p^\star$	0.24	0.30	**0.36**	0.42	0.50
Observed $\hat{p}^\star$	0.22 ± 0.02	0.28 ± 0.02	$\mathbf{0.33 \pm 0.02}$	0.40 ± 0.03	0.48 ± 0.03

D.2 Compute and Latency Overhead

All experiments run on 8×NVIDIA H100 (80GB) GPUs (Appendix F.3). Table 14 reports costs on Llama-3.3-70B. SRCL runs one forward and one backward pass per step, because the score requires the WM gradient g_t at every step; the gate itself is an inner product with μ. Consolidation adds a factored SVD per adapted matrix (Appendix A.4), but the gate opens on only 2.4% of the steps of the math stream at $\tau = 0.6$ (six events; Table 4), so on the 70B math stream consolidation accounts for about 11 s of 19 minutes of training. Continuous PER processes 32 replayed traces in addition to the 32 incoming ones at every step, which accounts for most of the difference in per-step cost; SRCL's per-step cost is therefore lower than PER's, while its memory overhead is slightly higher because of μ, Q, and the factored LTM.

Table 14: **Compute and latency overhead** on Llama-3.3-70B (8×H100). Latencies are per 1,024-token sequence; FLOPs and wall-clock time are per stream step of 32 incoming traces; memory is the overhead per GPU on top of the sharded frozen base. FLOPs are computed from the model size and the number of processed sequences, counting the recomputation of activation checkpointing. .

Method	Fwd (ms/seq)	Bwd (ms/seq)	Consolidation (s/event)	FLOPs/step	Wall-clock (s/step)	Memory overhead (GB/GPU)
Continuous PER	45	85	–	2.8×10^{16}	8.6	14.2
SRCL	48	92	1.9	1.4×10^{16}	4.6	15.1

Total compute. A full run on both streams takes about 25 minutes for Llama-3.1-8B and 1.2 hours for Llama-3.3-70B, including evaluation. Including the hyperparameter search, the baselines of Table 1, the additional architectures (Appendix C.2), all ablations and alternative gates, the sensitivity study, and the contamination sweeps (51 templated and 11 randomized rates at $\tau = 0.6$, and 11 rates at each of four further thresholds), all with three seeds, the experiments used approximately 1,500 H100 GPU-hours.

E Qualitative Traces and Failure Modes

E.1 A Successful WM-to-LTM Consolidation

Figure 3 shows a 100-step window of the 8B math stream that starts after two consolidation events (LTM rank 16); step indices are relative to the window start, which is stream step 101. A new axiom, `add'_succ` in the format of Listing 1, first appears at step 31. Its first occurrences produce high loss and high predictive entropy but a low score, because μ is still dominated by the rules seen earlier. As the axiom recurs, μ rotates toward its update direction and $S(x)$ rises to 0.79 at step 42; the gate opens, the WM is consolidated as a rank-8 expansion (LTM rank $16 \to 24$), and ϕ is reset. Evaluated immediately afterwards with the WM cleared, so that only consolidated knowledge is used, the model solves 9 of the 12 held-out probes that require this axiom, compared with 2 of 12 just before consolidation under the same evaluation. A second rise near step 78, from a rule that appears in only

a few batches, peaks at 0.54; it stays below τ, and the corresponding WM content decays instead of being consolidated.

E.2 Budget Exhaustion under Coherent Contamination

At a templated-corruption rate of $p = 0.40$ (8B, $\tau = 0.6$), above the observed boundary $\hat{p}^\star = 0.33$ (Table 12), the corrupted component dominates the reference μ. Because every batch contains corrupted traces, their shared direction lifts the score of almost every batch above τ (Remark 4), and the gate fires on 95% of steps, regular and corrupted content alike. The 256-rank budget is exhausted within 50 steps, after 34 steps on average, that is, within the first domain. Afterwards no new domain can be consolidated, and Math CRP falls to 24.8%, compared with 6.5% for the base model without updates (Table 1); kernel-graded proof generation has no chance level, so the base model is the relevant reference. Table 15 shows where the remaining accuracy comes from. The first domain is partly retained, because it was consolidated together with corrupted content; the last domain is still held in the WM at the end of the stream; and the three middle domains, learned only transiently, fall to near the base model.

Table 15: **End-of-stream accuracy per domain** under templated corruption at $p = 0.40$ and on the clean stream (Llama-3.1-8B, SRCL, $\tau = 0.6$, Math CRP %, mean over 3 seeds). .

Stream	M1	M2	M3	M4	M5	Avg.
Clean	67.2	69.4	66.6	68.5	70.3	**68.4**
Templated corruption, $p = 0.40$	52.3	8.9	7.9	7.4	47.5	**24.8**

The failure is therefore a loss of capacity rather than a corruption of what was protected before the contamination took over: corrupted directions occupy the budget, and the remaining domains can only be held transiently.

E.3 Protected Residuals and the Tightness of Proposition 1

Proposition 1 ties interference to the residual $r(h)$. At the end of the 8B math stream, the median relative residual $\|(I - QQ^\top)h\|_2/\|h\|_2$ of first-domain probe inputs is 0.31 (interquartile range 0.22–0.43), and the Spearman correlation between per-probe residual and per-probe drift is 0.64. Evaluated with the measured $\sigma_{\max}$ and residuals, the bound of Eq. (5) exceeds the measured drift by a median factor of 3.4, the value quoted in Section 5.4. Some slack is expected, because the Cauchy–Schwarz step assumes that the contributions of all events to the drift align.

E.4 Parameter Trajectories

We project the adapter parameters of the adapted matrices, recorded at each of the 50 steps of the window used in Figure 2, onto their first two principal components, fitted jointly on both methods. Continuous PER moves along a single drifting path, whereas the SRCL trajectory ($\Phi + \phi$) stays close to fixed points between consolidation events and moves at each event. The projection is qualitative. Orthogonality of consolidated input directions holds by construction and is verified numerically in Appendix A.4; a two-dimensional projection cannot establish it.

F Reproducibility, Prompts, and Hyperparameters

F.1 Prompt Templates

All models are evaluated zero-shot with greedy decoding and the same answer format. SRCL and all fine-tuned baselines use the *parametric* template, which contains the probe but not the rules; the in-context baselines of Appendix C.1 use the *in-context* template, which adds all rules of the stream. The templates below are for Lean; the code templates differ only in the instruction ("Output ONLY the function body") and in the field names.

```
[SYSTEM]  (parametric)
You are a formal verification
assistant. You will be given a
theorem. Output ONLY a valid
Lean 4 tactic sequence that
proves it. Do not output natural
language explanations.

[THEOREM]
{crp_probe}

[PROOF]
```

```
[SYSTEM]  (in-context)
You are a formal verification
assistant. You will be given a
set of axioms and a theorem.
Output ONLY a valid Lean 4
tactic sequence that proves the
theorem. Do not output natural
language explanations.

[AXIOMS]
{streamed_rules}

[THEOREM]
{crp_probe}

[PROOF]
```

F.2 Hyperparameters and Selection

Table 16 lists all hyperparameters. Hyperparameters are searched one at a time around the SRCL configuration and selected by Math CRP on validation probes of the 8B math stream, averaged over the three seeds; the selected configuration is evaluated once on test probes and reused without change for the code stream, for Llama-3.3-70B, and for the models of Appendix C.2. The threshold is selected from $\{0.2, 0.4, 0.6, 0.8\}$; Table 4 additionally reports $\tau = 0$ and $\tau = 1$ as reference points, and the contamination analysis uses $\tau \in \{0.4, \ldots, 0.8\}$. Each baseline receives the same learning-rate grid, and its rate is selected in the same way.

Table 16: **Hyperparameters.** Values marked as searched were selected on validation CRPs (8B, math). .

Hyperparameter	Values searched	Selected
Validation-selected settings		
Learning rate η	$\{1, 3, 5\} \times 10^{-5}$	**SRCL:** 3×10^{-5}; PER, sequential SFT, EWC: 3×10^{-5}; delayed SFT, O-LoRA, InfLoRA, joint: 5×10^{-5}
WM decay λ	$\{0.90, 0.95, 0.99\}$	**0.95**
Rank step Δr	$\{4, 8, 16\}$	**8**
Threshold τ	$\{0.2, 0.4, 0.6, 0.8\}$	**0.6**
EMA rate β	$\{0.8, 0.9, 0.95\}$	**0.9**
Fixed settings		
LTM budget R (per matrix)	–	256 ($K_{\max} = 32$ events)
WM and baseline LoRA rank r_{WM}	–	16
Adapted matrices $\mathcal{W}$	–	query, value, and output projections; MLP down-projection
Optimizer	–	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$), no weight decay
Batch size / maximum length	–	32 traces / 1,024 tokens
Passes over each stream	–	1
Gradient clipping (global norm G)	–	1.0
Numerical tolerance for singular values	–	$10^{-6}\,\sigma_1$
Decoding	–	greedy, pass@1
Maximum new tokens (Lean / code)	–	512 / 768
Seeds	–	42, 1337, 2026

F.3 Compute Infrastructure

All experiments run on a single node with 8×NVIDIA H100 (80GB) GPUs. The base model is frozen in bf16. For Llama-3.3-70B, the base weights (≈140 GB) are sharded with Fully Sharded Data Parallel in `FULL_SHARD` mode (≈17.5 GB per GPU), and activation checkpointing is applied to every decoder layer, which reduces peak activation memory by 64% relative to no checkpointing at our sequence length. Llama-3.1-8B and the models of Appendix C.2 use the same setup. WM gradients are all-reduced before computing $S(x)$, and consolidation SVDs are computed per matrix

in fp32 on the LoRA factors. Evaluation generates with tensor parallelism over the same eight GPUs. Lean probes are checked with a pinned Lean 4 toolchain and library version, and code probes run in isolated containers.