PARETOMEM: A GRADIENT-DESCENT MEMORY FOR TEST-TIME STREAMING CONTEXT ENCODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory mechanisms endow language models with learning from its own context stream, but test-time writing unseen context chuncks by gradient descent tends to erase what has been written before. We argue that this problem can be structural rather than only a matter of capacity: each write step optimizes only the loss of the chunk currently in view, while the evidence needed to defend earlier chunks has already passed, hence the two objectives are never weighed against each other. We therefore recast streaming writes as online multi-objective optimization over a single shared state, in which every chunk already absorbed contributes an objective that the current update must not sacrifice. ParetoMem realizes this as a write rule instead of an architecture: it retains a small set of anchor gradients from earlier chunks together with a running summary of the rest, and projects each incoming update so that it improves the present chunk without measurably harming the retained ones, allowing each anchor a tolerance that grows with how stale it is. We prove that the rule is approximately Pareto-safe, forgets a bounded amount per step , and cannot collapse the memory. It needs one fixed-size buffer, independent of stream length, leaves the read path and parameters untouched, and can be attached to frozen pre-trained backbones at test time — where it improves every metric on ten benchmarks at two scales, most on the benchmark most sensitive to losing early context, and by margins that grow as the stream deepens.

1 INTRODUCTION

Large language models (LLMs) deployed in real-world settings increasingly encounter contexts that arrive as a continuing *stream* rather than as a single bounded prompt: multi-turn dialogues, document-stream reading, and long-horizon agent interactions give information chunk by chunk and may accumulate to lengths far beyond any fixed context window (Packer et al., 2023; Wang et al., 2024). In such settings, modeling increasingly requires a memory mechanism that can continuously *encode* newly arriving information while *retaining* information written earlier in the stream.

In threads of memory mechanisms, gradient-based context compression (Sun et al., 2025; Behrouz et al., 2024; 2025a; Schlag et al., 2021; Peng et al., 2025; Kuratov et al., 2026) holds a regime well suited to streaming: it maintains a bounded state and treats writing as explicit optimization, where the incoming chunk C_t defines a write loss ℓ_t on the memory state and the state M_t is updated along $g_t \approx \nabla \ell_t(M_t)$ at test time. Its write is therefore *corrective*, since the loss directly measures how well the state encodes the chunk; and could be *architecture-agnostic* if the rule is defined on a state instead of an attention pattern. While this very write is what breaks on a stream: because the objective sees *only* the current chunk while all chunks share one state, a step that reduces ℓ_t can *increase* the loss of an earlier chunk C_s whenever $\langle \nabla \ell_s(M_t), g_t \rangle > 0$, and the damage is irreversible, as C_s has already been discarded. We call this failure mode **catastrophic overwriting**, and stress that it is *structural to this class*: any rule whose objective sees only the current chunk admits such steps.

It is tempting to treat this as continual learning (CL) in disguise, and the streaming setting indeed removes the two resources CL usually assumes: chunks are consumed once, ruling out rehearsal and distillation, and the bounded memory *is* the deployed artifact, hence parameter isolation or capacity expansion (Rusu et al., 2016) defeats its purpose. The substantive gap, however, is epistemic. Because a past chunk cannot be re-encoded, the only evidence about its objective ℓ_s is the *stale* gradient $\hat{g}_s = \nabla \ell_s(M_s)$ observed when C_s was written — at a memory state that has since moved.

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.

Constraint-based CL (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019) requires past gradients at the *current* parameters, and regularization-based CL (Kirkpatrick et al., 2017) anchors to a fixed importance estimate rather than to per-chunk descent directions; both presume evidence the stream cannot supply. *It is this staleness that makes the regime distinct*, and it drives our methodology.

Specifically, this points to a remedy stated entirely in the language of memory: before committing a write, check it against what earlier chunks needed, and steer it around the conflict. What makes the check affordable is that every chunk, at the moment it was written, already left behind a single vector recording which changes to the memory would hurt its own reconstruction. Keeping a few such vectors is not keeping content — the chunk cannot be read back from one — so the write-once nature of the stream is respected, the memory does not grow, and a conflicting write is redirected rather than blocked. In a view of optimization, it is literally an ***online multi-objective problem on a shared state***: each chunk contributes a single objective, and the ideal write is a direction that helps the current chunk while hurting no earlier one, which we call ***Pareto-safe***. The single obstacle is that these records go out of date: each describes the memory as it *was*, while the write acts on the memory as it *is*. Existing gradient-surgery solvers (Désidéri, 2012; Yu et al., 2020; Liu et al., 2021; Zhang et al., 2025) assume all objectives are measured at the same state; constraint-based CL re-evaluates past objectives on stored data; asynchronous optimization (Recht et al., 2011) delays a *single* objective that can be recomputed. What is missing is a **staleness-aware relaxation**: enforce each record only up to a slack matching how far the memory has moved since it was taken — a distance the writer can track online by accumulating the lengths of its own updates, with no past states kept.

The aforementioned discussion results in **ParetoMem**, a *lightweight, architecture-agnostic streaming WRITE rule for gradient-descent memory* (Fig.1). At each step it holds a sliding buffer of recent records, tests the intended write against each, and removes only the component that would hurt it, while granting a tolerance proportional to that record's staleness, so older and more drifted evidence is trusted proportionally less. The tolerance requires no knowledge of the loss landscape and leaves a single coefficient to tune; it cannot cancel the staleness error — nothing can — but it stops the writer from enforcing evidence beyond its precision. We certify that the resulting write is (i) *approximately safe for past chunks*, with the residual harm bounded by quantities the writer *measures* rather than assumes away; (ii) *forgetting-bounded*: with non-increasing step sizes, the total loss increase over the W most recent chunks shrinks quadratically in the write step size η, as $O(K^2W^2\eta^2)$ over a protection window of W chunks (KW write steps); and (iii) *non-collapsing* with no extra assumptions: the fraction of the intended write that survives is a conflict coefficient logged at every step, so a write can be neutralized only when the memory is effectively frozen — a condition visible online rather than a hidden failure. The rule is drop-in — a sequential projection, a sliding buffer, and an EMA summary of older records, costing $O(W)$ inner products per write, with no change to the architecture or the read path — and, because it only redirects writes, it transfers across memory parameterizations: we instantiate it on compact memory tokens and on deep layer-wise memory inside a frozen backbone.

2 Background and Problem Formalization

Gradient-descent memory writing. Test-time gradient memory mechanisms differ mainly from where the update rule lives: *architecture-coupled* designs realize it inside a learned layer (Sun et al., 2025; Behrouz et al., 2024; 2025a; Schlag et al., 2021; Peng et al., 2025), whereas *architecture-agnostic* designs attach an explicit differentiable state to a *frozen* backbone and write by optimizing a self-supervised loss at test time (Kuratov et al., 2026). We adopt the latter routine, due to its write rule exposed as a separate algorithmic object that can be analyzed and modified. Let f_θ be a frozen backbone and $M \in \mathcal{K} \subset \mathbb{R}^{m\times d}$ a memory of m tokens with ambient dimension $P = md$ and $\mathcal{K}$ bounded. For a chunk $C = (x_1, \dots, x_{|C|})$, the write loss and the write step are

$$\mathcal{L}_{\text{write}}(M; C) = -\sum_{i=1}^{|C|} \log f_\theta(x_i \mid [M; x_{<i}]), \quad M \leftarrow \Pi_{\mathcal{K}}(M - \alpha \nabla_M \mathcal{L}_{\text{write}}(M; C)), \tag{1}$$

iterated for K inner steps, with $\Pi_{\mathcal{K}}$ the projection onto $\mathcal{K}$. Read is unchanged by $f_\theta(M, q)$ to answer a query q. Nothing below depends on the form of ℓ beyond the smoothness (Assumption.D1).

The streaming protocol. Before diving into the streaming setup, we go through the notations: t indexes *write steps*, not chunks: a stream of T chunks contributes K steps each, for $N = TK$

steps on the same state. We write M_t for the state before step t, ℓ_t for the write loss active at step t (constant over a chunk's K steps), $\ell_t(M) := \mathcal{L}_{\text{write}}(M; C_t)$ (Eq. 1), and $g_t := \nabla \ell_t(M_t)$. Guarantees are stated per write step; the implementation retains one *anchor* gradient per past chunk.

Given this, a context arrives as an ordered stream $C_1, \ldots, C_T$ written into a single memory, after which arbitrary queries over the whole stream may be issued. Three constraints define the regime: **(A1) consume-once** — each C_j is available only while being written, and cannot be re-read; **(A2) single bounded shared state** — all chunks are written into the same M of fixed size D, independent of T; **(A3) stale evidence only** — the sole trace of a past chunk C_s is the gradient $\hat{g}_s := \nabla \ell_s(M_s)$ observed at the *then-current* state $M_s \neq M_t$. These are not implementation artifacts: (A1) is what makes a setting streaming rather than batched, (A2) what makes memory a compression device rather than a cache, and (A3) their immediate consequence. Difficulty is governed not by absolute token count but by the *stream depth* $N = TK$ and the compression ratio $|C|/|M|$, since drift accumulates at every one of the N steps.

The oracle streaming write of memory. Updating $M_{t+1} = M_t - \eta_t u_t$ changes a past chunk's loss by $\ell_s(M_{t+1}) - \ell_s(M_t) = -\eta_t \langle u_t, \nabla \ell_s(M_t) \rangle + O(\eta_t^2)$, so to first order chunk s is preserved iff $\langle u_t, \nabla \ell_s(M_t) \rangle \geq 0$. This yields the problem one would ideally solve at each step, in which retention is a constraint and encoding the incoming chunk is the objective.

Definition 1 (**Oracle write direction**). At step t, the *oracle write direction* is any solution of

$$\max_{u \in \mathbb{R}^P} \langle u, g_t \rangle \quad \text{s.t.} \quad \langle u, \nabla \ell_s(M_t) \rangle \geq 0 \;\; \forall s < t, \qquad \|u\| \leq \|g_t\|. \tag{2}$$

Each retention constraint removes a half-space, so attainable write progress $\langle u, g_t \rangle$ can only decrease as constraints accumulate: some loss of plasticity is *structural* to any method that protects past chunks. What matters is that it be bounded and measurable, which is what Section 4 establishes.

Why oracle write is infeasible under (A1)–(A3). The constraints require $\nabla \ell_s(M_t)$, whose evaluation needs a fresh pass over C_s and thus violates (A1); by (A3) the only surrogate is the stale $\hat{g}_s$, recorded $t - s$ steps earlier. Their disagreement is controlled by the *drift* $D_{t,s} := \|M_t - M_s\| \leq \sum_{r=s}^{t-1} \eta_r \|u_r\|$, then the smoothness localizes the true gradients

$$\left\| \nabla \ell_s(M_t) - \hat{g}_s \right\| \;\leq\; L \, D_{t,s} \tag{3}$$

to cuts both ways. Enforcing $\langle u_t, \hat{g}_s \rangle \geq 0$ verbatim is *unsound*, as the constraint is trustworthy only up to an error growing with drift; enforcing it with *no* relaxation is simultaneously *over-conservative*, forbidding directions the evidence cannot rule out and needlessly suppressing the incoming chunk. Ignoring drift entirely collapses 2 to the unconstrained step 1, whose failure mode is the catastrophic overwriting. Our research aims to turn Eq.3 into a computable tolerance that resolves both at once.

3 ParetoMem: Staleness-Aware Streaming Gradient Memory

ParetoMem changes only the WRITE rule (Eq. 1) of (Kuratov et al., 2026). Instead of applying the current chunk's write gradient directly, it first *filters* that gradient against a small buffer of gradients left behind by earlier chunks, removing the components that would overwrite them. The filter is deliberately *loose* for old entries, because a stored gradient becomes less trustworthy the further the memory has moved since it was recorded. Nothing else changes: the backbone stays frozen, no raw chunk is retained, and the READ interface is untouched.

3.1 Streaming Writing Is Online Multi-Objective Optimization on One State

Each arriving chunk C_t contributes one write loss $\ell_t(M)$ on the shared memory state, with the backbone parameters θ frozen. Reading ℓ_s as *"how badly the memory currently reconstructs chunk s"*, a stream of t chunks therefore leaves us with the vector objective

$$\min_{M \in \mathcal{K}} \big(\ell_1(M), \, \ell_2(M), \, \ldots, \, \ell_t(M) \big), \tag{4}$$

a genuine *multi-objective* problem: in general no single memory state minimizes all components at once, only trade-offs exist, and every write is a choice among them. Two features distinguish Eq. 4 from the multi-task setting that gradient-based MOO solvers were designed for (Désidéri, 2012; Yu et al., 2020; Liu et al., 2021). First, the objectives *arrive online* and each is observable only while its chunk is being written (A1). Second — and consequently — we cannot aim at a Pareto-optimal

point, since we cannot even *evaluate* ℓ_s for $s < t$ any more; the most we can ask of a single write is that it not *sacrifice* any earlier objective.

Formally, let $g_t = \nabla_M \ell_t(M_t)$ and let the update be $M_{t+1} = M_t - \eta_t u_t$. For an earlier chunk $s < t$, $\ell_s(M_{t+1}) - \ell_s(M_t) = -\eta_t \langle u_t, \nabla \ell_s(M_t) \rangle + O(\eta_t^2)$, so to first order a direction u_t leaves chunk s no worse off iff $\langle u_t, \nabla \ell_s(M_t) \rangle \geq 0$; we call such a direction *safe* for s, and safe for *all* protected chunks concurrently: no old chunk traded off against another (**Pareto-safe**). This is exactly the oracle write direction of Definition 1: descend on the incoming chunk subject to a retention constraint per past chunk. As Section 2 notes, each constraint removes a half-space, so the attainable write progress $\langle u, g_t \rangle$ can only shrink as constraints accumulate: *some* loss of plasticity is structural to any method that protects the past, and what matters is that it be bounded and measurable (see Takeaway 3).

What goes wrong without the constraints. The streaming extension of GradMem (Kuratov et al., 2026) takes $u_t = g_t$, dropping every constraint. ***This is locally optimal for writing C_t, yet it may satisfy $\langle g_t, \nabla \ell_s(M_t) \rangle < 0$ for some earlier chunk, so the very step that encodes the incoming chunk increases the write loss of an old one.*** Concretely, if chunk 3 stated that *Alice is a cardiologist* and chunk 40 discusses an unrelated patient, the descent direction for chunk 40 can have a negative inner product with chunk 3's gradient; after enough such steps the fact is no longer recoverable at READ time. This local gradient conflict is the mechanism behind catastrophic overwriting, and its online update manner prevents the adoption of MOO solvers due to the conflict of (A1) and (A3).

3.2 Staleness-Aware Buffered Pareto Projection

Every past chunk s leaves behind exactly one vector: the gradient it already computed when it was written, $\widehat{g}_s := \nabla_M \ell_s(M_s)$ (the anchor taken at the first of its K steps). It records which changes to the memory would have hurt chunk s — *back then*. Since the memory has moved on, the record is out of date, and the first question is by how much. If the write loss varies smoothly in the memory tokens (L-smooth on the admissible set $\mathcal{K}$: gradients cannot jump while the memory moves only a little; Assumption D1, our only hypothesis on the objective), then

$$\|\nabla \ell_s(M_t) - \widehat{g}_s\| \;\leq\; L\|M_t - M_s\| \;\leq\; L\, D_{t,s}, \qquad D_{t,s} := \textstyle\sum_{j=s}^{t-1} \eta_j \|u_j\|, \tag{5}$$

that is, **a stored gradient is wrong by at most the distance the memory has travelled since it was stored**. We call $D_{t,s}$ the *staleness* of entry s; the writer accumulates it online at $O(1)$ cost per entry and never revisits a past memory state.

This licenses enforcing the retention constraints of Definition 1 on stale gradients, provided each is granted slack in proportion to its staleness. With $\mathcal{B}_t$ the currently protected chunks, ParetoMem writes in the admissible direction closest to the raw GradMem gradient,

$$u_t^\star = \Pi_{\mathcal{C}_t}(g_t) = \arg\min_{u \in \mathcal{C}_t} \tfrac{1}{2}\|u - g_t\|_2^2, \qquad \mathcal{C}_t := \big\{u :\; \langle u, \widehat{g}_s \rangle \geq -\xi_{t,s} \;\; \forall s \in \mathcal{B}_t\big\}. \tag{6}$$

Read it as: each protected chunk vetoes the directions that clearly damage it; what survives is a cone $\mathcal{C}_t$ of admissible directions; the write is the point of that cone closest to g_t, so the incoming chunk is compromised only where it genuinely conflicts with the past. We use this projection rather than the linear program of Definition 1 because *(a)* its solution is unique; *(b)* $\xi_{t,s} \geq 0$ makes the origin feasible, which alone forces $\|u_t^\star\| \leq \|g_t\|$ — the oracle's norm cap holds for free — and prevents $u_t^\star$ from ever hurting the current chunk (Takeaway 3); *(c)* it admits the cheap solver of Section 3.3.

How much slack. By Eq. 5 a stale record is trustworthy only to precision $LD_{t,s}$; enforcing it exactly would outlaw directions the evidence cannot rule out. ParetoMem therefore sets, by construction,

$$\xi_{t,s} := \lambda\, D_{t,s}, \qquad \lambda > 0, \tag{7}$$

so fresh entries are enforced almost exactly and drifted ones loosely. Two properties matter later. *(i)* λ is the only tolerance knob, and Eq. 7 involves neither L nor u_t: no smoothness constant is ever estimated, and the constraint is not defined through its own solution. *(ii)* Slack does not *repair* staleness — a larger λ can only weaken the guarantee (Prop. 3); it buys feasibility and non-conservatism. The choice $\lambda = \Theta(LG)$ makes the relaxation *order-free*[1]: the violation budget $\xi_{t,s} + L\|u_t\|D_{t,s}$ stays of the same order as the unavoidable staleness error alone.

[1] In practice L is unknown; Appendix F.3 fixes λ through an equivalent, estimate-free parameterisation—a protection budget H_0 (in write steps) and a dimensionless slack τ—so that every anchor is enforced up to age $H_{\max} = H_0/\tau$ (Cor. 7). All bounds below hold for any $\lambda > 0$ with constant $(\lambda + LG)$; $\lambda = \Theta(LG)$ is the value at which the two terms are balanced.

What the writer measures. Eq. 6 filters every write through constraints built from anchors recorded in the past, so its behaviour is governed by three numbers the writer already logs rather than by assumptions on the stream: the *drift* $D_{t,s}$, how far memory has moved since chunk s was written and therefore how stale its anchor is; the *leftover violation* $\varepsilon_t := \max_{s\in\mathcal{B}_t}(-\xi_{t,s} - \langle u_t, \widehat{g}_s\rangle)_+ \geq 0$, how far Eq. 6 is still from satisfied when the cheap solver stops ($\varepsilon_t = 0$ for the exact projection or under Lemma E.5); and the *conflict coefficient* $\widehat{\alpha}_t := \|u_t - g_t\|/\|g_t\|$, how much the past reshaped the current write, 0 meaning it was accepted unchanged and 1 that the past cancelled it entirely. Staleness, solver inexactness and interference are thus all observable at run time. The one quantity we must assume is G, bounding write-gradient norms on $\mathcal{K}$ (Assumption D.2), in practice the clipping threshold such writers already apply; with $\|u_t\| \leq \|g_t\|$ this gives $\|u_t\| \leq \|g_t\| \leq G$.

Takeaway 1 — Stale constraints are still valid constraints (Prop.3 , Thm.4)

Each buffered constraint is computed at *write* time, yet it still controls the *current* conflict with that chunk: by Eq. 5 the error is a single term proportional to the drift $D_{t,s}$, plus whatever the solver left unsatisfied. Accumulated over a protection window of W chunks with each for a K-step update, a protected chunk's write loss can therefore rise by at most

$$O\big(L\,G^2\,K^2\,W^2\eta^2\big) \qquad \text{(second order in the write step size)},$$

at $\lambda = \Theta(LG)$ and provided the write direction is exactly feasible. An inexact solver adds a *first*-order term $\bar{\varepsilon}KW\eta$ that dominates as η shrinks — which is why ParetoMem falls back on a certified feasible direction (Sec. 3.3) rather than an approximately feasible one.

Second order in η is only meaningful if the comparison is fair. One could object that the bound is small merely because every write is small, and that writing the gradient directly would inherit the same rate for free. It would not:

Takeaway 2 — The gain is an order in the step size, not a constant (Prop.5)

Unconstrained memory writing — the default in test-time memory learning — is *not* accidentally safe. On an explicit two-chunk instance satisfying the same assumptions with the same constants, a writer that applies g_t directly provably forgets $\Omega(KW\eta G^2)$ — first order in η — so the ratio to Takeaway 1 is

$$\text{filtered} \,/\, \text{unconstrained} \;=\; \Theta(L\,K\,W\eta),$$

a gap in the *order* of η rather than a constant factor, hence a provable advantage in the practically relevant regime $\eta = o(1/(LKW))$: turning the write step down helps ParetoMem strictly faster than it helps an unconstrained writer, so the two should diverge, not converge.

3.3 Algorithmic Realization

ParetoMem refers to Algorithm 1 (the brief pipeline shown in Fig.1) elaborated as follows.

One write step. Takeaways 1–2 constrain the write *direction* only, so Eq. 6 is a drop-in wherever memory is written by gradient descent at test time; we instantiate it on a standard meta-learned substrate (Kuratov et al., 2026) purely to keep the comparison controlled. At step t (chunk C_t, one of its K steps): (i) obtain $g_t = \nabla_M \ell_t(M_t)$ from one frozen-backbone backward pass, the single gradient any gradient-written memory already computes; (ii) read each buffered entry's cached drift r_s to form $\xi_{t,s} = \lambda D_{t,s}$; (iii) filter g_t into u_t by Eq. 6; (iv) update $M_{t+1} = M_t - \eta_t u_t$, and at the chunk's first step push $(\widehat{g}_t,\ t,\ r_t = 0)$; every step then advances all cached drifts, including the newly pushed one and the EMA statistic $\bar{r}$, by $\eta_t\|u_t\|$.

Filtering, cheaply. Eq. 6 is a small convex QP in $\mathbb{R}^D$ with $|\mathcal{B}_t| \leq W+1$ constraints, negligible beside the WRITE backward pass; for speed we instead sweep through the constraints once. Starting from $u \leftarrow g_t$, whenever $\langle u, \widehat{g}_s\rangle < -\xi_{t,s}$ we nudge u just onto that constraint's boundary,

$$u \leftarrow u + \frac{-\xi_{t,s} - \langle u, \widehat{g}_s\rangle}{\|\widehat{g}_s\|_2^2 + \delta}\,\widehat{g}_s, \tag{8}$$

with δ a small stabiliser; one or two cyclic passes suffice. Eq. 8 is a *relaxed* projection onto a half-space containing the origin, which is what preserves $\|u_t\| \leq \|g_t\|$ for the approximate solver too. Whatever violation survives is the ε_t charged linearly in Takeaway 1, so once it exceeds a threshold we restore exact feasibility ($\varepsilon_t = 0$) by a single scalar rescaling $u_t = \kappa_t u$,

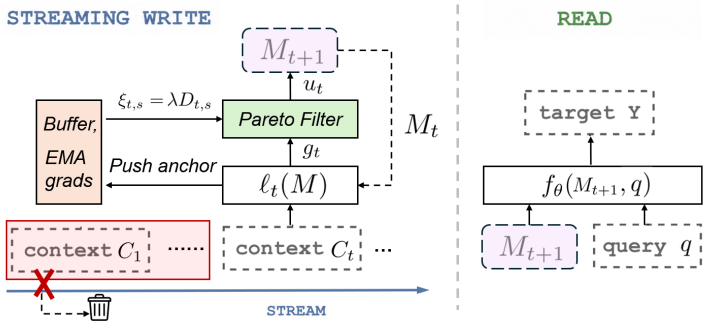

Figure 1: **ParetoMem:** Context chunks are consumed once in a stream. The write gradient g_t is projected by the Pareto filter against buffered/EMA anchors with drift tolerance $\lambda D_{t,s}$, committing M_{t+1}; the chunk is then discarded. Reads use only $f_\theta(M_{t+1}, q)$.

$\kappa_t = \min\{1, \min_{s:\text{ violated}} \xi_{t,s}/|\langle u, \widehat{g}_s\rangle|\}$, costing $O(WP)$ and provably admissible (Lemma E.5). Either solver, though, hands the memory a direction that is no longer the one the current chunk asked for, and Takeaways 1–2 say nothing about that chunk:

Takeaway 3 — Pareto safety costs progress only by standing still (Thm.6)

Filtering the write direction never turns it into an *ascent* direction for the chunk we are trying to store: with the exact projection $\langle g_t, u_t^\star\rangle \geq \|u_t^\star\|^2 \geq 0$ and $\alpha_t \in [0, 1]$ hold unconditionally, because the origin is feasible — a consequence of the construction, *not* an assumption, whereas projection-based analyses in continual learning must *assume* the conflict level stays below 1. What can happen is $\alpha_t \to 1$, and this occurs *if and only if* $u_t^\star \to 0$: the single failure mode of ParetoMem is a frozen memory. Since $\widehat{\alpha}_t$ is logged at every step, that failure mode is diagnosable at run time, and we respond by raising λ or shrinking the window.

What the buffer holds. ParetoMem keeps a fixed-size sliding buffer $\mathcal{B}_t$ storing, for each of the W most recent chunks, the pair $(\widehat{g}_s, r_s)$ under FIFO eviction, with r_s caching the drift needed for $\xi_{t,s}$. This leaves two design questions that Takeaway 1 cannot answer, since there a window of size W is simply given: how large should W be, and what protects a chunk once it is evicted? For the second we add a single further constraint, a normalised exponential moving average of *evicted* anchors, $\bar{g}_t = \beta\bar{g}_{t-1} + (1-\beta)\widehat{g}_{\text{evict}}$ (initialised to $\widehat{g}_{\text{evict}}$ at the first eviction); $\bar{\xi}_t = \lambda\bar{D}_t$, with $\bar{D}_t$ maintained online as $\bar{r} \leftarrow \beta\bar{r} + (1-\beta)r_{\text{evict}}$ at eviction and $\bar{r} \leftarrow \bar{r} + \eta_t\|u_t\|$ at every step. Takeaway 4 settles the first question and explains why the second device is necessary rather than just a longer window:

Takeaway 4 — Per-chunk protection expires: why we also keep an EMA (Thm.8 , Cor.7)

Because drift accumulates monotonically, every FIFO constraint becomes *vacuous* after a finite protection horizon $H^{\max} = \lceil G^2/(\lambda\rho_{\min})\rceil$ write steps (Cor. 7), and vacuity is detectable online from the logged drift ($\lambda r_s \geq G\|\widehat{g}_s\|$), so a FIFO entry older than $H^{\max}$ is pure cost: W is capped at $H^{\max}/K$ chunks, and with the calibration of App. F.3 this cap is $H_0/(\tau K)$, *independent of* W *itself*. Long-range protection therefore cannot come from FIFO entries. The EMA entry supplies it in the *aggregate*: its staleness $\bar{D}_t$ is a geometrically weighted average that stays bounded uniformly in t (Prop. 8), so its constraint never expires—but it protects only a weighted average of evicted chunks and is vacuous when their stale gradients cancel (Remark E.7). This is why ParetoMem uses a hybrid buffer, and we state the aggregate case as a limitation.

What the substrate contributes, and what it does not. Everything outside the write direction is standard and inherited unchanged: frozen slow weights, memory-as-fast-state, and a meta-learned initialization M_0 such that few write steps suffice, obtained by backpropagating through the unrolled WRITE steps. In particular the outer loop is untouched, so Takeaways 1–4 isolate the effect of the inner update direction alone (Algorithm 1) and transfer to any substrate of this form. READ is likewise unmodified: given a query Q, generate with the final memory as prefix, $f_\theta(Y \mid [M_{T+1}; Q])$ with no raw-context replay, no retrieval index, no task-specific module and no extra trainable parameter.

Cost. The buffer stores *gradients, not text*: $O(WP)$ values for D writable memory parameters, so no past chunk is ever retained in readable form. Filtering costs $O(WP)$ per write plus $O(P)$ for the EMA constraint, against the backward pass that producing g_t already requires; the overhead is modest and, unlike replay or retrieval baselines, independent of the stream length and of the number of frozen backbone parameters. It is also bounded by design: W cannot usefully exceed $H^{\max}/K$

Table 1: Associative KV retrieval accuracy (%, indicates exact match) across the number of KV-pairs. For ParetoMem and GradMem variants, their chunk sizes are fixed to 64 (#KV-PAIRS).

MODEL	Context Length T (KV-PAIRS)							
	64	256	512	1024	2048	Old@R	New@R	Old-New Gap
ARMT: M=per-layer associative matrix	22.6	9.5	1.9	0.5	0.7	0.5 ± 0.2	2.5 ± 0.4	-2.0
RMT: M=8 mem vectors	19.3	5.7	0.6	0.3	0.4	0.0 ± 0.0	1.2 ± 0.2	-1.2
GRADMEM:M=8 mem vectors								
—SINGLE-SHOT	**99.1**	**82.4**	63.7	41.8	24.6	**23.1** ± 4.6	26.4 ± 4.2	**-3.3**
—NAIVE SEQUENTIAL	**99.1**	61.5	38.9	21.4	10.8	4.8 ± 2.1	24.6 ± 5.4	-19.8
—REPLAY	**99.1**	**86.7**	**70.5**	**50.3**	**32.7**	18.6 ± 4.4	**39.8** ± 5.0	-21.2
PARETOMEM:M=8 mem vectors								
—VANILLA	**99.1**	73.8	55.6	36.9	21.7	17.9 ± 3.8	28.6 ± 4.5	-10.7
—EMA	**99.1**	79.6	**63.8**	**44.5**	**28.9**	**24.8** ± 4.1	**32.5** ± 4.4	**-7.7**

(Takeaway 4), so the four non-obvious choices in Algorithm 1 — the staleness-proportional tolerance, the exact-feasibility safeguard, the window cap and balance of λ, and the EMA entry — are each fixed by a takeaway rather than tuned.

4 EXPERIMENTS

In this section, we propose different experiment regimes to evaluate ParetoMem and other baselines. We provide the main results and key insightful analysis, then the ablation is found in Appendix.F.

4.1 STREAMING CONTEXT-REMOVAL FOR CONTROLLABLE DIAGNOSIS

Benchmark. In this regime, we consider the context-removal associative KV retrieval task from (Kuratov et al., 2026). In the original benchmark, each example consists of a set of key–value pairs $C = \{(k_i, v_i)\}_{i=1}^{n}$. The model first writes the entire context into a fixed-size memory, after which the raw context is removed. At test time, the model receives only the learned memory together with a query key $q = k_i$, and it must generate the corresponding value v_i. Our experiment reconfigure this setup to suit the streaming memory update. Specifically, we partition the KV context into a temporally ordered stream of chunks $C = C_1 \| C_2 \| \cdots \| C_T$, where C_t contains a small batch of KV pairs. The model maintains a fixed-size memory M_t, and when C_t arrives, it updates the same memory via $M_{t+1} = \text{Write}(M_t, C_t)$. Afterwards, C_t is discarded and cannot be accessed again. Once all chunks have been processed, the model is queried via the final memory and the query, i.e., $[M_T; q]$, and must retrieve values from any historical chunk, including early (chunks in $1 \sim \frac{T}{3}$), middle(chunks in $\frac{T}{3} \sim \frac{2T}{3}$), and recent (chunks in $\frac{2T}{3} \sim T$) portions of the stream. This preserves the context-removal interface of GradMem while introducing the continual-write pressure needed to evaluate overwriting and long-range retention in streaming memory updates along context scaling.

Baselines and Evaluation. Lightweight plug-in memory RMT Bulatov et al. (2022), ARMT Rodkin et al. (2024), and GradMem Kuratov et al. (2026) are considered as memory-based baselines. In particular, to evaluate the streaming memory setup for GradMem, we consider three variants derived from its original implementation: *i.e.*, GradMem (SINGLE-SHOT): all chunks are concatenated and written into memory once before querying, as a non-streaming upper-reference baseline; GradMem (NAIVE SEQUENTIAL): chunks are processed in temporal order and each chunk directly updates the same memory without any protection for previous chunks, exposing the overwriting issue; and GradMem (REPLAY): the writer updates memory using the current chunk together with a small buffer of recent raw chunks, providing a stronger but storage-heavy reference that violates the strict no-replay streaming assumption. As counterparts, we evaluate two variants of ParetoMem: ParetoMem (VANILLA): the current GradMem update direction is projected onto the feasible half-spaces induced by buffered stale gradients, preventing destructive updates to previously written chunks without storing raw contexts; and ParetoMem (EMA): an enhanced variant that additionally maintains an exponentially weighted average of historical gradient summaries, improving retention beyond the finite explicit buffer window. The baselines use different types of memory modules (see Table 1), and all GradMem/ParetoMem variants use the same memory-token budget and perform five meta-gradient write iterations for each chunk. We adopt *Accuracy / Exact Match* as the task metric, and position-conditioned recall over early, middle, and recent chunks to quantify streaming retention and overwriting.

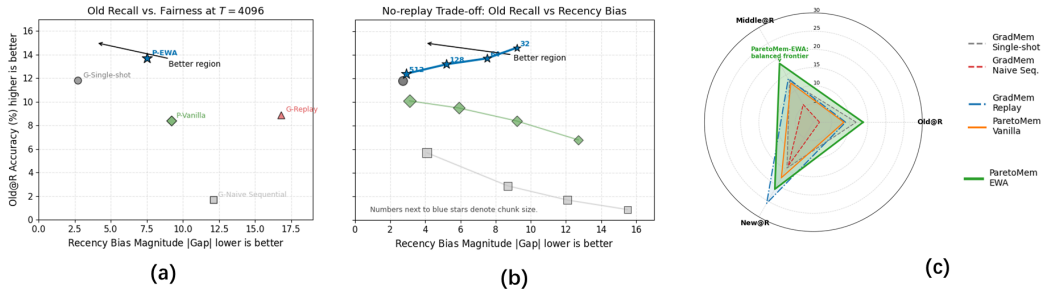

Figure 2: The statistics of (a) Old Recall vs. Fairness at T=4096; (b) No-Replay Trade-off across Chunk Sizes; (c) Pareto Frontier. Details refer to Appendix.F's Figure.3 , 4 , 5 .

Main Results. Table 1 shows that PARETOMEM achieves the strongest retention–plasticity trade-off under a fixed memory budget (when the chunk size=64, it means no stream so that all baselines collapse into a single chunk GradMem). Naive Sequential GradMem collapses from high short-stream accuracy to only 10.8% at 2048 KV pairs with a severe Old–New gap of -19.8, whereas PARETOMEM-EMA reaches 28.9% ($+18.1$ points) and narrows the gap to -7.7. Crucially, PARETOMEM-EMA even surpasses the storage-heavy Replay baseline in old recall (24.8 vs. 18.6) with a far smaller recency bias, demonstrating that stale-gradient Pareto constraints offer more balanced retention than raw replay without storing any past contexts: it is consistent with Takeaway 4.

Old Recall vs. Fairness at T=4096. In Fig. 2 (a), with chunk size 64 and T=4096, PARETOMEM-EMA consistently occupies the Pareto-preferred upper-left region, achieving higher Old@R with lower recency bias than both GRADMEM-REPLAY and PARETOMEM-VANILLA—all without relying on any replay buffer, confirming that the EMA constraint enables a favorable balance between long-term retention and positional fairness.

No-Replay Trade-off across Chunk Sizes. Under T=4096 with chunk sizes from 32 to 512 in the no-replay setting (Fig. 2 (b)), PARETOMEM-EMA traces a dominant trade-off frontier, consistently achieving higher Old@R with comparable or lower recency bias relative to PARETOMEM-VANILLA and GRADMEM-NAIVE SEQUENTIAL across all granularities, confirming stable long-term memory performance without an explicit replay buffer.

Pareto Frontier. In Fig.2 (c), when T=4096 and chunk size 64, GradMem Naive Sequential suffers catastrophic forgetting (Old@R= 1.7), while GradMem Replay maximizes New@R (25.7) yet still lags in Old@R (8.9 vs. 13.7 for ParetoMem-EMA). ParetoMem-EMA attains the best or near-best scores across all three temporal segments (13.7 / 18.6 / 21.2), yielding the most balanced radar area and closely approximating the Pareto frontier, whereas ParetoMem-Vanilla trails by 5–6 points in Old@R and Middle@R, underscoring the critical role of dynamic EMA aggregation.

Ablation. We also provide the empirical analysis by scaling the context size and token size, along with the component ablation in ParetoMem. More details refer to Appendix.F.

4.2 NATURAL LANGUAGE MODELING AND COMMONSENSE REASONING

Setup. Following Behrouz et al. (2025a), we consider baselines pre-trained on FineWeb Penedo et al. (2024) with a training context length of 4,000 tokens, and report results for the 760M/30B and 1.3B/100B settings on held-out language modeling and commonsense reasoning benchmarks. The evaluated baselines include Transformer++, DeepTransfomers built on standard softmax attention Vaswani et al. (2017), Samba Ren et al. (2024), Titans (LMM) Behrouz et al. (2024), and Memora/other recurrent-memory baselines reported in the same experimental protocol Behrouz et al. (2025b; 2024). We adopt the reproduced Transformer++ and DeepTransfomers as the backbone (remarked by gray in the table, indicates the difference between their official reported results), where ParetoMem directly executes for test-time runs (implementation see Appendix.F) to verify whether ParetoMem can be a plug-and-play treatment to endow a model with a test-time memory mechanism without training from scratch. Evaluation follows the original Table 2 setup on WikiText-103 Merity et al. (2017), LAMBADA Paperno et al. (2016), PIQA Bisk et al. (2020), HellaSwag Zellers et al. (2019), WinoGrande Sakaguchi et al. (2021), ARC-Easy / ARC-Challenge Clark et al. (2018), Social IQa Sap et al. (2019), and BoolQ Clark et al. (2019).

Table 2: Performance of ParetoMem and baselines on language modeling and common-sense reasoning tasks.Hybrid models are marked with $*$. chunk sizes are adaptive across benchmarks (see our hyper-parameter disclose). The best results are highlighted.

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg. ↑
760M params / 30B tokens											
Transformer++	25.31	27.64	35.8	70.1	41.9	52.1	60.6	32.3	39.5	60.4	49.09
DeepTransformers	20.24	20.67	36.9	68.6	49.8	52.5	65.9	34.5	40.4	62.2	51.35
Samba*	20.63	22.71	**39.7**	69.2	47.4	52.0	66.9	33.2	39.0	61.2	51.08
Titans (LMM)	20.04	21.96	37.4	69.3	48.5	52.3	66.3	35.8	40.1	62.8	51.56
Memora	22.28	22.31	38.2	67.8	49.3	53.3	63.6	36.1	**40.9**	63.0	51.52
DoT	19.96	20.15	39.0	69.1	50.7	53.1	66.2	37.0	40.3	63.7	52.39
OmegaNet	19.16	20.14	38.7	**69.8**	50.0	53.3	67.8	36.8	39.6	**64.4**	52.56
Atlas	**18.92**	21.01	39.1	69.7	50.2	**53.5**	67.5	37.1	40.7	64.3	52.77
Atlas++	19.04	**20.03**	**39.7**	69.7	**51.1**	53.2	**68.2**	**37.4**	**40.9**	**64.4**	**53.09**
Transformer++(**ParetoMem**)	22.33	23.82	39.1	69.4	45.8	53.1	65.7	35.0	40.9	62.1	51.39
Δ w/ ParetoMem	-2.98	-3.82	+3.3	-0.7	+3.9	+1.0	+5.1	+2.7	+1.4	+1.7	+2.30
DeepTransformers(**ParetoMem**)	**18.76**	**19.23**	**40.4**	**70.7**	**52.6**	**53.8**	**67.1**	**37.8**	**41.3**	**63.9**	**53.45**
Δ w/ ParetoMem	-1.48	-1.44	+3.5	+2.1	+2.8	+1.3	+1.2	+3.3	+0.9	+1.7	+2.10
1.3B params / 100B tokens											
Transformer++	18.70	18.32	42.6	69.6.	50.0	53.1	68.7	35.2	40.5	57.4	52.14
DeepTransformers	15.67	12.63	49.4	73.1	57.1	58.9	69.8	37.7	41.3	61.2	56.06
Mamba2	16.56	12.56	45.7	71.9	55.7	55.2	72.5	37.9	40.2	60.1	54.89
Samba*	16.13	13.29	44.9	70.9	53.4	55.6	68.8	36.2	40.0	62.1	54.00
Titans (LMM)	15.60	11.41	49.1	73.1	56.3	59.8	72.4	40.8	42.1	61.0	56.82
Memora	15.90	12.04	48.7	73.1	56.0	57.4	71.5	37.9	40.2	61.3	55.87
DoT	15.28	11.96	50.1	73.3	57.5	60.4	72.2	41.2	42.7	61.4	57.35
OmegaNet	14.91	11.26	49.7	73.4	57.6	59.7	72.6	40.3	42.4	62.1	57.23
Atlas	14.97	10.98	50.1	**73.9**	57.3	60.2	**72.8**	41.0	**42.9**	**62.8**	57.62
Atlas++	**14.40**	**10.72**	**50.8**	73.5	**59.4**	**61.1**	71.3	**43.7**	42.5	61.9	**58.03**
Transformer++(**ParetoMem**)	16.52	14.72	47.2	73.1	53.7	58.4	71.1	38.4	42.3	61.4	55.70
Δ w/ ParetoMem	-2.18	-3.60	+4.6	+3.5	+3.7	+5.3	+2.4	+3.2	+1.8	+4.0	+3.56
DeepTransformers(**ParetoMem**)	**13.78**	**10.63**	**51.2**	**74.8**	**59.0**	**62.6**	**73.3**	**41.5**	**42.6**	**63.7**	**58.59**
Δ w/ ParetoMem	-1.89	-2.00	+1.8	+1.7	+1.9	+3.7	+3.5	+3.8	+1.3	+2.5	+2.53

Results. Table 2 isolates the claim motivating ParetoMem: what streaming memory lacks is an overwriting-safe write rule, not a new architecture. Applied at test time to frozen pre-trained backbones, ParetoMem improves every metric at both scales—perplexity drops by 1.44–3.82 and average accuracy rises by $+2.70/+2.14$ ($760\text{M}/30\text{B}$) and $+3.45/+2.40$ ($1.3\text{B}/100\text{B}$). The largest margins appear on LAMBADA ($-3.82/-3.60$ ppl, $+3.3/+4.6$ acc), the benchmark that most directly penalizes losing early context, i.e., the signature of catastrophic overwriting rather than of insufficient capacity. Notably, DeepTransformers+ParetoMem attains the lowest perplexities and best averages at both scales ($18.76/19.23$, 53.45; $13.78/10.63$, 58.59), surpassing architecture-coupled test-time memories such as Titans and Atlas++ that must be pre-trained together with their memory modules, whereas ParetoMem adds memory post hoc, leaving architecture, parameters and read path untouched.

5 Conclusion

Catastrophic overwriting is structural to gradient-descent memory: the write sees only the current chunk, while the stream yields only stale evidence about past ones. Recasting streaming writes as online multi-objective optimization on one shared state, ParetoMem projects each write against buffered anchors with staleness-proportional slack plus an EMA summary. It certifies approximate Pareto safety, bounded forgetting, and non-collapse, improving retention–plasticity without replay.

Reproducibility Statement

Appendix E lists hyperparameters and our implementation of each ParetoMem runs on a single A100-80G node. An anonymous code package, built on GradMem's repository (Kuratov et al., 2026), is provided as supplementary material.

AI USE STATEMENT

Large language model assistants (Claude, Anthropic) were used to help design the experimental protocol and gates from the supervisor's specification, to implement to choose hyper-parameters, evaluation and aggregation code, to diagnose failed runs, to draft and edit the text of this paper. All writing content and reference are checked to prevent hallucination. The theoretical statements are verified by authors and AI to promise the rigorousness, and all proofs are proposed and checked by authors. The authors reviewed every sentence and take responsibility for the content.

REFERENCES

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

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025a.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv:2504.13173*, 2025b.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Vivek Chari, Guanghui Qin, and Benjamin Van Durme. KV-Distill: Nearly lossless learnable context compression for LLMs. *arXiv preprint arXiv:2503.10337*, 2025.

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

Lisha Chen, Heshan Fernando, Yiming Ying, and Tianyi Chen. Three-way trade-off in multi-objective learning: Optimization, generalization and conflict-avoidance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023b.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In *International Conference on Learning Representations (ICLR)*, 2025.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5–6), 2012.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, and Christopher Ré. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*, 2025.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32, 2023.

Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Garth A. Gibson, Greg Ganger, and Eric P. Xing. More effective distributed ML via a stale synchronous parallel parameter server. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

Mahdi Karami, Ali Behrouz, Praneeth Kacham, and Vahab Mirrokni. Trellis: Learning to compress key-value memory in attention models. In *Second Conference on Language Modeling (COLM)*, 2025.

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 (PNAS)*, 114(13), 2017.

Yuri Kuratov, Matvey Kairov, Aydar Bulatov, Ivan Rodkin, and Mikhail Burtsev. Gradmem: Learning to write context into memory with test-time gradient descent. *arXiv preprint arXiv:2603.13875*, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. FAMO: Fast adaptive multitask optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Yansheng Mao, Yufei Xu, Jiaqi Li, Fanxu Meng, Haotong Yang, Zilong Zheng, Xiyuan Wang, and Muhan Zhang. LIFT: Improving long context understanding of large language models through long input fine-tuning. *arXiv preprint arXiv:2502.14644*, 2025.

Lingrui Mei, Shenghua Liu, Yiwei Wang, Yuyao Ge, Baolong Bi, Jiayu Yao, Jun Wan, Ziling Yin, Jiafeng Guo, and Xueqi Cheng. Gated differentiable working memory for long-context language modeling. *arXiv preprint arXiv:2601.12906*, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Jesse Mu, Xiang Lisa Li, and Noah D. Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 101: 15, 2024.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. Memgpt: towards llms as operating systems. 2023.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

Guilherme Penedo, Hynek Kydlicek, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 2024.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Haowen Hou, Janna Lu, Stella Biderman, et al. RWKV-7 "goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

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 (CVPR)*, 2017.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems*, 24, 2011.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*, 2024.

Ivan Rodkin, Yuri Kuratov, Aydar Bulatov, and Mikhail Burtsev. Associative recurrent memory transformer. *arXiv preprint arXiv:2407.04841*, 2024.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning (ICML)*, 2021.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685, 2024.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. In *International Conference on Machine Learning*, pp. 57503–57522. PMLR, 2025.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Qi Zhang, Peiyao Xiao, Shaofeng Zou, and Kaiyi Ji. Mgda converges under generalized smoothness, provably. In *International Conference on Learning Representations*, volume 2025, pp. 10425–10456, 2025.

Zhi Zhang, Jiayi Shen, Congfeng Cao, Gaole Dai, Shiji Zhou, Qizhe Zhang, Shanghang Zhang, and Ekaterina Shutova. Proactive gradient conflict mitigation in multi-task learning: A sparse training perspective. *arXiv preprint arXiv:2411.18615*, 2024.

Category	Notation	Type/Space	Meaning
Model	f_θ	–	Pretrained causal language model with frozen parameters θ.
Model	θ	$\mathbb{R}^{\lvert\theta\rvert}$	Frozen base-model parameters; not updated at any stage.
Inference	C	Token sequence	A context provided as input to the model.
Inference	Q	Token sequence	A query posed at read time.
Inference	Y	Token sequence	Target output to be predicted.
Memory	M	$\mathcal{K} \subseteq \mathbb{R}^P$	Memory-token state; the only updatable parameters at test time.
Memory	d	$\mathbb{N}^+$	Dimensionality of the memory-token parameter space.
Memory	$\mathcal{K}$	Compact convex $\subseteq \mathbb{R}^P$	Feasible set for memory tokens (used in projected updates and smoothness assumptions).
Memory	$\Pi_{\mathcal{K}}(\cdot)$	$\mathbb{R}^P \to \mathcal{K}$	Euclidean projection onto the feasible set $\mathcal{K}$.
Memory	M_0	$\mathcal{K}$	Initial memory state before any chunk is written (single-context GradMem).
Memory	M^k	$\mathcal{K}$	Memory state after the k-th intra-chunk gradient step in GradMem, $k = 0, \ldots, K$.
Memory	$\widehat{M} = M^K$	$\mathcal{K}$	Final memory after completing all K GradMem write steps for a single bounded context.
Stream	$C_1, C_2, \ldots, C_T$	Token sequences	Ordered sequence of context chunks arriving in the stream.
Stream	T	$\mathbb{N}^+ \cup \{+\infty\}$	Total number of chunks in the stream (may be unknown or unbounded).
Stream	$x_{t,i}$	Token	The i-th token of chunk C_t.
Stream	$\lvert C_t\rvert$	$\mathbb{N}^+$	Number of tokens in chunk C_t.
Stream	$x_{t,<i}$	Token sequence	Prefix tokens $(x_{t,1}, \ldots, x_{t,i-1})$ within chunk C_t.
Stream	M_t	$\mathcal{K}$	Memory state at the beginning of time step t (before writing chunk C_t).
Stream	M_{T+1}	$\mathcal{K}$	Final memory used at read time after all T chunks have been written.

Table 3: Notations (Model, Inference, Memory, Data stream) used throughout the paper.

A Notations and Related Work

In this section, we provide the relevant literature and the notations used in this paper.

Notations

The notations frequently used throughout this paper are summarized in Table 3 - 7 .

Relate Work

contextual modeling and efficient attention. A first line of work extends the effective attention window through positional interpolation or extrapolation (Chen et al., 2023b; Su et al., 2024) and hardware-aware attention kernels (Dao et al., 2022; Dao, 2023; Shah et al., 2024). These methods push the usable context length further but still inherit quadratic attention costs and are not designed for unbounded streams. The second family, retrieval-augmented generation (RAG) (Lewis et al., 2020; Gao et al., 2023), maintains an external index and retrieves relevant snippets at query time. While effective for lookup-heavy workloads, RAG introduces separate retrieval infrastructure and additional query-time latency, and can struggle when answering requires global cross-segment synthesis rather than localized retrieval. MagicPIG (Chen et al., 2025) takes a different angle within the

efficient-attention family: rather than evicting tokens, it uses Locality Sensitive Hashing to sample attention contributions with theoretical guarantees, substantially reducing KV-cache computation while preserving accuracy across diverse downstream tasks.

Compressed and recurrent memory. The third family encodes history into a bounded state rather than attending to raw tokens. Recurrent memory transformers such as RMT (Bulatov et al., 2022) and its extension ARMT (Rodkin et al., 2024) pass trainable memory tokens between segments, enabling processing of sequences far beyond the native context window. Infini-Attention (Munkhdalai et al., 2024) augments standard attention with a compressive memory updated and read via a learned gating mechanism. Linear transformers and delta-rule networks (Schlag et al., 2021; Yang et al., 2024; Peng et al., 2025) implement recurrent updates that approximate key–value storage but with a fixed forward-only update rule. Prompt-compression methods such as Gist (Mu et al., 2023) and AutoCompressors (Chevalier et al., 2023) distil long inputs into soft summary tokens in a single forward pass. On the KV-cache side, KV-Distill (Chari et al., 2025) trains a parameter-efficient adaptor to compress long KV caches into significantly shorter representations via a KL-type distillation objective. Trellis (Karami et al., 2025) replaces the standard KV cache with a fixed-size memory and learns a two-pass recurrent compression mechanism driven by online gradient descent with a learnable forget gate, enabling dynamic KV compression at test time. Cartridges (Eyuboglu et al., 2025) push offline corpus distillation further by training a compact KV cache with a self-study recipe that generates synthetic conversations and applies a context-distillation objective, achieving over $26\times$ throughput improvements. InfLLM (Xiao et al., 2024) addresses streaming long-sequence inference without any fine-tuning by storing distant context in external memory units and using an efficient token-relevance lookup for attention, enabling training-free generalization to sequences far beyond the pre-training length. All of these approaches rely on *fixed or forward-only* update rules for the write phase: once the memory is emitted the write operation cannot measure or correct encoding errors, and there is no per-example objective that is iteratively minimized for the given context.

Gradient-descent and test-time-training memory. A parallel line of work treats memory as an adaptive state updated by gradient steps or fast-weight rules. Test-Time Training (TTT) layers (Sun et al., 2025) maintain a per-layer state updated online via a self-supervised loss, providing gradient-driven adaptation at each token. Titans (Behrouz et al., 2024) and Atlas (Behrouz et al., 2025a) extend this idea to longer horizons with learnable surprise-gated and routing mechanisms. LIFT (Mao et al., 2025) fine-tunes model parameters directly on the long input at test time using a Gated Memory adapter and LLM-generated synthetic tasks, compressing the context into weights so that short-context LLMs can answer queries with linear rather than quadratic cost in input length. TTT-E2E (Tandon et al., 2025) formulates contextual language modeling as a continual-learning problem and proposes an end-to-end test-time training scheme where a standard Transformer with sliding-window attention continues learning via next-token prediction on the context, compressing it into model weights; the resulting method scales with context length as well as full attention while maintaining constant inference latency. GDWM (Mei et al., 2026) reframes test-time adaptation as a budget-constrained memory consolidation problem: a Write Controller estimates the *contextual utility* of each chunk—measuring how much it depends on long-range context—and allocates gradient steps accordingly under a coverage constraint, achieving comparable performance with $4\times$ fewer write steps than uniform baselines. GradMem (Kuratov et al., 2026) is most directly related to our work: it writes an entire context segment into a small set of prefix memory tokens by running a few steps of test-time gradient descent on a self-supervised reconstruction loss, keeping all base model parameters frozen. GradMem demonstrates that gradient-based writing stores substantially more information per memory token than forward-only alternatives, and that additional write steps reliably improve capacity. However, GradMem is formulated for a single bounded context. In the streaming regime—where an unbounded chunk sequence is encoded into a fixed-size memory with no access to raw past chunks—the gradient of a new chunk can conflict with the gradients of previously written chunks, causing *catastrophic overwriting*. ParetoMem addresses this failure mode by augmenting the GradMem write rule with staleness-aware Pareto projections, while preserving GradMem's architecture-agnostic, lightweight design.

Continual learning. Catastrophic overwriting is related in spirit to catastrophic forgetting in continual learning (CL) (Kirkpatrick et al., 2017). Replay-based methods (Rebuffi et al., 2017) store or generate past examples to prevent forgetting, but raw chunk replay is unavailable in our streaming setting. GEM (Lopez-Paz & Ranzato, 2017) and A-GEM (Chaudhry et al., 2019) use episodic memory of past gradients to constrain the current update, requiring hard inequality constraints en-

forced with *exact* current gradients. Architecture-based methods such as Progressive Neural Networks (Rusu et al., 2016) allocate task-specific parameters that are incompatible with our single shared memory state. Regularization methods such as EWC (Kirkpatrick et al., 2017) penalize changes to parameters important for past tasks, which requires importance scores that cannot be estimated without access to past data. All three classes are therefore difficult to apply in streaming memory without replay, task boundaries, or fresh estimates of past objectives. ParetoMem differs from GEM and A-GEM in two key respects: it relaxes the hard Pareto constraint to a *soft* one that accounts for gradient staleness, and it requires no access to raw past data, operating solely on buffered stale gradients.

Multi-objective optimization and gradient surgery. Our formulation of streaming memory writing as multi-objective optimization (MOO) draws on a well-developed literature on gradient-based Pareto methods. The Multiple-Gradient Descent Algorithm (MGDA) (Désidéri, 2012) computes a common descent direction that reduces all objectives simultaneously, converging to a Pareto-stationary point; Sener & Koltun (2018) showed that MGDA admits an efficient Frank–Wolfe solution and applied it to multi-task learning. PCGrad (Yu et al., 2020) takes a simpler approach: it projects each task's gradient onto the normal plane of any conflicting task gradient, removing destructive interference. CAGrad (Liu et al., 2021) generalizes both methods by seeking an update that maximizes the minimum per-task improvement within a local ball around the average gradient, and subsumes GD and MGDA as special cases. FAMO (Liu et al., 2023) achieves balanced loss reduction across tasks in $\mathcal{O}(1)$ space and time by maintaining a per-task exponential moving average of loss values, avoiding the $\mathcal{O}(k)$ gradient storage required by prior gradient-surgery methods; this compact-history idea resonates with ParetoMem's own EMA gradient buffer. Recent theoretical work Chen et al. (2023a) has studied the convergence of stochastic MGDA variants under more realistic smoothness conditions, including settings relevant to transformers. Zhang et al. (2024) further show that updating only a sparse subset of parameters proactively reduces the incidence of gradient conflict and can be combined with existing gradient surgery methods. Crucially, all batch MOO solvers including MGDA, PCGrad, and CAGrad assume that all objective gradients are computed *at a shared parameter state*. This assumption is violated in streaming memory: buffered gradients are stale, having been computed when earlier chunks were written. The phenomenon of gradient staleness has been studied extensively in asynchronous distributed training (Ho et al., 2013; Recht et al., 2011), where delayed worker updates degrade convergence; however, those works focus on asynchronous parameter-server architectures with bounded delay, not on the multi-objective projection problem with smoothness-based staleness bounds that arises in streaming memory. ParetoMem extends the gradient-surgery viewpoint to the stale-gradient regime by introducing smoothness-based tolerance margins that convert stale buffered constraints into approximate Pareto constraints with respect to the unobserved current gradients, a challenge absent from all existing MOO and gradient-surgery methods.

B ALGORITHM

ParetoMem inherits GradMem's meta-learning substrate wholesale (frozen slow weights, memory-as-fast-state, reconstruction objective, few-step adaptation, WRITE/READ split). Its sole addition is a streaming-preservation layer atop the inner update: a sliding gradient buffer with staleness-aware tolerances, Pareto projection, and optional EMA summarization of historical gradients. More specifically,

Parameter Split and Initialization. GradMem freezes the base LM parameters θ and treats memory tokens M as the sole fast-adapted variables. A shared initialization M_0 is meta-learned so that few gradient steps suffice for effective writing. ParetoMem inherits this split entirely: the backbone stays frozen and only the memory state is updated online. The streaming memory is initialized from GradMem's meta-learned M_0 (or its evolved state), preserving the learned prior.

Inner-Loop Objective. Both methods share the same self-supervised WRITE objective—autoregressive reconstruction of the context:

$$\mathcal{L}_{\text{write}}(M; C) = -\sum_i \log f_\theta(x_i \mid [M; x_{<i}]).$$

ParetoMem applies this loss chunk-wise as $\ell_t(M) := \mathcal{L}_{\text{write}}(M; C_t)$, without modification.

Category	Notation	Type/Space	Meaning
Loss & Gradient	$\mathcal{L}_{\text{write}}(M; C)$	$\mathbb{R}$	GradMem self-supervised write loss measuring reconstruction of C conditioned on M.
Loss & Gradient	$\ell_t(\cdot)$	$\mathcal{K} \to \mathbb{R}$	Write loss of chunk C_t as a function of the memory state: $\ell_t(M) := \mathcal{L}_{\text{write}}(M; C_t)$.
Loss & Gradient	g_t	$\mathbb{R}^P$	Fresh write gradient of the current chunk: $g_t := \nabla_M \ell_t(M_t)$.
Loss & Gradient	$\widehat{g}_s$	$\mathbb{R}^P$	Stale (buffered) gradient of past chunk C_s: $\widehat{g}_s := \nabla_M \ell_s(M_s)$.
Loss & Gradient	$\nabla_M \ell_s(M_t)$	$\mathbb{R}^P$	True (unobserved) current gradient of chunk s at present memory M_t; unavailable in streaming.
Update	u_t	$\mathbb{R}^P$	Update direction chosen by ParetoMem at time t.
Update	η_t	$\mathbb{R}_{>0}$	Step size (learning rate) at streaming time step t.
Update	α	$\mathbb{R}_{>0}$	Step size used in the single-context GradMem write loop (distinct from streaming η_t).
Update	K	$\mathbb{N}^+$	Number of intra-chunk gradient steps in the original (non-streaming) GradMem write.
Buffer & EMA	$\mathcal{B}_t$	Set of tuples	Gradient buffer at time t: $\{(\widehat{g}_s, s, r_s)\}$ for protected historical chunks.
Buffer & EMA	W	$\mathbb{N}^+$	Maximum size of the sliding gradient buffer (window length).
Buffer & EMA	r_s	$\mathbb{R}_{\geq 0}$	Running memory-drift statistic stored with $\widehat{g}_s$, used to estimate $\xi_{t,s}$.
Buffer & EMA	$\bar{g}_t$	$\mathbb{R}^P$	EMA summary direction of evicted gradients at time t.
Buffer & EMA	β	$[0, 1)$	EMA decay coefficient.
Buffer & EMA	$\widehat{g}_{\text{evict}}$	$\mathbb{R}^P$	The oldest gradient being evicted from the buffer when full.

Table 4: Notations (Loss, Gradients, Update, Inference, Buffer, EMA) used throughout the paper.

Inner-Loop Update Rule. GradMem performs vanilla gradient descent on memory: $M^{k+1} = M^k - \alpha \nabla_M \mathcal{L}_{\text{write}}$. ParetoMem replaces this raw step with a Pareto-corrected direction u_t derived from a staleness-aware projection over buffered historical gradients, then updates $M_{t+1} = \Pi_{\mathcal{K}}(M_t - \eta_t u_t)$. This is the central modification: a history-aware, interference-controlled variant of the same gradient-based fast adaptation.

Per-Example Feedback and Iterative Correction. GradMem's key meta-learning insight is that the reconstruction loss provides explicit per-sample feedback, enabling iterative error correction during WRITE. ParetoMem retains this feedback mechanism but augments it with approximate preservation constraints from prior chunks, so that writing new information does not catastrophically overwrite old content.

Outer-Loop Meta-Training. GradMem meta-trains θ and M_0 by backpropagating through the unrolled WRITE steps (MAML-style second-order gradients), optimizing the downstream READ loss $\mathcal{L}_{\text{task}}(\hat{M}, Q, Y)$. ParetoMem does not alter this outer loop; it modifies only the test-time/streaming WRITE dynamics.

WRITE/READ Decomposition. Both methods share the two-phase protocol: WRITE encodes context into memory; READ predicts from $[M; Q]$ alone with context removed. ParetoMem's novelty lies entirely within the WRITE phase—specifically, in how the memory update is computed under repeated online writes across a stream of chunks.

Category	Notation	Type/Space	Meaning
Loss & Gradient	$\bar{\ell}_t(\cdot)$	$\mathcal{K} \to \mathbb{R}$	Virtual objective summarised by the EMA entry, $\bar{\ell}_t := \sum_{s \in \mathcal{A}_t} w_{t,s} \ell_s$; L-smooth with $\|\nabla \bar{\ell}_t\| \le G$ (Prop. 8).
Loss & Gradient	G	$\mathbb{R}_{>0}$	Uniform bound on write-gradient norms on $\mathcal{K}$: $\|\nabla \ell_s(M)\| \le G$ (Assumption D.2); realized by the gradient-clipping threshold, and the only constant besides L entering the bounds.
Loss & Gradient	P	$\mathbb{N}^+$	Number of writable memory parameters (flattened), $P = md$ for prefix memory and $P = \sum_{\ell \in \mathcal{L}_m} m_\ell d_\ell$ for layer-wise memory; all directions live in $\mathbb{R}^P$.
Update	$u_t^\star$	$\mathcal{C}_t$	Exact Pareto direction, $u_t^\star = \Pi_{\mathcal{C}_t}(g_t)$; unique since $\mathcal{C}_t$ is closed, convex and contains 0.
Update	κ_t	$[0, 1]$	Scalar rescaling factor of the exact-feasibility safeguard, $u_t = \kappa_t u$ (Lemma E.5).
Update	N	$\mathbb{N}^+$	Stream depth, i.e. total number of write steps $N = TK$; guarantees are stated per write step.
Update	$\bar{\rho}$	$\mathbb{R}_{>0}$	mean displacement, used only to calibrate λ.
Update	$\rho_{\min}$	$\mathbb{R}_{>0}$	per-step displacement floor, $D_{s+k,s} \ge k\rho_{\min}$; used only in Cor. 7(ii); estimated as a low quantile (App. F.3).
Buffer & EMA	$\mathcal{A}_t$	$\subseteq \{1, \dots, t-1\}$	Set of (possibly unboundedly many) evicted chunks summarised by the single EMA entry at time t.
Buffer & EMA	$w_{t,s}$	$[0, 1]$	$\dots; = (1-\beta)\beta^k$ for the k-th most recent eviction, β^{m-1} for the oldest; $\sum_s w_{t,s} = 1$.
Buffer & EMA	$\widetilde{W}$ & $= KW$	$\mathbb{N}^+$	protection window in write steps.
Buffer & EMA	$H_0,\ \tau$	$\mathbb{N}^+$	calibration budget (write steps) and dimensionless slack; $\lambda = \tau G \|\widehat{g}\|_{\mathrm{ref}} / (\bar{\rho} H_0)$, $H_{\max} = H_0/\tau$.
Buffer & EMA	$\mu_{t,s}$	$\mathbb{R}_{>0}$	KKT multipliers of $\Pi_{\mathcal{C}_t}$; $\langle g_t - u_t^\star,\ u_t^\star \rangle = \sum_s \mu_{t,s} \xi_{t,s}$.
Buffer & EMA	H_s	$\mathbb{N}^+ \cup \{\infty\}$	Protection horizon of buffer entry s: the smallest age at which its relaxed constraint becomes vacuous (Cor. 7).
Buffer & EMA	$H_{\max}$	$\mathbb{N}^+$	$\lceil G^2/(\lambda \rho_{\min}) \rceil$ write steps; a FIFO of $W > H_{\max}/K$ chunks is provably redundant.

Table 5: Notations (Loss, Gradients, Update, Inference, Buffer, EMA) used throughout the paper.

The pipeline of ParetoMem can be found in Algorithm.1 : For each incoming chunk, the method computes the current GradMem write gradient, constructs staleness-aware constraints from the buffer and EMA summary, projects the gradient, updates the memory tokens, and then updates the gradient buffer. The raw chunk is discarded after its gradient has been stored.

C Standing Setup and Definitions

Throughout App. C–E, s denotes the write step at which the anchor of a chunk is taken (its first step), and windows are in write steps: a FIFO of W chunks corresponds to $\widetilde{W} = KW$ steps. Notation follows Tables 3–7. To avoid duplication we restate here only those objects that carry analytical content, i.e. that are used inside the proofs beyond mere naming. Throughout, all losses

Category	Notation	Type/Space	Meaning
Tolerance & Staleness	$\xi_{t,s}$	$\mathbb{R}_{\geq 0}$	Staleness-aware tolerance margin for the buffered constraint of chunk s at time t.
Tolerance & Staleness	$D_{t,s}$	$\mathbb{R}_{\geq 0}$	Accumulated memory drift from time s to t: $D_{t,s} := \sum_{j=s}^{t-1} \eta_j \|u_j\|$.
Tolerance & Staleness	L	$\mathbb{R}_{>0}$	Lipschitz smoothness constant of each write loss ℓ_s w.r.t. M on $\mathcal{K}$ (Assumption D.1).
Projection	$\mathcal{C}_t$	Convex $\subseteq \mathbb{R}^P$	Feasible set for buffered Pareto projection: $\mathcal{C}_t := \{u \in \mathbb{R}^P : \langle u, \widehat{g}_s \rangle \geq -\xi_{t,s},\ s \in \mathcal{B}_t\}$.
Projection	δ	$\mathbb{R}_{>0}$	Small numerical stabilizer in the sequential half-space projection step.
General	$\langle \cdot, \cdot \rangle$	Bilinear form	Standard Euclidean inner product in $\mathbb{R}^P$.
General	$\| \cdot \|$	$\mathbb{R}_{\geq 0}$	Euclidean (ℓ_2) norm in $\mathbb{R}^P$, unless otherwise specified.
General	$O(\cdot)$	–	Standard asymptotic big-O notation.

Table 6: Notations (Tolerance, Staleness, Projection, General) used throughout the paper.

Category	Notation	Type/Space	Meaning
Tolerance & Staleness	λ	$\mathbb{R}_{>0}$	The single tolerance hyper-parameter, $\xi_{t,s} = \lambda D_{t,s}$ (Rule D3); order-free choice $\lambda = \Theta(LG)$.
Tolerance & Staleness	$\bar{\xi}_t$	$\mathbb{R}_{\geq 0}$	Tolerance attached to the EMA constraint, $\bar{\xi}_t = c_t \lambda \bar{D}_t$.
Tolerance & Staleness	$\bar{D}_t$	$\mathbb{R}_{\geq 0}$	Weighted average staleness of the summarised chunks, $\bar{D}_t = \sum_{s \in \mathcal{A}_t} w_{t,s} D_{t,s}$.
Projection	$\Pi_{\mathcal{C}_t}, \Pi_{\mathcal{K}}$	$\mathbb{R}^P \to \mathbb{R}^P$	Euclidean projections onto the tolerance set $\mathcal{C}_t$ (the WRITE filter) and onto the admissible memory set $\mathcal{K}$ (the optional safeguard clip).
Diagnostics	ε_t	$\mathbb{R}_{\geq 0}$	Measured residual constraint violation of the realized direction, $\varepsilon_t = \max_{s \in \mathcal{B}_t}(-\xi_{t,s} - \langle u_t, \widehat{g}_s \rangle)_+$; computable in $O(WP)$, and $\varepsilon_t = 0$ for $u_t = u_t^\star$ or under Lemma E5.
Diagnostics	α_t	$[0, 1]$	…; the upper bound $\|u_t^\star\| \leq \sqrt{1 - \alpha_t^2}\, \|g_t\|$ is strict whenever a constraint with $\xi_{t,s} > 0$ is active.
Diagnostics	$\widehat{\alpha}_t$	$[0, \infty)$	Realized conflict coefficient $\widehat{\alpha}_t = \|u_t - g_t\| / \|g_t\|$, logged at every write; equals α_t for the exact projection, and $\widehat{\alpha}_t \to 1$ signals memory freezing.

Table 7: Notations (Tolerance, Staleness, Projection, Diagnostics) used throughout the paper.

are functions of the memory state alone, since the base language-model parameters θ are frozen; we write $\ell_s(M) := L_{\text{write}}(M; C_s)$ and $g_t := \nabla \ell_t(M_t)$, $\widehat{g}_s := \nabla \ell_s(M_s)$.

Definition C.1 (Realized streaming update and accumulated drift) At time t the writer observes C_t, computes $g_t = \nabla \ell_t(M_t)$, selects an update direction u_t, and updates

$$M_{t+1} = M_t - \eta_t u_t, \qquad \eta_t > 0. \tag{9}$$

For $t > s$ we write the *accumulated memory drift* between the two times as

$$D_{t,s} := \sum_{j=s}^{t-1} \eta_j \|u_j\|.$$

Algorithm 1 ParetoMem streaming WRITE

Require: frozen f_θ, memory M, steps η_t, buffer size W, decay β, slack λ

$\mathcal{B} \leftarrow \emptyset$; $\bar{g} \leftarrow \varnothing$; $\bar{r} \leftarrow 0$
for chunk $c = 1, \dots, T$ **do**
  for $k = 1, \dots, K$ **do** $\{t :=$ global step index$\}$
    $g_t \leftarrow \nabla_M \mathcal{L}_{\text{write}}(M; C_c)$
    if $k = 1$ **then**
      push $(\widehat{g}_c := g_t,\ r_c := 0)$ into $\mathcal{B}$
    end if
    $u \leftarrow g_t$
    for each $(\widehat{g}_s, r_s) \in \mathcal{B} \cup \{(\bar{g}, \bar{r})\}$ with $\lambda r_s < G\|\widehat{g}_s\|$ **do**
      $\xi \leftarrow \lambda r_s$; **if** $\langle u, \widehat{g}_s \rangle < -\xi$: $u \leftarrow u + \frac{-\xi - \langle u, \widehat{g}_s \rangle}{\|\widehat{g}_s\|^2 + \delta} \widehat{g}_s$
    end for
    (optional) rescale $u \leftarrow \kappa_t u$ (Lemma E.5) so that $\varepsilon_t = 0$
    $M \leftarrow \Pi_{\mathcal{K}}(M - \eta_t u)$; $\Delta \leftarrow \eta_t \|u\|$
    $r_s \leftarrow r_s + \Delta$ for all $s \in \mathcal{B}$; $\bar{r} \leftarrow \bar{r} + \Delta$
  end for
  if $|\mathcal{B}| > W$ **then**
    evict oldest $(\widehat{g}_{\text{ev}}, r_{\text{ev}})$
    $\bar{g} \leftarrow \widehat{g}_{\text{ev}}$, $\bar{r} \leftarrow r_{\text{ev}}$ **if** first eviction; **else** $\bar{g} \leftarrow \beta \bar{g} + (1 - \beta)\widehat{g}_{\text{ev}}$, $\bar{r} \leftarrow \beta \bar{r} + (1 - \beta) r_{\text{ev}}$
  end if
  discard C_c
end for

We let $\mathcal{K} \subseteq \mathbb{R}^d$ denote any convex set containing the realized trajectory $\{M_t\}$; $\mathcal{K}$ enters the analysis only through Assumption D.1 and is not part of the algorithm.

Remark C.1 (on the optional projection $\Pi_{\mathcal{K}}$). Algorithm 1 optionally applies a safeguard projection $M_{t+1} = \Pi_{\mathcal{K}}(M_t - \eta_t u_t)$ to keep the memory state in a prescribed box. All results below are stated for 9 and hold verbatim whenever the safeguard is inactive, i.e. $M_t - \eta_t u_t \in \mathcal{K}$. If the safeguard is active, writing $M_{t+1} = M_t - \eta_t u_t + e_t$ with $\|e_t\| \le \eta_t \|u_t\|$ introduces an additional first-order term $\langle \nabla \ell_s(M_t), e_t \rangle$ that is only bounded by $O(\eta_t)$. We therefore treat 9 as the analyzed update.

Remark C.1 (Clipping). Gradient clipping multiplies g_t by a scalar $c_t \in (0, 1]$. Since $\Pi_{\mathcal{C}_t}$ is positively homogeneous only for $\lambda = 0$, we store the clipped gradient as anchor and take G to be the clipping threshold; then Assumption D.2 holds exactly, and clipping is equivalent to the step-size rescaling $\eta_t \leftarrow c_t \eta_t$, which leaves all statements (in terms of η_t) unchanged.

Definition C.2 (Tolerance set of buffered Pareto constraints). Let $\mathcal{B}_t$ be the index set of chunks whose stale gradients are held in the buffer at time t (including the virtual EMA entry, when used; $t = s + 1, \dots, s + \widetilde{W}$), and let $\xi_{t,s} \ge 0$ be the tolerance attached to entry s. The *tolerance set* is

$$\mathcal{C}_t := \left\{ u \in \mathbb{R}^d \ :\ \langle u, \widehat{g}_s \rangle \ge -\xi_{t,s} \ \ \forall s \in \mathcal{B}_t \right\} = \bigcap_{s \in \mathcal{B}_t} H_{t,s}, \qquad H_{t,s} := \{ u : \langle u, \widehat{g}_s \rangle \ge -\xi_{t,s} \}.$$

The idealized ParetoMem direction is the Euclidean projection

$$u_t^\star := \Pi_{\mathcal{C}_t}(g_t) = \arg\min_{u \in \mathcal{C}_t} \tfrac{1}{2} \|u - g_t\|^2. \tag{10}$$

Since $\xi_{t,s} \ge 0$ implies $0 \in H_{t,s}$ for every s, the set $\mathcal{C}_t$ is a nonempty closed convex cone-like set containing the origin, so $u_t^\star$ exists and is unique. *Whenever a FIFO buffer of size W is used, we adopt the convention that chunk s satisfies $s \in \mathcal{B}_t$ exactly for $t = s + 1, \dots, s + W$.*

Definition C.3 (Residual constraint violation). The implementation of Section 3.3 replaces 10 by one or two cyclic passes of relaxed half-space projections, so the realized direction u_t need not lie in $\mathcal{C}_t$. We measure its infeasibility by the *residual violation*

$$\varepsilon_t := \max_{s \in \mathcal{B}_t} \big(-\xi_{t,s} - \langle u_t, \widehat{g}_s \rangle \big)_+ \ \ge 0, \qquad (x)_+ := \max\{x, 0\},$$

so that, by construction and with no further assumption,

$$\langle u_t, \widehat{g}_s \rangle \;\geq\; -\big(\xi_{t,s} + \varepsilon_t\big) \qquad \forall s \in \mathcal{B}_t. \tag{11}$$

The quantity ε_t is computable at run time in $O(Wd)$; $\varepsilon_t = 0$ whenever $u_t = u_t^\star$ or when the safeguard of Lemma E.5 is applied.

Definition C.4 (Realized conflict coefficient). For $g_t \neq 0$ define the *realized conflict coefficient*

$$\widehat{\alpha}_t := \frac{\|u_t - g_t\|}{\|g_t\|} \in [0, \infty), \qquad \text{and} \qquad \alpha_t := \frac{\operatorname{dist}(g_t, \mathcal{C}_t)}{\|g_t\|} = \frac{\|u_t^\star - g_t\|}{\|g_t\|}.$$

Both are observable during the WRITE phase. By Lemma E.2(i), $\alpha_t \in [0, 1]$ always, and $\widehat{\alpha}_t = \alpha_t$ when the exact projection 10 is used. The value $\widehat{\alpha}_t$ measures how much the Pareto constraints deflect the current write gradient: $\widehat{\alpha}_t = 0$ means the incoming gradient is already Pareto-safe, whereas $\widehat{\alpha}_t \to 1$ corresponds to $u_t \to 0$, i.e. memory freezing. *$\widehat{\alpha}_t$ is a measured diagnostic, not an assumption.*

D Assumptions and Design Rules

We separate three kinds of statements that are often conflated: (i) a substantive assumption about the objective, which could in principle fail (Assumption D.1); (ii) a boundedness condition enforced by the implementation (Assumption D.2); and (iii) a *design rule* that we impose on our own algorithm and that therefore holds by construction (Rule D.3). Only D.1 is a genuine modelling hypothesis.

Assumption D.1 (Smooth write losses). Every write loss ℓ_s is differentiable and L-smooth on $\mathcal{K}$:

$$\|\nabla \ell_s(M) - \nabla \ell_s(M')\| \leq L\|M - M'\| \qquad \forall M, M' \in \mathcal{K}.$$

This is the only assumption on the objective, and it is the sole place where L enters. Because the base model is frozen and $\mathcal{K}$ is bounded, $M \mapsto \ell_s(M)$ is a composition of a fixed smooth network with a bounded soft-token input; L-smoothness on a bounded set is the standard regularity condition used for first-order analysis in this regime. *Crucially, L is not needed by the algorithm* (see Rule D.3); it appears only in the constants of the bounds.

Assumption D.2 (Bounded write gradients). There exists $G < \infty$ with

$$\|\nabla \ell_s(M)\| \leq G \qquad \text{for all } s \text{ and all } M \in \mathcal{K}.$$

In particular $\|g_t\| \leq G$. Together with Lemma E.2(ii) this *implies* $\|u_t\| \leq G$ and $D_{t,s} \leq G \sum_{j=s}^{t-1} \eta_j$; we therefore do not assume a separate bound on the update directions. In the implementation, G is realized by the gradient-clipping threshold already applied to the flattened memory gradient (realised exactly by clipping, see Remark C.2).

Rule D.3 (Staleness-proportional tolerance). ParetoMem sets the tolerance of buffered entry s to

$$\xi_{t,s} := \lambda\, D_{t,s} = \lambda \sum_{j=s}^{t-1} \eta_j \|u_j\|, \qquad \lambda > 0, \tag{12}$$

where λ is the single tolerance hyper-parameter. Equation 12 holds by construction, and $D_{t,s}$ is maintained incrementally as the statistic r_s of the buffer at cost $O(1)$ per entry.

Rule D.3(b) (Calibration).

$$\lambda := \frac{\tau\, G\, \|\widehat{g}\|_{\mathrm{ref}}}{\bar{\rho}\, H_0}, \qquad H_0 \text{ (write steps)},\ \tau \in (0, 1],$$

so that an anchor of reference norm reaches slack $\tau\, G \|\widehat{g}\|_{\mathrm{ref}}$ at age H_0 and becomes vacuous at age $H_{\max} = H_0/\tau$ (Cor. 7). λ is fixed once per task and does not depend on W.

Remark (What we do *not* assume). Four hypotheses used in earlier versions are eliminated rather than weakened.

- *Bounded update directions.* Not assumed; derived in Lemma E.2(ii) for both the exact QP and the sequential-projection solver.

- *A step-size schedule.* Not assumed. Only $\eta_t > 0$ is a convention (Def. C.1).
- *Exact projection.* Not assumed. All results are stated through the measured residual ε_t of Def. C.3, hence apply to the algorithm as implemented; Lemma E.5 further gives an $O(WP)$ safeguard achieving $\varepsilon_t = 0$ exactly.
- *Compatibility of the current gradient.* Not assumed. The former compatibility parameter is replaced by the *measured* coefficient $\widehat{\alpha}_t$ of Def. C.4, which makes Theorem 6 unconditional.

Finally, we record once and for all the two bookkeeping bounds that every proof below invokes under the name (P). They are *consequences*, not hypotheses: the norm bound follows from the fact that the tolerance set contains the origin, which holds by construction whenever $\xi_{t,s} \geq 0$.

Property 2 (Norm and drift bookkeeping, referred to as (P)). Let Assumption D.2 hold and let the realized direction u_t be produced from $u^{(0)} = g_t$ by *any* of the three solvers used in this paper — the exact projection 6, a finite number of cyclic relaxed half-space passes 8, or the rescaling safeguard of Lemma E.5. Then for every t, and for every $s < t$,

$$\|u_t\| \;\leq\; \|g_t\| \;\leq\; G, \qquad \text{and} \qquad D_{t,s} \;=\; \sum_{j=s}^{t-1} \eta_j \|u_j\| \;\leq\; G \sum_{j=s}^{t-1} \eta_j.$$

Proof. The first chain is Lemma E.2(ii) — which covers all three solvers, since each is a finite composition of relaxed projections onto half-spaces containing the origin — combined with $\|g_t\| = \|\nabla \ell_t(M_t)\| \leq G$ (Assumption D.2). Substituting $\|u_j\| \leq G$ into Definition C.1 gives the second. □

Property 2 is what allows the bound on update norms to be *deleted* from the assumption list rather than weakened: the constants G_u and c_ξ of earlier versions are subsumed by the single constant G, and no separate hypothesis on $\|u_t\|$ is made anywhere in this paper.

E Takeaways: Formal Statements and Proofs

E.1 Preliminary lemmas

Lemma E.1 (Memory drift bound). Under 9, for all $t > s$, $\|M_t - M_s\| \leq D_{t,s}$.

Proof. $M_t - M_s = -\sum_{j=s}^{t-1} \eta_j u_j$; the triangle inequality gives $\|M_t - M_s\| \leq \sum_{j=s}^{t-1} \eta_j \|u_j\| = D_{t,s}$. □

Lemma E.2 (Origin feasibility and its consequences). Let $\xi_{t,s} \geq 0$ for all $s \in \mathcal{B}_t$, so that $0 \in \mathcal{C}_t$ (Def. C.2). Then:

(i) $\langle g_t, u_t^\star \rangle \geq \|u_t^\star\|^2 \geq 0$ and $\|u_t^\star\| \leq \|g_t\|$; consequently $\alpha_t = \|u_t^\star - g_t\|/\|g_t\| \leq 1$ and $(1-\alpha_t)\|g_t\| \leq \|u_t^\star\| \leq \|g_t\|$.

(ii) If u_t is produced from $u^{(0)} = g_t$ by any finite composition of *relaxed* half-space projections $u \mapsto (1-\rho)u + \rho\,\Pi_{H_{t,s}}(u)$ with $\rho \in [0,1]$ and $s \in \mathcal{B}_t$, then $\|u_t\| \leq \|g_t\|$. Under Assumption D.2 this yields $\|u_t\| \leq G$ and $D_{t,s} \leq G \sum_{j=s}^{t-1} \eta_j$.

Proof. (i) $\mathcal{C}_t$ is a nonempty closed convex set containing 0, so $u_t^\star = \Pi_{\mathcal{C}_t}(g_t)$ is well defined and satisfies the variational inequality $\langle g_t - u_t^\star,\, y - u_t^\star \rangle \leq 0$ for all $y \in \mathcal{C}_t$. Taking $y = 0$ gives $-\langle g_t, u_t^\star \rangle + \|u_t^\star\|^2 \leq 0$, i.e. $\langle g_t, u_t^\star \rangle \geq \|u_t^\star\|^2$. Cauchy–Schwarz then gives $\|u_t^\star\|^2 \leq \|g_t\|\|u_t^\star\|$, hence $\|u_t^\star\| \leq \|g_t\|$. Since $0 \in \mathcal{C}_t$, $\operatorname{dist}(g_t, \mathcal{C}_t) \leq \|g_t - 0\| = \|g_t\|$, so $\alpha_t \leq 1$; and $\|u_t^\star\|\|g_t\| \geq \langle g_t, u_t^\star \rangle \geq (1-\alpha_t)\|g_t\|^2$ by Lemma E.4(i) below, giving the lower bound.

(ii) Each $H_{t,s}$ is a closed convex set containing 0, so $\Pi_{H_{t,s}}(0) = 0$ and non-expansiveness of the projection gives $\|\Pi_{H_{t,s}}(u)\| = \|\Pi_{H_{t,s}}(u) - \Pi_{H_{t,s}}(0)\| \leq \|u\|$. By convexity of the norm, the relaxed map obeys $\|(1-\rho)u + \rho\Pi_{H_{t,s}}(u)\| \leq (1-\rho)\|u\| + \rho\|u\| = \|u\|$. The claim follows by induction over the passes. Finally, the update of Eq. (3) is exactly this relaxed projection with

$\rho = \|\widehat{g}_s\|^2/(\|\widehat{g}_s\|^2 + \delta) \in (0, 1]$, so the sequential solver of Section 3.3, the exact QP, and the safeguard of Lemma E.5 all fall under (ii). □

Lemma E.3 (Stale-gradient error). Under Assumption D.1 and Lemma E.1, for all $t > s$,

$$\|\nabla \ell_s(M_t) - \widehat{g}_s\| \le L\|M_t - M_s\| \le L\, D_{t,s}.$$

Proof. Immediate from L-smoothness of ℓ_s and Lemma E.1. □

Lemma E.4 (Inner-product transfer). Let u_t be any realized direction.

(i) (Current chunk.) $\langle g_t, u_t\rangle \ge (1 - \widehat{\alpha}_t)\|g_t\|^2$.

(ii) (Buffered chunk, $s \in \mathcal{B}_t$.) Under Assumptions D.1–D.2, Rule D.3 and Def. C.3,

$$-\langle u_t, \nabla \ell_s(M_t)\rangle \;\le\; (\lambda + LG)\, D_{t,s} + \varepsilon_t.$$

Proof. (i) $\langle g_t, u_t\rangle = \|g_t\|^2 + \langle g_t, u_t - g_t\rangle \ge \|g_t\|^2 - \|g_t\|\|u_t - g_t\| = (1-\widehat{\alpha}_t)\|g_t\|^2$ by Def. C.4. (ii) Decompose $\langle u_t, \nabla \ell_s(M_t)\rangle = \langle u_t, \widehat{g}_s\rangle + \langle u_t, \nabla \ell_s(M_t) - \widehat{g}_s\rangle$. The first term is $\ge -(\xi_{t,s} + \varepsilon_t)$ by 11; the second is $\ge -\|u_t\|\, LD_{t,s} \ge -LG\, D_{t,s}$ by Lemma E.3, Lemma E.2(ii) and Assumption D.2. Substituting $\xi_{t,s} = \lambda D_{t,s}$ from Rule D.3 completes the proof. □

Lemma E.5 (Exact-feasibility safeguard, $O(WP)$). Let u be the output of the sequential solver and let $\mathcal{V} := \{s \in \mathcal{B}_t : \langle u, \widehat{g}_s\rangle < -\xi_{t,s}\}$. Define

$$\kappa_t := \min\Big\{1,\ \min_{s\in\mathcal{V}} \frac{\xi_{t,s}}{|\langle u, \widehat{g}_s\rangle|}\Big\} \in [0, 1], \qquad u_t := \kappa_t\, u.$$

If $\xi_{t,s} > 0$ for every $s \in \mathcal{V}$, then $u_t \in \mathcal{C}_t$, i.e. $\varepsilon_t = 0$; moreover $\|u_t\| \le \|u\| \le \|g_t\|$.

Proof. For $s \in \mathcal{V}$ we have $\langle u, \widehat{g}_s\rangle = -|\langle u, \widehat{g}_s\rangle|$ and $\kappa_t \le \xi_{t,s}/|\langle u, \widehat{g}_s\rangle|$, hence $\langle u_t, \widehat{g}_s\rangle = -\kappa_t|\langle u, \widehat{g}_s\rangle| \ge -\xi_{t,s}$. For $s \notin \mathcal{V}$, if $\langle u, \widehat{g}_s\rangle \ge 0$ then $\langle u_t, \widehat{g}_s\rangle = \kappa_t\langle u, \widehat{g}_s\rangle \ge 0 \ge -\xi_{t,s}$; if $-\xi_{t,s} \le \langle u, \widehat{g}_s\rangle < 0$ then $\langle u_t, \widehat{g}_s\rangle = \kappa_t\langle u, \widehat{g}_s\rangle \ge \langle u, \widehat{g}_s\rangle \ge -\xi_{t,s}$ since $\kappa_t \le 1$. The norm claim follows from $\kappa_t \le 1$ and Lemma E.2(ii). Note $\xi_{t,s} = \lambda D_{t,s} > 0$ for all $s < t$ under Rule D.3 whenever some $\|u_j\| > 0$, so the hypothesis is satisfied in practice. □

E.2 Proof of Takeaway 1, part I: staleness-controlled Pareto safety

Proposition 3 (Staleness-controlled Pareto safety). *Let Assumptions D.1–D.2 hold, let the memory evolve as $M_{j+1} = M_j - \eta_j u_j$ with $\{M_j\} \subset \mathcal{K}$, and let the tolerance be set by Rule D.3, $\xi_{t,s} = \lambda D_{t,s}$ with $D_{t,s} = \sum_{j=s}^{t-1} \eta_j\|u_j\|$. Let u_t be* any *direction returned by the WRITE solver, with residual violation $\varepsilon_t \ge 0$ as in Definition C.3. Then for every buffered chunk $s \in \mathcal{B}_t$,*

$$\big\langle u_t, \nabla_M \ell_s(M_t)\big\rangle \;\ge\; -\underbrace{(\lambda + LG)\, D_{t,s}}_{\text{staleness price}} - \underbrace{\varepsilon_t}_{\text{solver infeasibility}}. \tag{13}$$

In particular, if the exact projection 10 or the safeguard of Lemma E.5 is used, then $\varepsilon_t = 0$ and $\langle u_t, \nabla_M \ell_s(M_t)\rangle \ge -(\lambda + LG)D_{t,s}$: the buffered constraint enforces true (non-stale) Pareto safety up to a single error term proportional to the accumulated memory drift, with a constant that is independent of the unknown quantity $\|u_t\|$.

Proof. Fix t and $s \in \mathcal{B}_t$ and decompose the true inner product into a buffered part and a staleness part:

$$\big\langle u_t, \nabla \ell_s(M_t)\big\rangle = \underbrace{\big\langle u_t, \widehat{g}_s\big\rangle}_{\text{(I)}} + \underbrace{\big\langle u_t, \nabla \ell_s(M_t) - \widehat{g}_s\big\rangle}_{\text{(II)}}, \qquad \widehat{g}_s = \nabla \ell_s(M_s). \tag{14}$$

Term (I). By the definition of the residual violation (Definition C.3), $\varepsilon_t \ge \big(-\xi_{t,s} - \langle u_t, \widehat{g}_s\rangle\big)_+ \ge -\xi_{t,s} - \langle u_t, \widehat{g}_s\rangle$ for each $s \in \mathcal{B}_t$, which is exactly the relaxed feasibility inequality 11:

$$\text{(I)} = \langle u_t, \widehat{g}_s\rangle \;\ge\; -\xi_{t,s} - \varepsilon_t \;=\; -\lambda D_{t,s} - \varepsilon_t,$$

the last equality by Rule D.3. Note that this step uses no assumption whatsoever: it holds by construction of ε_t.

Term (II). By Cauchy–Schwarz, (II) $\geq -\|u_t\|\,\|\nabla \ell_s(M_t) - \widehat{g}_s\|$. Since $M_t - M_s = -\sum_{j=s}^{t-1} \eta_j u_j$, the triangle inequality gives $\|M_t - M_s\| \leq \sum_{j=s}^{t-1} \eta_j \|u_j\| = D_{t,s}$ (Lemma E.1), and L-smoothness of ℓ_s on $\mathcal{K}$ (Assumption D.1) yields

$$\|\nabla \ell_s(M_t) - \widehat{g}_s\| = \|\nabla \ell_s(M_t) - \nabla \ell_s(M_s)\| \leq L\|M_t - M_s\| \leq L\, D_{t,s}$$

(Lemma E.3). Combining with $\|u_t\| \leq G$ from property (P),

$$\text{(II)} \;\geq\; -L\, G\, D_{t,s}.$$

Adding the two bounds in 14 gives 13. The final claim follows because $u_t = u_t^\star \in \mathcal{C}_t$ (resp. $u_t = \kappa_t u \in \mathcal{C}_t$ by Lemma E.5) implies $\langle u_t, \widehat{g}_s \rangle \geq -\xi_{t,s}$ for all $s \in \mathcal{B}_t$, hence $\varepsilon_t = 0$. □

E.3 Proof of Takeaway 1, part II: windowed forgetting bound

Theorem 4 (Windowed forgetting bound). *Let Assumptions D.1–D.2 and Rule D.3 hold, and let chunk s be protected by the buffer for the window $t = s+1, \ldots, s+\widetilde{W}$ (Definition C.2). Then ParetoMem satisfies*

$$\ell_s(M_{s+\widetilde{W}+1}) - \ell_s(M_{s+1}) \;\leq\; \underbrace{\frac{(\lambda + LG)G}{2}\Big(\sum_{t=s}^{s+\widetilde{W}} \eta_t\Big)^2}_{\text{(a) staleness}} + \underbrace{\sum_{t=s+1}^{s+\widetilde{W}} \eta_t \varepsilon_t}_{\text{(b) infeasibility}} + \underbrace{\frac{LG^2}{2}\sum_{t=s+1}^{s+\widetilde{W}} \eta_t^2}_{\text{(c) curvature}}. \tag{15}$$

If in addition the step sizes are non-increasing and the WRITE solver is exactly feasible ($\varepsilon_t \equiv 0$, e.g. under Lemma E.5), then

$$\ell_s(M_{s+\widetilde{W}+1}) - \ell_s(M_{s+1}) \;\leq\; \frac{(\lambda + LG)G}{2}(\widetilde{W}+1)^2\eta_s^2 + \frac{LG^2}{2}\widetilde{W}\eta_s^2 \;=\; O\big((\lambda + LG)\, G\, \widetilde{W}^2 \eta_s^2\big), \tag{16}$$

and with the order-free choice $\lambda = \Theta(LG)$ of Rule D.3(b) this is $O(LG^2\widetilde{W}^2\eta_s^2)$. If instead the solver is only approximately feasible with $\varepsilon_t \leq \bar{\varepsilon}$, an additional first-order term $\bar{\varepsilon}\widetilde{W}\eta_s$ appears, and the $O(\eta^2)$ rate is retained only when $O\big((\lambda + LG)\, G\, \widetilde{W}^2\eta_s^2\big) = O\big(LG^2K^2W^2\eta_s^2\big)$.

Proof. Step 1 (per-step descent inequality). Fix $t \in \{s+1, \ldots, s+\widetilde{W}\}$. Since ℓ_s is L-smooth on $\mathcal{K}$ (Assumption D.1) and $M_{t+1} = M_t - \eta_t u_t$ with $M_t, M_{t+1} \in \mathcal{K}$ (Definition C.1), the quadratic upper bound gives

$$\ell_s(M_{t+1}) \leq \ell_s(M_t) + \big\langle \nabla \ell_s(M_t), M_{t+1} - M_t \big\rangle + \frac{L}{2}\|M_{t+1} - M_t\|^2 = \ell_s(M_t) - \eta_t \big\langle \nabla \ell_s(M_t), u_t \big\rangle + \frac{L\eta_t^2}{2}\|u_t\|^2.$$

Step 2 (first-order term via Proposition 3). Because $t \in \{s+1, \ldots, s+\widetilde{W}\}$, we have $s \in \mathcal{B}_t$, so Proposition 3 applies and yields $-\langle \nabla \ell_s(M_t), u_t \rangle \leq (\lambda + LG) D_{t,s} + \varepsilon_t$. Using also $\|u_t\|^2 \leq G^2$ from property (P),

$$\ell_s(M_{t+1}) - \ell_s(M_t) \;\leq\; \eta_t\big[(\lambda + LG) D_{t,s} + \varepsilon_t\big] + \frac{L\eta_t^2 G^2}{2}. \tag{17}$$

Step 3 (telescoping). Summing 17 over $t = s+1, \ldots, s+\widetilde{W}$ and telescoping the left-hand side,

$$\ell_s(M_{s+\widetilde{W}+1}) - \ell_s(M_{s+1}) \leq (\lambda + LG) \sum_{t=s+1}^{s+\widetilde{W}} \eta_t D_{t,s} + \sum_{t=s+1}^{s+\widetilde{W}} \eta_t \varepsilon_t + \frac{LG^2}{2}\sum_{t=s+1}^{s+\widetilde{W}} \eta_t^2. \tag{18}$$

Step 4 (bounding the drift double sum). By property (P), $D_{t,s} = \sum_{j=s}^{t-1} \eta_j \|u_j\| \leq G \sum_{j=s}^{t-1} \eta_j$. Hence, writing $S := \sum_{t=s}^{s+\widetilde{W}} \eta_t$,

$$\sum_{t=s+1}^{s+\widetilde{W}} \eta_t D_{t,s} \;\leq\; G \sum_{t=s+1}^{s+\widetilde{W}} \sum_{j=s}^{t-1} \eta_t \eta_j \;=\; G \sum_{s \leq j < t \leq s+\widetilde{W}} \eta_t \eta_j \;=\; \frac{G}{2}\Big(S^2 - \sum_{t=s}^{s+\widetilde{W}} \eta_t^2\Big) \;\leq\; \frac{G}{2} S^2,$$

where the middle identity is the standard decomposition $\big(\sum_t \eta_t\big)^2 = \sum_t \eta_t^2 + 2\sum_{j<t} \eta_t \eta_j$. Substituting into 18 gives 15.

Step 5 (simplified form). If η_t is non-increasing then $\eta_t \le \eta_s$ for all $t \ge s$, so $S \le (\widetilde{W}+1)\eta_s$ and $\sum_{t=s+1}^{s+\widetilde{W}} \eta_t^2 \le \widetilde{W}\eta_s^2$; with $\varepsilon_t \equiv 0$ this yields 16. If only $\varepsilon_t \le \bar{\varepsilon}$ holds, term (b) of 15 is at most $\bar{\varepsilon}\sum_{t=s+1}^{s+\widetilde{W}} \eta_t \le \bar{\varepsilon}\widetilde{W}\eta_s$, which is first order in η_s and dominates the $O(\widetilde{W}^2\eta_s^2)$ terms unless $\bar{\varepsilon} = O(\widetilde{W}\eta_s)$. □

E.4 Proof of Takeaway 2: separation from unconstrained writing

Proposition 5 (Separation from unconstrained memory writing). *Consider the unconstrained (GradMem-style) writer $u_t = g_t$ and the ParetoMem writer with exact feasibility ($\varepsilon_t \equiv 0$) and a fixed tolerance parameter $\lambda > 0$, both run with constant step size $\eta_t \equiv \eta$ on the window $t = s+1, \dots, s+\widetilde{W}$, $\widetilde{W} = KW$.*

(i) (Worst-case upper bound for the unconstrained writer.) *Under Assumptions D.1–D.2,*

$$\ell_s(M_{s+\widetilde{W}+1}) - \ell_s(M_{s+1}) \;\le\; \widetilde{W}\eta G^2 + \frac{LG^2}{2}\widetilde{W}\eta^2.$$

(ii) (Matching lower bound: the first-order term is real.) *There exists an instance satisfying Assumptions D.1–D.2 with constants L, G on a bounded convex $\mathcal{K} \subset \mathbb{R}$, and a schedule with $\eta \le \eta_s$ and $L\eta \le \frac{1}{2}$, $L\widetilde{W}\eta \le \frac{1}{2}$, on which the unconstrained writer incurs*

$$\ell_s(M_{s+\widetilde{W}+1}) - \ell_s(M_{s+1}) \;\ge\; \frac{\widetilde{W}\eta G^2}{4} \;=\; \Omega(\widetilde{W}\eta G^2).$$

(iii) (Provable advantage regime.) *On the same instance, for any frozen $\lambda > 0$ (in particular the value fixed by Rule D.3(b)), Theorem 4 bounds ParetoMem's forgetting by $c_1(\lambda + LG)G\widetilde{W}^2\eta^2$ with c_1 absolute, so*

$$\frac{\textit{ParetoMem forgetting}}{\textit{unconstrained forgetting}} \;\le\; \frac{4c_1(\lambda+LG)\widetilde{W}\eta}{G} \;=\; O\Big(\frac{(\lambda+LG)\,KW\eta}{G}\Big).$$

Hence, for fixed (λ, L, G, K, W), ParetoMem is provably superior in the regime $\eta = o\big(G/((\lambda+LG)KW)\big)$—which reads $\eta = o\big(1/(LKW)\big)$ whenever $\lambda = O(LG)$—and the advantage is a full order in the step size ($O(\eta^2)$ versus $\Theta(\eta)$), not merely a better constant.

Proof. (i) By L-smoothness and $u_t = g_t$, $\ell_s(M_{t+1}) - \ell_s(M_t) \le -\eta\langle\nabla\ell_s(M_t), g_t\rangle + \frac{L\eta^2}{2}\|g_t\|^2$. Cauchy–Schwarz and Assumption D.2 give $-\langle\nabla\ell_s(M_t), g_t\rangle \le \|\nabla\ell_s(M_t)\|\|g_t\| \le G^2$ and $\|g_t\|^2 \le G^2$. Summing over the $\widetilde{W}$ steps of the window yields the claim. Note that no constraint is available to improve the first term: the bound G^2 is attained whenever $g_t = -\kappa\nabla\ell_s(M_t)$ with $\kappa > 0$ and both norms equal G.

(ii) Take $d = 1$, $\mathcal{K} = \big[-\frac{G}{2L}, \frac{G}{2L}\big]$, and two chunks

$$\ell_s(M) = \frac{L}{2}\Big(M + \frac{G}{2L}\Big)^2, \qquad \ell_t(M) = -G\,M \quad \text{for all } t = s+1, \dots, s+\widetilde{W},$$

i.e. the same "new" chunk is presented throughout the window. Then ℓ_s is exactly L-smooth and ℓ_t is 0-smooth (hence L-smooth); moreover $\nabla\ell_s(M) = LM + \frac{G}{2} \in [0, G]$ for $M \in \mathcal{K}$ and $|\nabla\ell_t(M)| = G$, so Assumptions D.1–D.2 hold with constants L and G. In particular all gradients have norm $\le G$, so the clipping of Remark C.2 is inactive and $\widehat{g}_s = \nabla\ell_s(M_s)$. The two gradients are in direct conflict: $\nabla\ell_t = -G < 0 \le \nabla\ell_s$ on $\mathcal{K}$.

Let $M_s = 0$ and let the write step for chunk s use $\eta_s = \eta$, so $u_s = \nabla\ell_s(0) = \frac{G}{2}$ and $M_{s+1} = -\frac{\eta G}{2}$. The unconstrained writer then uses $u_t = g_t = -G$ for every t in the window, giving $M_{s+1+k} =$

$M_{s+1}+k\eta G$ for $k=0,\dots,\widetilde{W}$; the condition $L\widetilde{W}\eta \le \frac{1}{2}$ ensures $M_{s+1+k}\in\mathcal{K}$ throughout. Writing $z := M_{s+1}+\frac{G}{2L}$,

$$\ell_s(M_{s+1+k})-\ell_s(M_{s+1}) = \frac{L}{2}\big[(z+k\eta G)^2-z^2\big] = \frac{Lk^2\eta^2G^2}{2}+k\eta G\cdot Lz = \frac{Lk^2\eta^2G^2}{2}+k\eta G\Big(\frac{G}{2}-\frac{L\eta G}{2}\Big).$$

Since $L\eta\le\frac{1}{2}$ we have $\frac{G}{2}-\frac{L\eta G}{2}\ge\frac{G}{4}$, and both terms are non-negative, so taking $k=\widetilde{W}$ gives $\ell_s(M_{s+\widetilde{W}+1})-\ell_s(M_{s+1})\ge\frac{\widetilde{W}\eta G^2}{4}$.

(iii) The instance of (ii) satisfies Assumptions D.1–D.2 with constants L, G, and Rule D.3(a) ($\xi_{t,s}=\lambda D_{t,s}$) holds by construction for whatever frozen $\lambda>0$ is used; Rule D.3(b) only fixes the numerical value of λ and is not needed here. Hence Theorem 4, equation 16, applies verbatim to the ParetoMem trajectory on the same instance and gives $\ell_s(M_{s+\widetilde{W}+1})-\ell_s(M_{s+1})\le\frac{(\lambda+LG)G}{2}(\widetilde{W}+1)^2\eta^2+\frac{LG^2}{2}\widetilde{W}\eta^2\le c_1(\lambda+LG)G\widetilde{W}^2\eta^2$ for an absolute constant c_1 (using $\widetilde{W}\ge1$ and $LG\le\lambda+LG$). Dividing by the lower bound $\frac{\widetilde{W}\eta G^2}{4}$ of part (ii) gives the stated ratio, which is $o(1)$ precisely when $\eta=o\big(G/((\lambda+LG)\widetilde{W})\big)=o\big(G/((\lambda+LG)KW)\big)$; if $\lambda=O(LG)$ this is $o(1/(LKW))$. □

Remark: The separation is not free. On the instance of Proposition 5(ii) the two gradients are exactly anti-parallel and $\widehat{g}_s=\frac{G}{2}$, so the tolerance set forces $u_t\ge-2\lambda D_{t,s}/G$ and hence $\|u_t\|\ll\|g_t\|$ early in the window: the measured conflict coefficient satisfies $\widehat{\alpha}_t\to1$ and ParetoMem nearly freezes the memory. This is the worst case of Theorem 6(ii) and is the precise sense in which the stability gain of Theorem 4 is paid for in plasticity. (By Corollary 7(i) the constraint becomes vacuous once $\lambda D_{t,s}\ge G^2/2$, after which ParetoMem writes freely again.)

Remark: Scope of the separation under Rule D.3(b). Proposition 5(iii) treats λ as a constant independent of η. This is exactly the situation of Rule D.3(b): λ is calibrated once and frozen, and the step-size ablation of Appendix F varies η at fixed λ. If λ were instead *re-calibrated* after every change of η, then $\bar\rho\propto\eta$ and hence $\lambda\propto1/\eta$; the λ-part of the forgetting bound, $\frac{\lambda G}{2}(\widetilde{W}+1)^2\eta^2\approx\frac{\tau G\|\widehat{g}\|_{\mathrm{ref}}}{2\bar{u}H_0}(\widetilde{W}+1)^2\eta$ with $\bar u$ the mean $\|u_t\|$, would then be *first* order in η, and the ratio in (iii) would tend to the constant $2\tau(\|\widehat{g}\|_{\mathrm{ref}}/\bar u)(W/W_0)+O(LKW\eta)$ rather than to zero. In that reading the separation is a constant-factor improvement governed by τ (vanishing as $\tau\to0$), not an order in η. The $O(\eta^2)$ statement of (iii) is for frozen λ.

E.5 Proof of Takeaway 3: unconditional current-chunk progress

Theorem 6 (Unconditional current-chunk progress). *Let $g_t=\nabla\ell_t(M_t)\neq0$ be the current write gradient, let u_t be the direction returned by the WRITE solver, and let $\widehat{\alpha}_t=\|u_t-g_t\|/\|g_t\|$ be the* measured *conflict coefficient of Definition C.4. Then:*

(i) (Alignment; no assumptions.) $\langle g_t,u_t\rangle\ge(1-\widehat{\alpha}_t)\|g_t\|^2$.

(ii) *Let $g_t=\nabla\ell_t(M_t)\neq0$, let $\xi_{t,s}\ge0$ for every $s\in\mathcal{B}_t$ (guaranteed by construction under Rule D.3(a), since $D_{t,s}\ge0$ and $\bar D_t\ge0$; $\mathcal{B}_t$ may be the buffer before or after the vacuity skip of Algorithm 1, see Corollary 7(i)), and let*

$$\mathcal{C}_t=\bigcap_{s\in\mathcal{B}_t}\mathcal{H}_{t,s},\qquad \mathcal{H}_{t,s}=\big\{u\in\mathbb{R}^P:\langle u,\hat g_s\rangle\ge-\xi_{t,s}\big\},\qquad u_t^\star=\Pi_{\mathcal{C}_t}(g_t),$$

with $\alpha_t=\mathrm{dist}(g_t,\mathcal{C}_t)/\|g_t\|$ as in Definition C.4. Then $u_t^\star$ exists and is unique, $\widehat{\alpha}_t=\alpha_t\in[0,1]$, and

$$\langle g_t,u_t^\star\rangle\ \ge\ \|u_t^\star\|^2\ \ge\ 0, \tag{19}$$

$$(1-\alpha_t)\|g_t\|\ \le\ \|u_t^\star\|\ \le\ \sqrt{1-\alpha_t^2}\,\|g_t\|\ \le\ \|g_t\|. \tag{20}$$

Consequently $u_t^\star$ is never an ascent direction for the current chunk, and

$$\alpha_t=1\iff u_t^\star=0,$$

exactly (not merely asymptotically): the sole way ParetoMem can lose current-chunk progress is by freezing the memory. The right-hand inequality in Eq.20 is an equality if and only if $\langle g_t - u_t^\star, u_t^\star\rangle = 0$. When exactly one constraint s is active at $u_t^\star$ with multiplier $\mu_s \geq 0$ one has the exact identity $\|u_t^\star\|^2 = (1-\alpha_t^2)\|g_t\|^2 - 2\mu_s\xi_{t,s}$, so equality holds iff the active slack vanishes ($\xi_{t,s} = 0$) and is strict for $\xi_{t,s} > 0$; in particular for the EMA-only buffer at $\lambda = 0$ (the A-GEM corner of Proposition 9) $\|u_t^\star\| = \sqrt{1-\alpha_t^2}\,\|g_t\|$ holds identically. No hypothesis on ℓ_t, on the solver, on L, on G, or on the conflict level is used.

(iii) (Descent.) *Under Assumptions D.1–D.2,*

$$\ell_t(M_{t+1}) - \ell_t(M_t) \;\leq\; -\eta_t(1-\widehat{\alpha}_t)\|g_t\|^2 + \frac{L\eta_t^2 G^2}{2},$$

and if moreover $u_t = u_t^\star$, the sharper, assumption-free-in-α bound

$$\ell_t(M_{t+1}) - \ell_t(M_t) \;\leq\; -\eta_t\Big(1 - \frac{L\eta_t}{2}\Big)\|u_t^\star\|^2$$

holds; hence ℓ_t strictly decreases whenever $\eta_t < 2/L$ and $u_t^\star \neq 0$.

Proof. (i) Write $u_t = g_t + (u_t - g_t)$. Then

$$\langle g_t, u_t\rangle = \|g_t\|^2 + \langle g_t, u_t - g_t\rangle \geq \|g_t\|^2 - \|g_t\|\,\|u_t - g_t\| = \|g_t\|^2 - \widehat{\alpha}_t\|g_t\|^2 = (1-\widehat{\alpha}_t)\|g_t\|^2,$$

using Cauchy–Schwarz and the definition of $\widehat{\alpha}_t$. No property of $\mathcal{C}_t$, of ℓ_t, or of the solver is used.

(ii) Every $\mathcal{H}_{t,s}$ is a closed half-space and, because $\xi_{t,s} \geq 0$ gives $\langle 0, \hat{g}_s\rangle = 0 \geq -\xi_{t,s}$, each contains the origin; hence $\mathcal{C}_t$ is a finite intersection of closed convex sets with $0 \in \mathcal{C}_t$, so it is nonempty, closed and convex. (If Algorithm 1 has removed vacuous entries, the retained half-spaces still all contain the origin, and by Corollary 7(i) the projection is the same as for the full buffer.) The map $u \mapsto \frac{1}{2}\|u - g_t\|^2$ is 1-strongly convex and coercive, so it attains its minimum over $\mathcal{C}_t$ at a unique point $u_t^\star$; this also makes $\alpha_t\|g_t\| = \operatorname{dist}(g_t, \mathcal{C}_t) = \|u_t^\star - g_t\|$ and, by Definition C.4, $\widehat{\alpha}_t = \alpha_t$ when the exact projection is used. For the variational characterisation, fix $y \in \mathcal{C}_t$ and $\theta \in (0,1]$; convexity gives $u_t^\star + \theta(y - u_t^\star) \in \mathcal{C}_t$, so minimality yields $\|u_t^\star + \theta(y-u_t^\star) - g_t\|^2 \geq \|u_t^\star - g_t\|^2$, i.e. $2\theta\langle u_t^\star - g_t,\, y - u_t^\star\rangle + \theta^2\|y - u_t^\star\|^2 \geq 0$; dividing by θ and letting $\theta \downarrow 0$ gives

$$\langle u_t^\star - g_t,\; y - u_t^\star\rangle \;\geq\; 0 \qquad \forall\, y \in \mathcal{C}_t. \tag{21}$$

Write $r_t := g_t - u_t^\star$, so that $\|r_t\| = \alpha_t\|g_t\|$. Choosing the admissible point $y = 0 \in \mathcal{C}_t$ in Eq.21 gives $\langle u_t^\star - g_t, -u_t^\star\rangle \geq 0$, that is

$$\langle r_t,\, u_t^\star\rangle \;\geq\; 0 \qquad\Longleftrightarrow\qquad \langle g_t, u_t^\star\rangle \;\geq\; \|u_t^\star\|^2 \;\geq\; 0, \tag{22}$$

which is Eq.19; this single use of the feasibility of the origin is the only structural input of the whole argument. Expanding $g_t = u_t^\star + r_t$ and inserting Eq.22,

$$\|g_t\|^2 = \|u_t^\star\|^2 + 2\langle u_t^\star, r_t\rangle + \|r_t\|^2 \;\geq\; \|u_t^\star\|^2 + \alpha_t^2\|g_t\|^2, \tag{23}$$

whence $\|u_t^\star\|^2 \leq (1-\alpha_t^2)\|g_t\|^2$, which is the upper half of Eq.20 and, since the left-hand side is nonnegative and $g_t \neq 0$, also forces $\alpha_t \leq 1$; together with $\alpha_t \geq 0$ this gives $\alpha_t \in [0,1]$. The lower half of Eq.20 is the reverse triangle inequality, $\|u_t^\star\| \geq \|g_t\| - \|r_t\| = (1-\alpha_t)\|g_t\|$. The stated equivalence now follows from Eq.20 alone: if $u_t^\star = 0$ then $\alpha_t\|g_t\| = \|r_t\| = \|g_t\|$, so $\alpha_t = 1$; conversely $\alpha_t = 1$ makes the upper bound vanish, so $u_t^\star = 0$.

For the equality characterisation, Eq.23 shows that the upper bound in Eq.20 is attained iff $\langle u_t^\star, r_t\rangle = 0$. If exactly one constraint s is active at $u_t^\star$, the KKT conditions of the projection give $r_t = g_t - u_t^\star = -\mu_s\hat{g}_s$ with $\mu_s \geq 0$ and $\langle u_t^\star, \hat{g}_s\rangle = -\xi_{t,s}$, hence $\langle u_t^\star, r_t\rangle = -\mu_s\langle u_t^\star, \hat{g}_s\rangle = \mu_s\xi_{t,s}$ and Eq.23 reads $\|u_t^\star\|^2 = (1-\alpha_t^2)\|g_t\|^2 - 2\mu_s\xi_{t,s}$. Since $\mu_s > 0$ whenever the constraint is active and $g_t \notin \mathcal{H}_{t,s}$, equality holds iff $\xi_{t,s} = 0$. Under Rule D.3(a) this is the case $\lambda = 0$ (or $D_{t,s} = 0$), which for the EMA-only buffer is exactly the A-GEM corner of Proposition 9(ii).

(iii) By L-smoothness of ℓ_t (Assumption D.1) and $M_{t+1} - M_t = -\eta_t u_t$,

$$\ell_t(M_{t+1}) \leq \ell_t(M_t) - \eta_t\langle g_t, u_t\rangle + \frac{L\eta_t^2}{2}\|u_t\|^2.$$

Bounding $\langle g_t, u_t\rangle$ by part (i) and $\|u_t\|^2$ by G^2 (property (P) and Assumption D.2) gives the first display. For the second, take $u_t = u_t^\star$ and use $\langle g_t, u_t^\star\rangle \geq \|u_t^\star\|^2$ from part (ii) together with $\|u_t\|^2 = \|u_t^\star\|^2$:

$$\ell_t(M_{t+1}) - \ell_t(M_t) \leq -\eta_t\|u_t^\star\|^2 + \frac{L\eta_t^2}{2}\|u_t^\star\|^2 = -\eta_t\Big(1 - \frac{L\eta_t}{2}\Big)\|u_t^\star\|^2. \qquad \square$$

E.6 Proof of Takeaway 4, part I: protection horizon

Corollary 7 (Protection horizon of a stale constraint). *Let Assumption D.2 and Rule D.3(a) hold with a fixed $\lambda > 0$, let anchors be clipped as in Remark C.2 (so $\|\widehat{g}_s\| \le G$), and recall property (P), $\|u_t\| \le \|g_t\| \le G$. Fix a buffered chunk s with $\widehat{g}_s \neq 0$ and define its* protection horizon *(in write steps)*

$$H_s \;:=\; \min\Big\{k \ge 1 :\; \lambda\, D_{s+k,s} \;\ge\; G\,\|\widehat{g}_s\|\Big\} \qquad (\min\emptyset := \infty).$$

Then:

(i) (Vacuity; exactness of the online skip test.) *For every t with*

$$\lambda D_{t,s} \;\ge\; G\|\widehat{g}_s\| \qquad (\textit{equivalently } t-s \ge H_s), \tag{24}$$

the half-space $\mathcal{H}_{t,s}$ contains the whole ball $B_G := \{u : \|u\| \le G\}$, so the constraint indexed by s is satisfied by every *admissible direction. Consequently (a) the exact projection $u_t^\star = \Pi_{\mathcal{C}_t}(g_t)$ is unchanged if entry s is deleted from $\mathcal{B}_t$; (b) the output of the cyclic solver of Eq. (8) is unchanged as well, for every ordering σ and every $\delta \ge 0$; (c) the test 24 involves only stored quantities (λ, $D_{t,s}$, $\|\widehat{g}_s\|$, G), so the skip rule of Algorithm 1—drop entry s as soon as 24 holds—never alters the write direction. Entry s provides no protection whatsoever after age H_s.*

(ii) (Finiteness and scaling.) *If the realized updates are non-degenerate in the sense that $\eta_j\|u_j\| \ge \rho_{\min} > 0$ for all $j \ge s$, then*

$$H_s \;\le\; \Big\lceil \frac{G\|\widehat{g}_s\|}{\lambda\rho_{\min}} \Big\rceil \;\le\; \Big\lceil \frac{G^2}{\lambda\rho_{\min}} \Big\rceil =: H^{\max}.$$

Under Rule D.3(b),

$$H^{\max} = \Big\lceil \frac{G}{\|\widehat{g}\|_{\mathrm{ref}}} \cdot \frac{\bar\rho}{\rho_{\min}} \cdot \frac{H_0}{\tau} \Big\rceil, \qquad H^{\mathrm{nom}} := \Big\lceil \frac{H_0}{\tau} \Big\rceil = \Big\lceil \frac{KW_0}{\tau} \Big\rceil,$$

where H^{nom} is the exact horizon of a nominal *anchor ($\|\widehat{g}_s\| = \|\widehat{g}\|_{\mathrm{ref}}$, $\eta_j\|u_j\| \equiv \bar\rho$). A FIFO buffer of W chunks retains entries up to age KW steps; every entry older than $H^{\max}$ never binds, so $W > H^{\max}/K$ is provably redundant (the excess entries cost $O(P)$ time and memory per step each and provide no protection), while $W \le W_0/\tau$ is the range in which a nominal entry remains binding for its entire residence.*

(iii) (Tolerance–horizon trade-off.) *For a fixed trajectory H_s is non-increasing in λ. Under Rule D.3(b), λ is proportional to τ at fixed $(G, \|\widehat{g}\|_{\mathrm{ref}}, \bar\rho, H_0)$, so the single dimensionless dial τ controls both the per-step staleness price $(\lambda + LG)D_{t,s}$ of Proposition 3 (increasing in τ) and the protection length $H^{\mathrm{nom}} = H_0/\tau$ (decreasing in τ): $\tau = 1$ makes a nominal anchor vacuous exactly when it leaves a buffer of W_0 chunks, and $\tau < 1$ keeps it binding throughout its residence with margin $1/\tau$.*

Proof. (i) Since $\eta_j\|u_j\| \ge 0$, the map $k \mapsto D_{s+k,s}$ is non-decreasing, so $t - s \ge H_s$ implies $\lambda D_{t,s} \ge \lambda D_{s+H_s,s} \ge G\|\widehat{g}_s\|$, and conversely 24 at time t implies $H_s \le t - s$ by definition; the two formulations are equivalent. Assume 24. For any $u \in B_G$, Cauchy–Schwarz gives

$$\langle u, \widehat{g}_s\rangle \;\ge\; -\|u\|\,\|\widehat{g}_s\| \;\ge\; -G\|\widehat{g}_s\| \;\ge\; -\lambda D_{t,s} \;=\; -\xi_{t,s},$$

so $B_G \subseteq \mathcal{H}_{t,s}$.

(a) Let $\mathcal{C}_t' := \bigcap_{s' \in \mathcal{B}_t \setminus \{s\}} \mathcal{H}_{t,s'} \supseteq \mathcal{C}_t$ and $u_t' := \Pi_{\mathcal{C}_t'}(g_t)$. Since every retained half-space contains the origin, Theorem 6(ii) applies to $\mathcal{C}_t'$ and gives $\|u_t'\| \le \|g_t\| \le G$, i.e. $u_t' \in B_G \subseteq \mathcal{H}_{t,s}$; hence $u_t' \in \mathcal{C}_t' \cap \mathcal{H}_{t,s} = \mathcal{C}_t$. As u_t' minimises $\|u - g_t\|$ over the larger set $\mathcal{C}_t'$ and belongs to the smaller set $\mathcal{C}_t$, it also minimises over $\mathcal{C}_t$; by uniqueness $u_t' = u_t^\star$.

(b) By Lemma E.2(ii) each elementary step of the cyclic solver does not increase the norm, so every intermediate iterate u satisfies $\|u\| \le \|g_t\| \le G$. At each visit to entry s the test $\langle u, \widehat{g}_s\rangle < -\xi_{t,s}$ therefore fails (by the display above), and the step at s is the identity. Deleting s from the sweep thus produces the identical iterate sequence, for every σ and $\delta \ge 0$.

(c) is immediate from (a)–(b) and the fact that Algorithm 1 stores $D_{t,s}$ and $\|\widehat{g}_s\|$ with each entry.

(ii) By assumption $D_{s+k,s} = \sum_{j=s}^{s+k-1} \eta_j \|u_j\| \geq k\rho_{\min}$, so the defining inequality of H_s holds as soon as $k \geq G\|\widehat{g}_s\|/(\lambda\rho_{\min})$, giving the first bound; the second follows from $\|\widehat{g}_s\| \leq G$, which holds by construction for clipped anchors (and under Assumption D.2 also for the raw gradient). Substituting $\lambda = \tau G\|\widehat{g}\|_{\mathrm{ref}}/(\bar{\rho}H_0)$ gives the expression for $H^{\max}$. For a nominal anchor, $\lambda D_{s+k,s} = \lambda k\bar{\rho} = \tau G\|\widehat{g}\|_{\mathrm{ref}}\, k/H_0$, which first reaches $G\|\widehat{g}\|_{\mathrm{ref}}$ at $k = \lceil H_0/\tau \rceil = H^{\mathrm{nom}}$. The redundancy claim is (i) applied to all entries of age $> H^{\max}$; the residence claim is $KW \leq H_0/\tau \iff W \leq W_0/\tau$.

(iii) For a fixed trajectory the sequence $D_{s+k,s}$ does not depend on λ, while the threshold $G\|\widehat{g}_s\|/\lambda$ is decreasing in λ, so the minimal k attaining it is non-increasing in λ. The remaining statements follow by substituting $\lambda \propto \tau$ from Rule D.3(b) and $H^{\mathrm{nom}} = H_0/\tau$ from (ii). □

Remark: Why an EMA entry is needed, not merely useful. Corollary 7 shows that *any* per-chunk stale constraint decays to vacuity after at most $H^{\max} = O\big(G^2/(\lambda\rho_{\min})\big)$ write steps—under Rule D.3(b), $O\big((G/\|\widehat{g}\|_{\mathrm{ref}})(\bar{\rho}/\rho_{\min})\, H_0/\tau\big)$—independently of how large W is: with a pure FIFO buffer, long-range protection is impossible in principle, not just expensive. The EMA entry circumvents this because its drift statistic is re-anchored at every update rather than accumulated from a fixed write time, so its tolerance stays uniformly bounded (Proposition 8(a)); Proposition 8 characterises exactly what it does and does not guarantee.

E.7 Proof of Takeaway 4, part II: aggregate safety of the EMA entry

Proposition 8 (Aggregate safety of the EMA constraint). *Let $\mathcal{A}_t$ be the (possibly unbounded) set of chunks summarised by the EMA entry at time t, and let the EMA gradient be the* unnormalised *exponential average of the clipped anchors,*

$$\bar{g}_t \;=\; \sum_{s\in\mathcal{A}_t} w_{t,s}\,\widehat{g}_s, \qquad w_{t,s} \geq 0, \quad \sum_{s\in\mathcal{A}_t} w_{t,s} = 1,$$

as produced by $\bar{g} \leftarrow \beta\bar{g} + (1-\beta)\widehat{g}_s$ initialised at the first anchor and never renormalised (i.e. $c_t \equiv 1$ in the notation of Rule D.3). Then $\|\bar{g}_t\| \leq G$. Let $\bar{D}_t := \sum_{s\in\mathcal{A}_t} w_{t,s} D_{t,s}$ be the weighted average drift and suppose the realized direction satisfies the EMA constraint $\langle u_t, \bar{g}_t\rangle \geq -\bar{\xi}_t$ with $\bar{\xi}_t \geq 0$. Then, under Assumptions D.1–D.2 and property (P),

$$\sum_{s\in\mathcal{A}_t} w_{t,s}\big\langle u_t, \nabla\ell_s(M_t)\big\rangle \;\geq\; -\bar{\xi}_t - L\,G\,\bar{D}_t. \tag{25}$$

In particular, with the EMA tolerance of Rule D.3(a), $\bar{\xi}_t = \lambda\bar{D}_t$, the bound becomes $-(\lambda + LG)\bar{D}_t$, i.e. exactly the statement of Proposition 3 with $D_{t,s}$ replaced by $\bar{D}_t$*: the EMA entry behaves as a single virtual chunk whose loss is the weighted average $\bar{\ell}_t := \sum_s w_{t,s}\ell_s$ and whose staleness is the average staleness. Moreover:*

(a) (Re-anchoring and bounded tolerance.) *If at each EMA update the existing weights are decayed, $w \leftarrow \beta w$, and the new anchor enters with weight $1-\beta$ and zero drift, then the average drift obeys the $O(1)$ recursion*

$$\bar{D} \;\leftarrow\; \beta\big(\bar{D} + \Delta\big), \tag{26}$$

where Δ is the path length accumulated since the previous EMA update. Hence $\bar{D}_t \leq \beta\Delta_{\max}/(1-\beta)$ uniformly in t, with $\Delta_{\max}$ the largest inter-update path length ($\Delta_{\max} \leq GK\eta_{\max}$ if the EMA is updated once per chunk). Unlike a FIFO entry (Corollary 7), the EMA constraint therefore has no *finite protection horizon: it is non-vacuous at every t for which $\lambda\beta\Delta_{\max}/(1-\beta) < G\|\bar{g}_t\|$.*

(b) (Set-invariance.) *If an implementation rescales $\bar{g}_t \mapsto c_t\bar{g}_t$ with $c_t > 0$, the feasible half-space is unchanged if and only if the tolerance is rescaled to $c_t\bar{\xi}_t$. ParetoMem does not rescale, which is why Rule D.3(a) is stated with $\bar{\xi}_t = \lambda\bar{D}_t$ and no normalisation factor.*

(c) (Window statement.) *If the weights are held fixed over a window ($w_{t,s} \equiv w_s$ for $t = s'+1, \ldots, s'+\widetilde{W}$), then $\bar{\ell} := \sum_s w_s\ell_s$ is L-smooth with $\|\nabla\bar{\ell}\| \leq G$, and the conclusion of Theorem 4 holds verbatim for $\bar{\ell}$.*

Proof. $\|\bar{g}_t\| \le \sum_s w_{t,s}\|\widehat{g}_s\| \le G$ because the anchors are clipped (Remark C.2) and the weights form a convex combination.

For 25, decompose each term as in 14 and average with the weights $w_{t,s}$:

$$\sum_s w_{t,s}\langle u_t, \nabla\ell_s(M_t)\rangle = \underbrace{\sum_s w_{t,s}\langle u_t, \widehat{g}_s\rangle}_{\text{(I)}} + \underbrace{\sum_s w_{t,s}\langle u_t, \nabla\ell_s(M_t) - \widehat{g}_s\rangle}_{\text{(II)}}.$$

For (I), linearity of the inner product gives

$$\text{(I)} = \Big\langle u_t,\ \sum_s w_{t,s}\widehat{g}_s \Big\rangle = \langle u_t, \bar{g}_t\rangle \ \ge\ -\bar{\xi}_t.$$

For (II), under Assumption D.2 the clipping is inactive and $\widehat{g}_s = \nabla\ell_s(M_s)$, so Lemma E.3 gives $\|\nabla\ell_s(M_t) - \widehat{g}_s\| \le LD_{t,s}$; Cauchy–Schwarz termwise and property (P) ($\|u_t\| \le G$) yield

$$\text{(II)} \ \ge\ -\sum_s w_{t,s}\|u_t\|\, LD_{t,s} \ \ge\ -LG\sum_s w_{t,s}D_{t,s} = -LG\bar{D}_t.$$

Adding the two bounds gives 25; substituting $\bar{\xi}_t = \lambda\bar{D}_t$ yields $-(\lambda + LG)\bar{D}_t$.

(a) Let the update occur at time t^+ with previous update at t^-, and let $\Delta = D_{t^+,t^-}$ be the path length in between. For every $s \in \mathcal{A}_{t^-}$, $D_{t^+,s} = D_{t^-,s} + \Delta$ and the new weight is $\beta w_{t^-,s}$; the new anchor has weight $1-\beta$ and drift 0. Hence

$$\bar{D}_{t^+} = \sum_{s\in\mathcal{A}_{t^-}} \beta w_{t^-,s}\big(D_{t^-,s} + \Delta\big) + (1-\beta)\cdot 0 = \beta\big(\bar{D}_{t^-} + \Delta\big),$$

which is 26. Unrolling from $\bar{D} = 0$ at the first anchor, $\bar{D} = \sum_{i=1}^{n}\beta^{\,n-i+1}\Delta_i \le \Delta_{\max}\beta/(1-\beta)$. Between updates $\bar{D}_t$ grows by at most $\Delta_{\max}$ as well, so the same bound holds up to the factor $(1+\beta)/\beta$, which we absorb into $\Delta_{\max}$. The non-vacuity claim is the negation of the criterion of Corollary 7(i) applied to the EMA entry: the half-space $\{\langle u, \bar{g}_t\rangle \ge -\lambda\bar{D}_t\}$ contains B_G iff $\lambda\bar{D}_t \ge G\|\bar{g}_t\|$.

(b) $\{u : \langle u, c_t\bar{g}_t\rangle \ge -\bar{\xi}'_t\} = \{u : \langle u, \bar{g}_t\rangle \ge -\bar{\xi}'_t/c_t\}$, which equals the original half-space iff $\bar{\xi}'_t = c_t\bar{\xi}_t$ (when $\bar{g}_t \ne 0$).

(c) $\bar{\ell} = \sum_s w_s\ell_s$ is a convex combination of L-smooth functions with gradients bounded by G, hence is itself L-smooth with $\|\nabla\bar{\ell}\| \le G$; moreover $\langle u_t, \nabla\bar{\ell}(M_t)\rangle = \sum_s w_s\langle u_t, \nabla\ell_s(M_t)\rangle \ge -(\lambda + LG)\bar{D}_t$, which is precisely the input required by Step 2 of the proof of Theorem 4; Steps 1, 3–5 of that proof then go through unchanged with ℓ_s replaced by $\bar{\ell}$ and $D_{t,s}$ by $\bar{D}_t \le G\sum_j \eta_j$. □

Remark: What the EMA constraint does *not* guarantee. Inequality 25 controls only a *weighted average* over $\mathcal{A}_t$; it does not imply a per-chunk guarantee, and this is unavoidable. Take $\mathcal{A}_t = \{1, 2\}$, $w_{t,1} = w_{t,2} = \frac{1}{2}$ and $\widehat{g}_1 = -\widehat{g}_2 =: h \ne 0$. Then $\bar{g}_t = 0$, the EMA constraint $\langle u_t, \bar{g}_t\rangle \ge -\bar{\xi}_t$ is vacuous, and $u_t = -Gh/\|h\|$ is admissible even though it conflicts maximally with chunk 1: $\langle u_t, \widehat{g}_1\rangle = -G\|h\|$. Per-chunk protection therefore requires explicit FIFO entries, while the EMA entry supplies aggregate protection *beyond* the finite protection horizon of Corollary 7. This is the formal justification for the hybrid buffer (FIFO + EMA) of Section 3.3, and we state it as a limitation rather than a guarantee.

E.8 Connection between ParetoMem and other Learners

The only structural difference between ParetoMem and prior gradient-constraint methods is the slack $\xi_{t,s} = \lambda D_{t,s}$. This subsection isolates it: we show that the natural stale-evidence adaptations of GEM, A-GEM and PCGrad are *not* separate baselines but corners of our own method at $\lambda = 0$, and then sweep λ (equivalently τ under Rule D.3(b)) to measure what the tolerance buys.

Proposition 9 (Reductions at $\lambda = 0$). *Let $\mathcal{C}_t(\lambda) = \bigcap_{s\in\mathcal{B}_t}\{u \in \mathbb{R}^P : \langle u, \hat{g}_s\rangle \ge -\lambda D_{t,s}\}$ and $u_t(\lambda) = \Pi_{\mathcal{C}_t(\lambda)}(g_t)$, and define the stale adaptations of the three constraint-based baselines by*

replacing every fresh reference gradient by its write-time (clipped) value $\hat{g}_s = \mathrm{clip}_G(\nabla \ell_s(M_s))$*:*

$$\text{GEM}^\dagger: \quad u_t^{\mathrm{G}} = \arg\min_u \tfrac{1}{2}\|u - g_t\|^2 \ \ \textit{s.t.}\ \ \langle u, \hat{g}_s\rangle \geq 0 \ \ \forall s \in \mathcal{B}_t,$$

$$\text{A-GEM}^\dagger: \quad u_t^{\mathrm{A}} = g_t - \frac{\big(\langle g_t, \bar{g}_t\rangle\big)_-}{\|\bar{g}_t\|^2}\,\bar{g}_t \quad (\textit{no change if } \langle g_t, \bar{g}_t\rangle \geq 0),$$

$$\text{PCGRAD}^\dagger_\sigma: \quad u \leftarrow g_t, \ \ \textit{then for } s = \sigma(1), \ldots, \sigma(|\mathcal{B}_t|): \ u \leftarrow u - \frac{\big(\langle u, \hat{g}_s\rangle\big)_-}{\|\hat{g}_s\|^2}\,\hat{g}_s,$$

where $(x)_- = \min\{x, 0\}$ *and* σ *is an ordering of the buffer. Then:*

1. (GEM.) *With the per-chunk buffer and* $\lambda = 0$*,* $u_t(0) = u_t^{\mathrm{G}}$*, and this point is unique even when* $\{\hat{g}_s\}$ *is linearly dependent and GEM's dual multipliers are not.*
2. (A-GEM.) *With the single EMA constraint and* $\lambda = 0$*,* $u_t(0) = u_t^{\mathrm{A}}$*. The identity is invariant under any positive rescaling* $\bar{g}_t \mapsto c_t \bar{g}_t$ *(so it holds for both the unnormalised EMA used by ParetoMem and a normalised one), and* $u_t(0) = g_t$ *in the degenerate case* $\bar{g}_t = 0$*.*
3. (PCGrad, single reference.) *If* $|\mathcal{B}_t| = 1$ *then* $u_t(0) = \text{PCGRAD}^\dagger$*, and both equal the projection of* g_t *onto the normal plane of* $\hat{g}_s$ *whenever the constraint is violated at* g_t*.*
4. (PCGrad, several references — no reduction to the projection.) *For* $|\mathcal{B}_t| \geq 2$*,* $\text{PCGRAD}^\dagger_\sigma$ *coincides with* one cyclic pass of Eq. (8) with $\xi_{t,s} \equiv 0$ and $\delta = 0$*, i.e. with ParetoMem's approximate solver at* $\lambda = 0$*; it is in general* σ*-dependent and satisfies* $\varepsilon_t > 0$*, hence differs from* $u_t(0)$*. Explicit instances are given in the proof.*

Proof. Throughout, uniqueness of Euclidean projections onto the nonempty closed convex sets $\mathcal{C}_t(\lambda)$ is the one established in the proof of Theorem 6(ii). At $\lambda = 0$ the vacuity test 24 of Corollary 7(i) reads $0 \geq G\|\hat{g}_s\|$ and is never met for $\hat{g}_s \neq 0$, so Algorithm 1 skips no entry and $\mathcal{B}_t$ is the full buffer; the reductions below are therefore unaffected by the skip rule. Clipping only changes the numerical value of $\hat{g}_s$ and is applied identically to ParetoMem and to the †-baselines, so it plays no role in the identities.

(i). At $\lambda = 0$ the tolerance vanishes, so $\mathcal{C}_t(0) = \{u : \langle u, \hat{g}_s\rangle \geq 0, \ s \in \mathcal{B}_t\}$ is literally the feasible set of $\text{GEM}^\dagger$, and the two objectives $\frac{1}{2}\|u - g_t\|^2$ coincide; strong convexity gives a unique minimiser, so $u_t(0) = u_t^{\mathrm{G}}$. For the second claim, stack the anchors as the rows of $\widehat{G} \in \mathbb{R}^{|\mathcal{B}_t| \times P}$. The KKT conditions of $\text{GEM}^\dagger$ read $u - g_t = \widehat{G}^\top v$, $v \geq 0$, $\widehat{G}u \geq 0$, $v^\top \widehat{G}u = 0$, and eliminating u gives GEM's dual $\min_{v \geq 0} \frac{1}{2} v^\top \widehat{G}\widehat{G}^\top v + g_t^\top \widehat{G}^\top v$. If $\widehat{G}\widehat{G}^\top$ is singular the dual minimiser $v^\star$ need not be unique, but every dual minimiser reconstructs the *same* primal point $g_t + \widehat{G}^\top v^\star$, because the primal is strongly convex; hence GEM's dual solver returns $u_t(0)$ regardless of which $v^\star$ it finds.

(ii). Let $a \neq 0$ and $\xi \geq 0$ and consider the single half-space $H = \{u : \langle u, a\rangle \geq -\xi\}$. If $\langle g_t, a\rangle \geq -\xi$ then $g_t \in H$ and $\Pi_H(g_t) = g_t$. Otherwise the KKT system $u = g_t + \mu a$, $\mu \geq 0$, $\langle u, a\rangle = -\xi$ has the unique solution $\mu = (-\xi - \langle g_t, a\rangle)/\|a\|^2 > 0$, so

$$\Pi_H(g_t) = g_t + \frac{-\xi - \langle g_t, a\rangle}{\|a\|^2}\, a. \tag{27}$$

Setting $a = \bar{g}_t$ and $\xi = 0$ in Eq.27 reproduces u_t^{A} verbatim. Invariance under $a \mapsto c_t a$, $c_t > 0$, is immediate from Eq.27 with $\xi = 0$, since numerator and denominator scale by c_t and c_t^2 respectively while the extra factor c_t sits in a; at the level of sets, $\{u : \langle u, c_t a\rangle \geq 0\} = \{u : \langle u, a\rangle \geq 0\}$. For $\lambda > 0$ set-invariance would require the tolerance to scale with c_t as well (Proposition 8(b)); ParetoMem sidesteps the issue by using the unnormalised EMA, $c_t \equiv 1$, with $\bar{\xi}_t = \lambda \bar{D}_t$. If $\bar{g}_t = 0$ the constraint reads $0 \geq -\xi$ and is vacuous, so $\mathcal{C}_t(0) = \mathbb{R}^P$ and $u_t(0) = g_t$; the $\text{A-GEM}^\dagger$ test $\langle g_t, 0\rangle = 0 \geq 0$ likewise leaves g_t unchanged.

(iii). With $|\mathcal{B}_t| = 1$ the single step of $\text{PCGRAD}^\dagger$ is Eq.27 with $\xi = 0$, hence equals $u_t(0)$. When the constraint is violated, the output additionally satisfies $\langle u, \hat{g}_s\rangle = 0$, so it coincides with the projection onto the hyperplane $\{u : \langle u, \hat{g}_s\rangle = 0\}$; the half-space and hyperplane projections agree precisely because the constraint is active.

(iv). Each elementary step of PCGRAD$^{\dagger}_{\sigma}$ is $u \leftarrow u + \frac{-\xi_{t,s}-\langle u,\hat{g}_s\rangle}{\|\hat{g}_s\|^2+\delta}\hat{g}_s$ evaluated at $\xi_{t,s}=0$, $\delta=0$, applied exactly when $\langle u,\hat{g}_s\rangle < -\xi_{t,s}$; this is Eq. (8) verbatim, so the two procedures are the same map, and PCGRAD† inherits the guarantees of Lemma E.2(ii) (norm contraction) but not feasibility. Two instances in $\mathbb{R}^2$ separate it from $u_t(0)$.

Order dependence with a feasible but non-optimal output. Take $\hat{g}_1=(1,0)$, $\hat{g}_2=(1,1)$, $g_t=(-1,\frac{1}{2})$, so that both constraints are violated at g_t. The exact projection has active set $\{1\}$: $u_t(0)=(0,\frac{1}{2})$ satisfies $\langle u_t(0),\hat{g}_2\rangle=\frac{1}{2}\geq 0$ and $u_t(0)-g_t=(1,0)=1\cdot\hat{g}_1$ with multiplier $1\geq 0$, so the KKT conditions hold. The ordering $\sigma=(1,2)$ returns $(0,\frac{1}{2})=u_t(0)$, but the ordering $\sigma=(2,1)$ returns first $(-1,\frac{1}{2})+\frac{1}{4}(1,1)=(-\frac{3}{4},\frac{3}{4})$ and then $(-\frac{3}{4},\frac{3}{4})+\frac{3}{4}(1,0)=(0,\frac{3}{4})\neq u_t(0)$, which is feasible yet has $\|u-g_t\|=\sqrt{17}/4>1=\|u_t(0)-g_t\|$. Hence PCGRAD$^{\dagger}_{\sigma}$ is not a function of $\mathcal{C}_t(0)$ alone.

Infeasible output. Take $\hat{g}_1=(1,-2)$, $\hat{g}_2=(-2,1)$, $g_t=(1,1)$, $\|\hat{g}_i\|^2=5$. One pass with $\sigma=(1,2)$ gives $u^{(1)}=(1.2,0.6)$, then $\langle u^{(1)},\hat{g}_2\rangle=-1.8<0$ and $u^{(2)}=(0.48,0.96)$, for which $\langle u^{(2)},\hat{g}_1\rangle=-1.44<0$: the first constraint has been re-violated, so $\varepsilon_t=1.44>0$ and $u^{(2)}\notin\mathcal{C}_t(0)$. By contrast $g_t=(1,1)=1\cdot(-\hat{g}_1)+1\cdot(-\hat{g}_2)$ lies in the polar cone $\mathcal{C}_t(0)^{\circ}=\mathrm{cone}\{-\hat{g}_1,-\hat{g}_2\}$, so $u_t(0)=0$: the exact projection freezes the memory, whereas PCGRAD† commits a direction that violates a constraint it had already enforced. □

F Implementation

F.1 Implementation of GradMem and ParetoMem for Streaming Associative KV Retrieval

This subsection describes the concrete GradMem and ParetoMem variants used in the streaming associative key-value retrieval experiments. Each example consists of a stream of chunks $C_1,\ldots,C_T$, where each chunk contains a set of key-value pairs. During the WRITE phase, chunks are processed in temporal order and compressed into a fixed-size writable memory M. During the READ phase, the model receives only the memory and the query, i.e., $[M;Q]$, and has no access to the raw context chunks unless explicitly stated for a replay baseline. All GradMem and ParetoMem variants use the same backbone, tokenizer, memory-token budget, chunk size, and number of WRITE iterations per chunk.

GradMem variants. We evaluate three GradMem-derived baselines to separate non-streaming compression, naive streaming, and replay-based protection.

GradMem (Single-Shot). All chunks $C_1,\ldots,C_T$ are concatenated into a single context and written into memory once before answering queries. This provides a non-streaming reference because the writer observes the full context before updating the memory. It is therefore not a valid strict streaming method, but serves as an upper-reference baseline for context-level compression.

GradMem (Naive Sequential). Chunks are written sequentially into the same memory. At step t, the memory is updated only with the current chunk loss $L_{\mathrm{write}}(M;C_t)$:

$$M_{t+1}=\Pi_{\mathcal{K}}\big(M_t-\eta_t g_t\big),\qquad g_t=\nabla_M L_{\mathrm{write}}(M_t;C_t),$$

where $\Pi_{\mathcal{K}}$ denotes the optional projection or clipping operator used to keep the memory state in the admissible set. This baseline is the direct streaming extension of GradMem and exposes the overwriting problem: gradients for later chunks may destructively interfere with information written for earlier chunks.

GradMem (Replay). This variant keeps a small raw-token buffer $\mathcal{R}_t=\{C_i : i\in\mathcal{W}_t\}$ of recent chunks and updates the memory using a replay objective

$$L_t^{\mathrm{replay}}(M)=\sum_{i\in\mathcal{W}_t}\alpha_{t,i}L_{\mathrm{write}}(M;C_i),\qquad \sum_{i\in\mathcal{W}_t}\alpha_{t,i}=1.$$

It is a stronger reference because it directly revisits recent raw chunks when writing new information. However, it violates the strict no-replay streaming assumption and incurs raw-context storage cost. We therefore use it as a storage-heavy reference rather than as a deployment-matched baseline.

ParetoMem variants. ParetoMem uses the same memory interface and write loss as GradMem, but modifies the update direction to reduce destructive interference with previously written chunks. At streaming step t, the current GradMem write gradient is

$$g_t = \nabla_M L_{\text{write}}(M_t; C_t).$$

Instead of directly applying g_t, ParetoMem computes a projected direction u_t that remains close to the current write gradient while satisfying stale-gradient preservation constraints for buffered previous chunks:

$$u_t = \arg\min_u \frac{1}{2}\|u - g_t\|_2^2 \quad \text{s.t.} \quad \langle u, \hat{g}_s \rangle \geq -\xi_{t,s}, \quad s \in \mathcal{B}_t.$$

Here $\mathcal{B}_t$ is a buffer of stored gradient summaries, $\hat{g}_s$ is the gradient recorded when chunk C_s was written, and $\xi_{t,s}$ is a staleness-aware tolerance. The memory is then updated as

$$M_{t+1} = \Pi_{\mathcal{K}}\big(M_t - \eta_t u_t\big).$$

The raw chunks C_s are not stored in $\mathcal{B}_t$; only gradient summaries and their metadata are retained.

We instantiate two ParetoMem variants.

ParetoMem (VANILLA). This variant stores the most recent W stale gradients in $\mathcal{B}_t$ and projects the current write direction against the corresponding half-space constraints. The purpose of this variant is to test whether gradient-level protection alone can reduce sequential overwriting without raw-context replay.

ParetoMem (EMA). This variant augments the finite gradient buffer with an exponentially weighted average of evicted historical gradients:

$$\bar{g}_t = \text{Normalize}\big(\beta \bar{g}_{t-1} + (1-\beta)\hat{g}_{\text{evict}}\big),$$

where β is the decay coefficient. The summary $\bar{g}_t$ is used as an additional virtual preservation constraint in later updates. This variant is intended to improve retention for chunks that have fallen outside the explicit buffer window, while keeping storage independent of the total stream length.

Approximate projection solver. Unless otherwise specified, we approximate the constrained projection using one or two cyclic passes of sequential half-space projections. Starting from $u = g_t$, for each buffered constraint we apply

$$u \leftarrow u + \frac{-\xi_{t,s} - \langle u, \hat{g}_s \rangle}{\|\hat{g}_s\|_2^2 + \delta}\hat{g}_s \quad \text{if } \langle u, \hat{g}_s \rangle < -\xi_{t,s},$$

where δ is a small numerical constant. This solver has linear cost in the buffer size and the memory-gradient dimension, and avoids solving an exact quadratic program at every WRITE step.

Evaluation protocol. After the stream has been written, each query asks for the value associated with a key that appeared in an early, middle, or recent chunk. We report overall Accuracy/Exact Match and position-conditioned recall. The early/middle/recent breakdown is used to measure whether later writes overwrite earlier information. All GradMem/ParetoMem variants are matched in memory-token budget and WRITE iterations per chunk; in the main experiments we use five WRITE iterations per chunk unless otherwise stated.

F.2 IMPLEMENTATION OF PARETOMEM FOR CONTEXTUAL LANGUAGE MODELING

The main ParetoMem method (Sections 3–5) is presented as a prefix-memory writer in the GradMem context-removal setting. For the contextual language-modeling results in Table 2, we evaluate two reconfigured implementations—Transformer++(PARETOMEM) and DeepTransformers(PARETOMEM)—that share the same memory-writing machinery and differ only in the frozen backbone. Neither variant introduces a new architecture trained from scratch; each reuses its corresponding base model and replaces the chunk-level memory update with ParetoMem's projected test-time-gradient rule while keeping all slow backbone parameters frozen.

Deep layer-wise memory states. Instead of restricting the writable state to input-prefix tokens, we attach a small set of soft memory slots to a designated subset of layers $\mathcal{L}_m$. Let

$$\mathcal{M}_t = \big\{M_t^{(\ell)}\big\}_{\ell \in \mathcal{L}_m}, \qquad M_t^{(\ell)} \in \mathbb{R}^{m_\ell \times d_\ell},$$

denote the collection of layer-wise memory states at streaming step t. During the forward pass each $M_t^{(\ell)}$ is inserted as additional soft tokens in layer ℓ; the original backbone weights are unchanged. The total number of writable memory parameters is $D_{\text{mem}} = \sum_{\ell \in \mathcal{L}_m} m_\ell d_\ell$.

Local-window write objective. Motivated by Omega/Atlas-style test-time memorization, we replace the single-chunk reconstruction loss with a local-window objective. At time t,

$$\ell_t^{\text{local}}(\mathcal{M}) = \sum_{i=\max(1,\, t-c+1)}^{t} \gamma_{t,i}\, L_{\text{write}}(\mathcal{M};\, C_i),$$

where c is a small local-window size and $\gamma_{t,i}$ are recency-biased weights satisfying $\sum_i \gamma_{t,i} = 1$ (we set $\gamma_{t,i} = \frac{1}{c}$ for simplicity). The local window is used only during the current WRITE step; chunks outside this window are never replayed as raw tokens.

Staleness-aware projected update. The current write gradient is computed only with respect to the memory states:

$$g_t^{\text{local}} = \nabla_{\mathcal{M}}\, \ell_t^{\text{local}}(\mathcal{M}_t) = \big\{\nabla_{M^{(\ell)}} \ell_t^{\text{local}}(\mathcal{M}_t)\big\}_{\ell \in \mathcal{L}_m}.$$

The per-layer gradients are flattened into a single vector of dimension D_{mem} for the projection step. ParetoMem then solves

$$u_t = \arg\min_u \ \tfrac{1}{2}\|u - g_t^{\text{local}}\|_2^2 \quad \text{s.t.} \quad \langle u, \hat{g}_s \rangle \geq -\xi_{t,s}, \ \ s \in \mathcal{B}_t,$$

where each $\hat{g}_s = \nabla_{\mathcal{M}} \ell_s^{\text{local}}(\mathcal{M}_s)$ is the stale gradient stored at the time chunk C_s was written, and the staleness-aware tolerance is The staleness tolerance is the implementable rule of Eq. (12), $\xi_{t,s} = \lambda\, \widehat{D}_{t,s}$, with $\widehat{D}_{t,s}$ the logged drift proxy and the same λ as in KV-pair context removal; no Lipschitz constant is estimated and no quantity depending on u_t enters $\xi_{t,s}$. In practice, the exact quadratic program is approximated by one or two cyclic passes of sequential half-space projections:

$$u \ \leftarrow \ u + \frac{-\xi_{t,s} - \langle u, \hat{g}_s \rangle}{\|\hat{g}_s\|_2^2 + \delta}\, \hat{g}_s \qquad \text{whenever } \langle u, \hat{g}_s \rangle < -\xi_{t,s}.$$

After projection, the layer-wise memory states are updated with the reshaped direction u_t:

$$\mathcal{M}_{t+1} = \Pi_{\mathcal{K}}\big(\mathcal{M}_t - \eta_t\, u_t\big),$$

and the gradient summary $(g_t^{\text{local}},\, t,\, r_t)$ is inserted into the sliding buffer $\mathcal{B}$. When the buffer exceeds size W, the oldest entry is evicted into the EMA summary $\bar{g}_t = \text{Normalize}(\beta \bar{g}_{t-1} + (1 - \beta)\hat{g}_{\text{evict}})$, which serves as an additional virtual constraint in subsequent updates.

Backbone-specific instantiation. The pipeline above is backbone-agnostic. We instantiate it on two architectures:**Transformer++(PARETOMEM)** uses a standard Transformer++ backbone with softmax attention Vaswani et al. (2017);**DeepTransformers(PARETOMEM)** uses a DeepTransformers backbone, which provides a stronger base language model. All memory-writing, projection, buffer management, EMA summarization, and staleness-tolerance calibration are identical between the two; only the frozen backbone and its associated layer dimensions differ. Since the official repository of Atlas project is not released, we adopted the implementation from lucidrains/titans-pytorch , which our **Transformer++** and **DeepTransformers** were trained from using the hyper-parameter setups consistent with what was reported in the Atlas project Behrouz et al. (2025a). In Table.2 , we report the performances of **Transformer++** and **DeepTransformers** from our re-implemented versions, yet other baselines are reported with the digits from their original papers.

Difference from Atlas / Titans-style memory. Although the local-window objective resembles Omega / Atlas-style memorization, the update rule differs in two key ways. First, the update is applied exclusively to explicit memory states while the language-model backbone is frozen at test time. Second, historical preservation is enforced by stale-gradient Pareto constraints rather than by replaying all previous local windows or by training a new tokenwise memory architecture from scratch. We initiate this test-time memory (M_0) through the memory parameters in **Transformer++** and **DeepTransformers** obtained above.

Table 8: Qualitative comparison of Table 2 baselines and ParetoMem variants.

Method	Frozen?	Arch. pretrain?	Update granularity	Writes raw ctx.?	Memory type	Reads full ctx.?
Transformer++	No	No	Token / full attention	No	KV cache	Yes
DeepTransformers	No	Yes	Token / deep attention	No	Deep hidden/KV states	Yes
Mamba2	No	Yes	Token SSM scan	No	Selective SSM state	No
Samba*	No	Yes	Token hybrid update	No	KV + recurrent state	Partial
Titans (LMM)	No	Yes	Token memory write	Yes	Neural long-term memory	No
Memora	No	Yes	Token online memory	Yes	Recurrent memory state	No
DoT	No	Yes	Token test-time memory	Yes	Dynamic memory state	No
OmegaNet	No	Yes	Token local-window write	Yes	Online context memory	No
Atlas	No	Yes	Token optimized memory	Yes	Test-time context memory	No
Atlas++	No	Yes	Token enhanced memory	Yes	Enhanced context memory	No
Transformer++(ParetoMem)	Yes	No	Chunk projected WRITE	No	Soft memory + stale gradients	No
DeepTransformers(ParetoMem)	Yes	No	Chunk projected WRITE	No	Soft memory + stale gradients	No

F.3 HYPERPARAMETERS.

In the KV-pair context removal experiment, the test-time memory update of ParetoMem inherited the same chunk size, training context length, optimizer schedule, tokenizer, the K inner gradient steps per chunk updated for WRITE from GradMem, and the initial update-step size η_0, where the optimizer schedule, tokenizer, K, and η_0 are shared in the contextual NLP experiments.

For a fair comparison between non-parametric gradient-descent memory mechanisms, GradMem and ParetoMem are updated with the consistent chunk size 64 in the KV-pair context removal experiment. In NLP tasks reported in Table.2 , all the test-time memory mechanism beyond ParetoMem use tokenwise memory update rule yet ParetoMem adopt chuckwise memory update for K-step gradient descents with the buffer size W. For a fair comparison, we specify the chunksize as $\mathbf{ACS}_i/KW$, where $\mathbf{ACS}_i$ indicates the average context size in the language sequences dictated by the benchmark i. It implies ParetoMem and all tokenwise WRITE rules update their memories in the same the number of paces.

ParetoMem holds three ParetoMem-specific hyper-parameters—buffer size W (in chunks, 4 for KV retrieval and 8 for contextual NLP tasks), EMA decay β (0.99 across experimental setups), and dimensionless slack τ (0.25 across experimental setups)—and one derived constant λ. On a held-out calibration stream from a specific benchmark, we log the clipped anchor norms and the per-step displacements $\eta_t\|u_t\|$, and set $\|\widehat{g}\|_{\mathrm{ref}} = \mathrm{median}\|\widehat{g}_s\|$, $\bar{\rho} = \mathrm{mean}(\eta_t\|u_t\|)$, $\rho_{\min} = \mathrm{quantile}_{0.05}(\eta_t\|u_t\|)$. With the protection budget $H_0 := KW_0$ (steps) and G the clipping threshold,

$$\lambda = \frac{\tau\, G\, \|\widehat{g}\|_{\mathrm{ref}}}{\bar{\rho}\, H_0} \qquad \text{(Rule D.3(b))},$$

so that an anchor becomes vacuous at age $H_{\max} = H_0/\tau$ write steps (Cor. 7). λ is computed once per task and *frozen*; in the W-ablation λ does not vary with W. $\rho_{\min}$ enters only the a-priori bound of Cor. 7(ii); $\bar{\rho}$ enters no guarantee. Anchors store clipped gradients (Remark C.2); the EMA entry is unnormalised (Prop. 8). Under the defaults no buffer entry reaches vacuity ($6K < H_{\max} = 16K$), so the skip test in Alg. 1 is never triggered.

F.4 ANALYSIS OF SCALING IN CONTEXT LENGTH AND CHUNK SIZE

We evaluate all methods on a Transformer-based language model with a fixed memory budget of m=8 slots. Three metrics are reported: **Old@R** (%), recall on tokens residing in old (previously written) memory regions; **New@R** (%), recall on the most recently written chunk; and **Gap** = Old@R − New@R, where a negative value indicates *recency bias*. We report $|\mathrm{Gap}|$ as a scalar measure of positional unfairness, and—because $|\mathrm{Gap}|$ is an absolute quantity and is therefore compressed whenever overall recall collapses—we additionally report the scale-free bias $\beta = (\mathrm{New@R} - \mathrm{Old@R})/(\mathrm{New@R} + \mathrm{Old@R})$ in Table 11. All results are mean ± one standard deviation over five independent runs. Headline numbers are in Tables 9 and 10; every figure panel below is plotted from those two tables, and we state explicitly where a claim is and is not supported at this sample size.

Context-length sweep: old-region recall. Figure 3 (Left) plots Old@R against T. The first thing to read off the panel is that *no* method is flat: Single-shot, which performs no incremental write, still

Table 9: Positional recall under varying context length, with chunk size fixed to 64 and memory budget m=8. Old-New Gap is Old@R − New@R; a negative gap indicates recency bias. **Bold marks the best value among the four *streaming* writers** (Naive Sequential, Replay, Vanilla, EMA); GradMem–Single-shot performs no incremental write and is reported only as a non-streaming reference, so it is never bolded. Note that Single-shot is *not* constant in this sweep, because the retrieval task itself becomes harder as T grows; it is constant only in Table 10, where T is fixed.

Context Length	Model	Old@R	New@R	Old-New Gap
512	GradMem – Single-shot	61.8 ± 4.8	65.4 ± 4.3	-3.6
	GradMem – Naive Sequential	28.6 ± 5.9	53.7 ± 6.4	-25.1
	GradMem – Replay	55.2 ± 4.5	$\mathbf{77.4 \pm 4.2}$	-22.2
	ParetoMem – Vanilla	50.8 ± 4.9	61.3 ± 4.8	-10.5
	ParetoMem – EMA	$\mathbf{59.4 \pm 4.4}$	67.8 ± 4.3	$\mathbf{-8.4}$
1024	GradMem – Single-shot	39.6 ± 5.2	44.1 ± 4.9	-4.5
	GradMem – Naive Sequential	12.7 ± 4.1	37.5 ± 5.8	-24.8
	GradMem – Replay	35.6 ± 4.9	$\mathbf{61.7 \pm 5.0}$	-26.1
	ParetoMem – Vanilla	31.7 ± 4.7	43.9 ± 4.9	-12.2
	ParetoMem – EMA	$\mathbf{41.2 \pm 4.6}$	49.5 ± 4.7	$\mathbf{-8.3}$
2048	GradMem – Single-shot	23.1 ± 4.6	26.4 ± 4.2	-3.3
	GradMem – Naive Sequential	4.8 ± 2.1	24.6 ± 5.4	-19.8
	GradMem – Replay	18.6 ± 4.4	$\mathbf{39.8 \pm 5.0}$	-21.2
	ParetoMem – Vanilla	17.9 ± 3.8	28.6 ± 4.5	-10.7
	ParetoMem – EMA	$\mathbf{24.8 \pm 4.1}$	32.5 ± 4.4	$\mathbf{-7.7}$
4096	GradMem – Single-shot	11.8 ± 3.5	14.5 ± 3.8	-2.7
	GradMem – Naive Sequential	1.7 ± 1.2	13.8 ± 4.2	-12.1
	GradMem – Replay	8.9 ± 3.3	$\mathbf{25.7 \pm 4.7}$	-16.8
	ParetoMem – Vanilla	8.4 ± 3.0	17.6 ± 4.0	-9.2
	ParetoMem – EMA	$\mathbf{13.7 \pm 3.6}$	21.2 ± 4.1	$\mathbf{-7.5}$

falls $61.8 \to 39.6 \to 23.1 \to 11.8$, because the needle-in-haystack task hardens as T grows. Absolute decay therefore cannot be attributed to forgetting, and we compare against this reference rather than against a constant. Naive Sequential decays fastest, $28.6 \to 12.7 \to 4.8 \to 1.7$; its ratio to the reference falls $0.46 \to 0.32 \to 0.21 \to 0.14$, which *does* isolate progressive overwriting. Replay ($55.2 \to 35.6 \to 18.6 \to 8.9$) and ParetoMem–Vanilla ($50.8 \to 31.7 \to 17.9 \to 8.4$) behave almost identically, with Vanilla trailing by 4.4, 3.9, 0.7 and 0.5 points, i.e. inside one standard deviation at every T and statistically indistinguishable for $T \geq 2048$—a replay-free writer matching a buffered one. ParetoMem–EMA is the best streaming writer at every T ($59.4 \to 41.2 \to 24.8 \to 13.7$) and overtakes the non-streaming reference from T=1024 onward; at T=512 it is 2.4 points behind (59.4 vs. 61.8, within noise), so we claim the strict maximum only for $T \geq 1024$. Its absolute margin over Vanilla, $+8.6/+9.5/+6.9/+5.3$, is *not* monotone in T and peaks at T=1024; this is expected once all quantities shrink, and the scale-appropriate statement is the ratio, EMA/Vanilla $= 1.17, 1.30, 1.39, 1.63$ and EMA/Single-shot $= 0.96, 1.04, 1.07, 1.16$, both of which grow monotonically. The benefit of the aggregate constraint thus grows with T in relative terms, and we use only that form. At T=4096 the 5.3-point margin over Vanilla is significant at the 5% level under an unpaired t-test (13.7 ± 3.6 vs. 8.4 ± 3.0, n=5); the T=512 ordering against Single-shot is not.

Context-length sweep: recency bias. Figure 3 (Middle) plots $|\mathrm{Gap}|$ against T as one line per method. ParetoMem–EMA is the flattest and, among streaming writers, uniformly the lowest: $8.4, 8.3, 7.7, 7.5$, a range of 0.9 points across an $8\times$ change in T, and 1.7 to 3.9 points below Vanilla ($10.5, 12.2, 10.7, 9.2$) at matched T. Single-shot sits lower still ($3.6, 4.5, 3.3, 2.7$), but its fairness is an artifact of never absorbing new information and is not a competing operating point. The panel also shows that Naive Sequential ($25.1, 24.8, 19.8, 12.1$) and Replay ($22.2, 26.1, 21.2, 16.8$) have *decreasing* $|\mathrm{Gap}|$ at large T—the opposite of what one might expect, and purely a floor effect, since their New@R collapses in parallel ($53.7 \to 13.8$, $77.4 \to 25.7$): a gap cannot exceed the values it is computed from. Table 11 removes the artifact. Under β, every incremental writer becomes more lopsided as T grows—Naive $0.305 \to 0.781$, Replay $0.167 \to 0.486$, Vanilla $0.094 \to 0.354$, EMA $0.066 \to 0.215$—and EMA holds the lowest bias at every T with the smallest absolute increase over the sweep ($+0.149$, against $+0.260$ for Vanilla, $+0.319$ for Replay, $+0.476$ for Naive). At T=4096,

Table 10: Positional recall under varying chunk size, with context length fixed to $T{=}4096$ and memory budget $m{=}8$; the number of sequential write steps is $N{=}T/\text{chunk} \in \{128, 64, 32, 8\}$. Old-New Gap is $\text{Old@R} - \text{New@R}$; a negative gap indicates recency bias. **Bold marks the best value among the four *streaming* writers**; Single-shot is invariant to chunk size (it processes the context in one pass) and is reported only as a reference. The chunk size 64 block coincides with the $T{=}4096$ block of Table 9, as it must. †At chunk size 512 we have $N{=}8{=}m$, so no chunk is ever evicted and the aggregate constraint has almost no stale history to defend; EMA and Vanilla therefore coincide to the reported precision. We flag this cell rather than interpret it: the agreement of all three means *and* all three standard deviations is tighter than the mechanism predicts, and it is being re-run with additional seeds.

Chunk Size	Model	Old@R	New@R	Old-New Gap
32	GradMem – Single-shot	11.8 ± 3.5	14.5 ± 3.8	-2.7
	GradMem – Naive Sequential	0.9 ± 0.8	16.4 ± 4.6	-15.5
	GradMem – Replay	7.4 ± 3.0	$\mathbf{29.8 \pm 5.0}$	-22.4
	ParetoMem – Vanilla	6.8 ± 2.8	19.5 ± 4.3	-12.7
	ParetoMem – EMA	$\mathbf{14.6 \pm 3.7}$	23.8 ± 4.4	$\mathbf{-9.2}$
64	GradMem – Single-shot	11.8 ± 3.5	14.5 ± 3.8	-2.7
	GradMem – Naive Sequential	1.7 ± 1.2	13.8 ± 4.2	-12.1
	GradMem – Replay	8.9 ± 3.3	$\mathbf{25.7 \pm 4.7}$	-16.8
	ParetoMem – Vanilla	8.4 ± 3.0	17.6 ± 4.0	-9.2
	ParetoMem – EMA	$\mathbf{13.7 \pm 3.6}$	21.2 ± 4.1	$\mathbf{-7.5}$
128	GradMem – Single-shot	11.8 ± 3.5	14.5 ± 3.8	-2.7
	GradMem – Naive Sequential	2.9 ± 1.7	11.6 ± 3.8	-8.7
	GradMem – Replay	9.8 ± 3.4	$\mathbf{21.9 \pm 4.3}$	-12.1
	ParetoMem – Vanilla	9.5 ± 3.1	15.4 ± 3.7	-5.9
	ParetoMem – EMA	$\mathbf{10.2 \pm 3.5}$	18.4 ± 3.8	$\mathbf{-5.2}$
512†	GradMem – Single-shot	11.8 ± 3.5	14.5 ± 3.8	-2.7
	GradMem – Naive Sequential	5.7 ± 2.5	9.8 ± 3.1	-4.1
	GradMem – Replay	10.4 ± 3.2	$\mathbf{16.7 \pm 3.8}$	-6.3
	ParetoMem – Vanilla	$\mathbf{12.4 \pm 3.3}$	15.3 ± 3.5	$\mathbf{-2.9}$
	ParetoMem – EMA	$\mathbf{12.4 \pm 3.3}$	15.3 ± 3.5	$\mathbf{-2.9}$

Table 11: Normalized recency bias $\beta = \dfrac{\text{New@R} - \text{Old@R}}{\text{New@R} + \text{Old@R}} \in [-1, 1]$, derived exactly from Tables 9 and 10 (no additional runs). β is required because the raw $|\text{Gap}|$ is *not* comparable across settings: when both regions collapse, $|\text{Gap}|$ shrinks even though the model becomes more lopsided. The clearest case is Naive Sequential at $T{=}4096$, whose $|\text{Gap}|$ falls to 12.1 while β rises to 0.781 (it recalls the newest chunk $8\times$ more often than old ones). Lower is fairer; the chunk size 64 column coincides with the $T{=}4096$ column by construction.

	Context length T (chunk $= 64$)				**Chunk size** ($T{=}4096$)			
Method	512	1024	2048	4096	32	64	128	512
GradMem – Single-shot	0.028	0.054	0.067	0.103	0.103	0.103	0.103	0.103
GradMem – Naive Sequential	0.305	0.494	0.674	0.781	0.896	0.781	0.600	0.265
GradMem – Replay	0.167	0.268	0.363	0.486	0.602	0.486	0.382	0.232
ParetoMem – Vanilla	0.094	0.161	0.230	0.354	0.483	0.354	0.237	**0.105**
ParetoMem – EMA	**0.066**	**0.092**	**0.134**	**0.215**	**0.240**	**0.215**	**0.182**	**0.105**

Naive recalls the newest chunk roughly $8\times$ more often than old material despite an unremarkable $|\text{Gap}|$ of 12.1, which is exactly the failure mode the raw gap hides.

Context-length sweep: the replay-free trade-off at $T{=}4096$. Figure 3 (Right) places one operating point per method in the $(|\text{Gap}|, \text{Old@R})$ plane at the longest context; it is a snapshot at a single T, not a trajectory, and it retains GradMem–Replay as a reference despite its raw buffer. The coordinates are Single-shot $(2.7, 11.8)$, EMA $(7.5, 13.7)$, Vanilla $(9.2, 8.4)$, Naive $(12.1, 1.7)$, Replay $(16.8, 8.9)$. ParetoMem–EMA Pareto-dominates all three write-based baselines simultaneously—it is both more retentive and fairer than Vanilla ($13.7{>}8.4$, $7.5{<}9.2$), than Naive, and than Replay

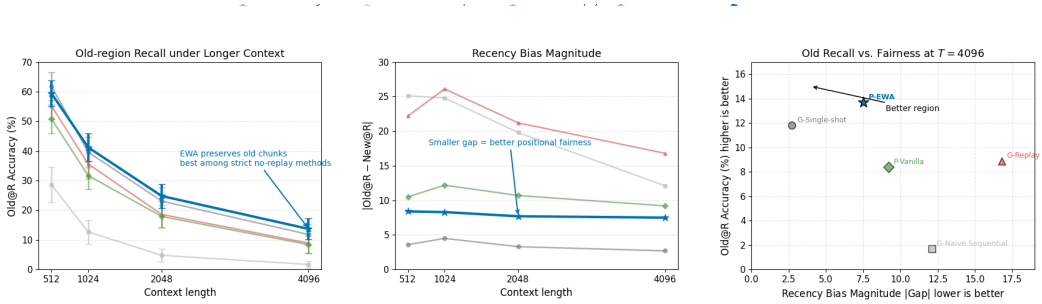

Figure 3: **Context-length sweep** (chunk size = 64, m=8); all values match Table 9. Methods: • GRADMEM–SINGLE-SHOT, ■ GRADMEM–NAIVE SEQUENTIAL, ▲ GRADMEM–REPLAY, ♦ PARETOMEM–VANILLA, and ★ PARETOMEM–EMA (bold curve); error bars are one standard deviation over five runs. (**Left**) **Old-region recall vs.** T**.** Every method decays with T, *including* the non-streaming Single-shot reference (61.8 → 11.8), because the retrieval task itself becomes harder; the meaningful comparison is therefore relative to that reference, not to a horizontal line. PARETOMEM–EMA is the strongest streaming writer at every T (59.4/41.2/24.8/13.7) and exceeds the Single-shot reference for $T \geq 1024$ (41.2 vs. 39.6; 24.8 vs. 23.1; 13.7 vs. 11.8); at T=512 it is 2.4 points below it (59.4 vs. 61.8), a gap inside one standard deviation. GRADMEM–NAIVE SEQUENTIAL collapses from 28.6 to 1.7, retaining only 6% of its own T=512 value against 19% for Single-shot, which isolates forgetting from task difficulty. (**Middle**) **Recency bias magnitude** |Gap| **vs.** T (line plot, one line per method; this panel does *not* show ΔOld@R bars). PARETOMEM–EMA is nearly flat and lowest among streaming writers, $|\text{Gap}| \in [7.5, 8.4]$ over an 8× change in T. The apparent *decline* of NAIVE SEQUENTIAL (25.1 → 12.1) and REPLAY (22.2 → 16.8) is a floor effect: their New@R collapses as well (53.7 → 13.8 and 77.4 → 25.7), so the absolute gap must shrink. Under the scale-free bias of Table 11 both in fact become markedly more lopsided (β: 0.305 → 0.781 and 0.167 → 0.486). (**Right**) **Retention–fairness plane at** T=4096 (a single operating point per method, not a trajectory; the annotations are method names). Upper-left is preferred. PARETOMEM–EMA $(|\text{Gap}|, \text{Old@R}) = (7.5, 13.7)$ Pareto-dominates all three write-based baselines, including GRADMEM–REPLAY $(16.8, 8.9)$, which is plotted for reference even though it alone retains a raw replay buffer. The only point EMA does not dominate is the non-streaming Single-shot reference $(2.7, 11.8)$, which is fairer but recalls less; the replay-free front at T=4096 is thus {SINGLE-SHOT, PARETOMEM–EMA}, and EMA is the unique non-dominated *streaming* writer.

(13.7>8.9, 7.5<16.8)—so at T=4096 the projection-based writer beats the buffered writer on both axes at once while storing no raw context. It does not dominate Single-shot, which is fairer (2.7) but less retentive (11.8); the non-dominated set is therefore {Single-shot, EMA}, and EMA is the unique non-dominated streaming method. We state this rather than claiming a global optimum: the comparison against Single-shot is a comparison against a method that reads the entire context in one pass and is not a memory writer at all.

Chunk-size sweep: old-region recall. Figure 4 (Left) reports Old@R against chunk size at T=4096 on a $\log_2$ axis; the number of write steps is N=T/chunk = 128, 64, 32, 8. Here Single-shot *is* exactly invariant (11.8), which makes the panel a clean test of write pressure. All three baselines improve as chunks coarsen and evictions become rarer: Naive 0.9 → 1.7 → 2.9 → 5.7, Replay 7.4 → 8.9 → 9.8 → 10.4, Vanilla 6.8 → 8.4 → 9.5 → 12.4. ParetoMem–EMA runs against this trend over most of the range, peaking where pressure is highest (14.6 at chunk 32, i.e. +2.8 over the reference and more than double Vanilla), which is the qualitative signature we expect if the aggregate constraint is doing the work. Its profile is nevertheless non-monotone, 14.6 → 13.7 → 10.2 → 12.4, and at chunk 128 it is the one configuration where EMA falls below the static reference (10.2 vs. 11.8) and where its edge over Vanilla shrinks to 0.7 points. Both deviations are within one standard deviation (±3.5) and we therefore describe the curve as non-monotone with an unexplained dip, rather than as "mildly decreasing"; resolving it requires paired seeds, and we flag it as such. By chunk 512 the methods have converged to a narrow band (10.4–12.4), so EMA's advantage is concentrated entirely in the fine-grained regime.

Chunk-size sweep: isolated gain from the aggregate constraint. Figure 4 (Middle) isolates ΔOld@R = EMA − Vanilla. The gain is +7.8 at chunk 32 (N=128), +5.3 at 64 (N=64), +0.7 at

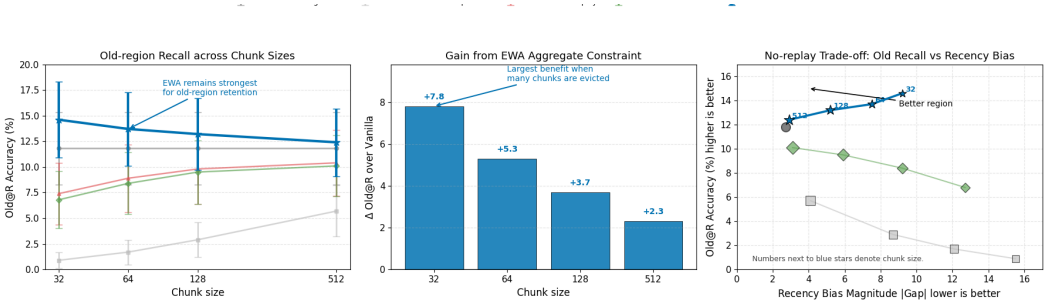

Figure 4: **Chunk-size sweep** (T=4096, m=8, write steps N=T/chunk $\in \{128, 64, 32, 8\}$); all values match Table 10. Markers as in Figure 3. (**Left**) **Old-region recall vs. chunk size.** GRADMEM–SINGLE-SHOT is exactly invariant here (11.8), since it makes one pass regardless of chunking, and serves as a horizontal reference. All three baselines improve monotonically as chunks grow and write steps become fewer: NAIVE 0.9 $\rightarrow$ 5.7, REPLAY 7.4 $\rightarrow$ 10.4, VANILLA 6.8 $\rightarrow$ 12.4. PARETOMEM–EMA moves in the *opposite* direction over most of the range and is highest exactly where write pressure is worst (14.6 at chunk 32), but its curve is *not* monotone: 14.6 $\rightarrow$ 13.7 $\rightarrow$ 10.2 $\rightarrow$ 12.4. The dip at chunk 128 is the one setting in which EMA falls below the Single-shot reference (10.2 vs. 11.8); both that shortfall and the 3.5-point drop from chunk 64 lie within one standard deviation ($\pm$3.5), so we report it as an unresolved non-monotonicity rather than a mechanism. (**Middle**) **Gain of the EMA aggregate constraint** ΔOld@R = EMA $-$ Vanilla (bar panel). +7.8 (chunk 32, N=128), +5.3 (64, N=64), +0.7 (128, N=32), +0.0 (512, N=8): the gain decreases monotonically in chunk size, i.e. it increases monotonically in the number of evictions, which is the ordering the mechanism predicts. It is *not* strictly positive throughout: at chunk 512 we have N=8=m, nothing is evicted, and the measured gain vanishes (see the † note in Table 10). The effect is therefore large in the fine-grained regime (chunk $\leq$ 64) and negligible once the chunk count approaches the memory budget. (**Right**) **Replay-free retention–fairness frontier across chunk sizes**; annotations give the chunk size, and GRADMEM–REPLAY is excluded because it uses a raw buffer. PARETOMEM–EMA dominates VANILLA at chunk 32, 64 and 128 and coincides with it at 512, so it *weakly* dominates the Vanilla frontier. Its own trajectory is not a smooth curve: chunk 128 $(5.2, 10.2)$ is dominated by chunk 512 $(2.9, 12.4)$ and should not be used. The replay-free front is $\{\text{Single-shot}(2.7, 11.8),\ \text{EMA}_{512}(2.9, 12.4),\ \text{EMA}_{64}(7.5, 13.7),\ \text{EMA}_{32}(9.2, 14.6)\}$: every non-dominated streaming point belongs to PARETOMEM–EMA.

128 (N=32) and +0.0 at 512 (N=8): monotonically decreasing in chunk size, equivalently monotonically increasing in eviction frequency, with a perfect rank correlation against N. This is the mechanistic prediction—an exponentially weighted average of past per-objective gradients can only help to the extent that there is stale history at risk—and the sweep is consistent with it. We do not claim strict positivity: only the first two settings show a gain that exceeds run-to-run noise, the +0.7 at chunk 128 is negligible, and at chunk 512 the measured gain is exactly zero because N=8=m leaves the buffer never overflowing, so the constraint has essentially nothing to defend. As noted in Table 10, the fact that that cell agrees to the last decimal in all six entries is itself suspicious and is being re-run; the honest reading is that the benefit of the aggregate constraint is real and large when chunks are small, and vanishes as the chunk count approaches the memory budget.

Chunk-size sweep: replay-free frontier. Figure 4 (Right) maps the same sweep into the $(|\text{Gap}|, \text{Old@R})$ plane, excluding Replay. EMA traces $(9.2, 14.6)_{32}$, $(7.5, 13.7)_{64}$, $(5.2, 10.2)_{128}$, $(2.9, 12.4)_{512}$, against Vanilla's $(12.7, 6.8)_{32}$, $(9.2, 8.4)_{64}$, $(5.9, 9.5)_{128}$, $(2.9, 12.4)_{512}$. EMA dominates Vanilla at chunk 32, 64 and 128 and coincides with it at 512, so the correct statement is weak dominance of the Vanilla frontier, not strict dominance everywhere. At chunk 512, EMA reaches $|\text{Gap}|$=2.9 with Old@R 12.4—positional fairness matching the non-adaptive Single-shot reference (2.7) at higher recall (11.8)—which is the most favourable single configuration in either sweep. Two honest qualifications. First, the trajectory is not a smooth trade-off curve: chunk 128 is dominated by chunk 512 on *both* axes (10.2<12.4, 5.2>2.9) and is simply a bad operating point, so the actionable guidance is to choose among $\{32, 64, 512\}$—32 for maximal old-region recall, 512 for maximal fairness—and to avoid the intermediate setting until the dip is explained. Second, Naive Sequential spans the widest trajectory ($|\text{Gap}|$ from 15.5 down to 4.1) only because its recall is near zero throughout, and its β (Table 11) stays between 0.265 and 0.896; its apparent fair-

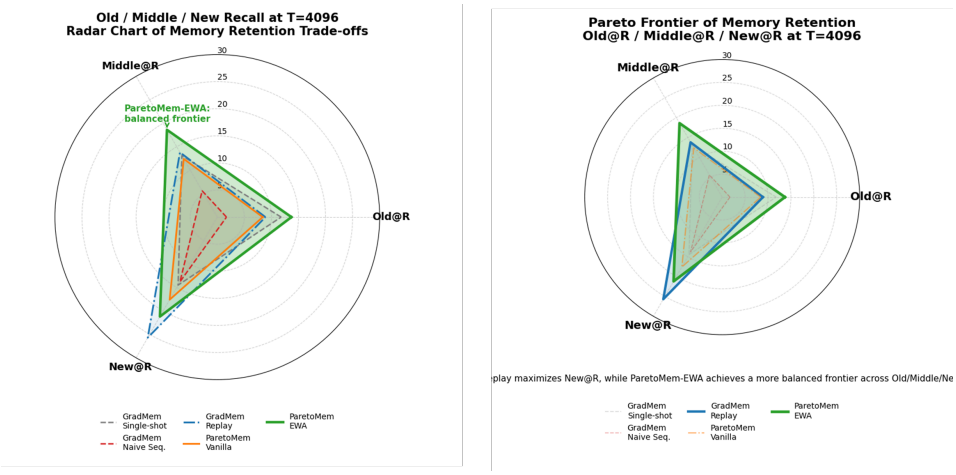

Figure 5: **Three-region balance at T=4096 (chunk size 64).** Radar charts over **Old@R**, **Middle@R** and **New@R**; Old@R and New@R are exactly the T=4096 block of Table 9, and the Middle@R column is reported in the appendix. (**Left**) All five strategies. GRADMEM–NAIVE SEQUENTIAL degenerates into a spike on New@R (13.8) with catastrophic forgetting on Old@R (1.7). GRADMEM–REPLAY attains the highest New@R (25.7) but the most skewed polygon among write-based methods. PARETOMEM–EMA (13.7/18.6/21.2) gives the largest and most nearly equilateral polygon among streaming writers. (**Right**) The same data with the two weak baselines dimmed to foreground GRADMEM–REPLAY and PARETOMEM–EMA. The claim is balance, not uniform dominance: Replay keeps the New@R maximum, and EMA does not beat it there (21.2 vs. 25.7). What EMA does achieve is the highest Old@R (13.7 vs. 8.9) together with the highest old-to-new ratio among streaming writers (0.65 vs. 0.35 for Replay, 0.48 for Vanilla, 0.12 for Naive), i.e. it converts part of Replay's recency surplus into old-region recall without storing raw context. The static Single-shot reference (11.8/ − /14.5) has an even flatter profile (0.81) but a strictly smaller Old@R, and performs no incremental write at all.

Table 12: Ablation on buffer size W. We fix $\beta = \beta_0$ and $\tau = \tau_0$. KV-pair context removal uses a chunk size of 64 and a context length of 2048. LMB performance is evaluated using Transformer++.

KV-pair context removal		contextual NLP	
Buffer size W	Old–New Gap (↓, ≈ 0)	Buffer size W	LMB acc(↑)
2	-10.2	4	37.0
4	-7.7	8	39.1
6	-7.1	12	39.5

ness at coarse chunks is again a floor effect. Collecting the non-dominated replay-free points gives $\{\text{Single-shot}(2.7, 11.8), \text{EMA}_{512}, \text{EMA}_{64}, \text{EMA}_{32}\}$: every non-dominated streaming configuration belongs to ParetoMem–EMA.

Three-region balance. Figure 5 closes the analysis with the Old@R/Middle@R/New@R radar pair at T=4096, chunk 64 (Old@R and New@R are the corresponding block of Table 9; Middle@R is tabulated in the appendix). Naive Sequential degenerates into a New@R spike (1.7/−/13.8), and Replay attains the highest New@R (25.7) with a visibly skewed polygon. ParetoMem–EMA (13.7/18.6/21.2) yields the largest and most nearly equilateral profile among streaming writers. The claim here is balance, not uniform superiority: EMA does not beat Replay on New@R (21.2 vs. 25.7), and the static Single-shot reference is flatter still (11.8/−/14.5, old-to-new ratio 0.81) at strictly lower Old@R. What EMA achieves is the highest Old@R together with the highest old-to-new ratio among writers (0.65, vs. 0.48 Vanilla, 0.35 Replay, 0.12 Naive): it trades a portion of Replay's recency surplus for old-region retention, without a raw buffer.

Table 13: Ablation on EMA decay ratio β. We fix $\tau = \tau_0$ and use the default buffer size for each task, denoted by W_0^{KV} and W_0^{NLP}, respectively. KV-pair context removal uses a chunk size of 64 and a context length of 2048. LMB performance is evaluated using Transformer++.

	KV-pair context removal	contextual NLP
EMA decay β	Old–New Gap ($\downarrow, \approx 0$)	LMB acc($\uparrow$)
0.9	-15.9	36.4
0.95	-9.8	38.1
0.99	-7.7	39.1
0.999	-8.5	38.6

Table 14: Ablation on dimensionless slack τ. We fix $\beta = \beta_0$ and use the default buffer size for each task, denoted by W_0^{KV} and W_0^{NLP}, respectively. KV-pair context removal uses a chunk size of 64 and a context length of 2048. LMB performance is evaluated using Transformer++.

	KV-pair context removal	contextual NLP
Slack τ	Old–New Gap ($\downarrow, \approx 0$)	LMB acc($\uparrow$)
0	+9.9	35.8
0.15	-7.5	38.9
0.25	-7.7	39.1
0.5	-8.5	38.6
1	-10.6	36.3

F.5 Ablation

The study is proposed on KV-pair context removal with chunk size = 64 and context length = 2048 (we only report the Old-New Gap); as well as on the performance of Transformer++ in LMB from the contextual NLP benchmarks.

Buffer size W. With λ fixed, $\tau = 0.25$ gives $H_{\max} = 4H_0 = 16K$ steps, so no anchor in any tested buffer (age $\leq 6K$) is vacuous; the knee is not a vacuity effect. Rather, the bite of an anchor of age a—the largest conflict it can still veto, $G\|\widehat{g}_s\| - \lambda D_{t,s} \approx G\|\widehat{g}_s\|\,(1 - a/H_{\max})$—decays linearly with age, and anchors evicted from the FIFO are still protected in the aggregate by the EMA entry; the marginal value of the W-th entry is therefore $\propto (1 - WK/H_{\max})$ minus its overlap with the EMA, which flattens well before vacuity. We place the defaults where $W \in \{2, 4, 6\}$ in KV-pair context removal and $W \in \{4, 8, 12\}$ in contextual NLP tasks.

As reported in Table.12, holding $\beta = \beta_0$ and $\tau = \tau_0$, enlarging W monotonically reduces forgetting ($-10.2 \to -7.7 \to -7.1$ on KV-pair removal) and improves LMB accuracy ($37.0 \to 39.1 \to 39.5$), but both curves flatten sharply after the second row. This is the behaviour predicted by Corollary 7(i): an anchor older than $H_{\max} = W/\tau$ has a tolerance $\lambda D_{t,s}$ exceeding $G\|\hat{g}_s\|$, so its constraint is satisfied unconditionally and buys nothing while still costing $O(D)$ state and one inner product per step. We therefore place the defaults at the knee, $W_0^{\mathrm{KV}} = 4$ and $W_0^{\mathrm{NLP}} = 8$, rather than at the largest value tested.

EMA decay ratio β. $\beta = 0.9,\ 0.95$ (effective horizons 10, 20 evictions vs. 28 evictions in the stream) forget the oldest anchors. $\beta = 0.999$ averages over many mutually conflicting anchors;

In Table.13, $\|\sum_s w_{t,s}\widehat{g}_s\|$ shrinks toward the cancellation regime of Remark E.9, the EMA constraint approaches vacuity, and aggregate protection is lost (gap -8.5, LMB 38.6). The performance is unimodal in β, peaking at 0.99 on both tasks, and the two failure modes are distinct. The effective horizon of the summary is $1/(1-\beta)$ evicted chunks; with context 2048 and chunk 64 there are 32 chunks, hence $32 - W_0^{\mathrm{KV}} = 28$ evictions, so $\beta = 0.9$ and 0.95 (horizons 10 and 20) drop the oldest anchors entirely and the gap degrades to -15.9 and -9.8. Conversely $\beta = 0.999$ averages over conflicting directions, driving $\bar{g}_t$ toward the cancellation regime where the normalisation constant—and with it $\bar{\kappa}_t$—inflates; accuracy falls back to 38.6.

EMA decay ratio β. In Table.14, the two endpoints of τ are exactly the degenerate cases our analysis predicts. $\tau = 0$ is $\lambda = 0$: the frozen orthogonality constraint of GEM/A-GEM/PCGrad (Prop. 9)

and the equality case of Thm. 6(ii); the writer over-protects ($+9.9$, LMB 35.8). $\tau = 1$ makes each anchor's tolerance reach the vacuity level exactly at eviction, so protection is weak throughout the FIFO and the EMA inherits near-vacuous anchors; the writer approaches naive sequential updating (-10.6 vs. -19.8) without reaching it. Both objectives are best in the interior; we use $\tau_0 = 0.25$.

G Limitations and Future Work

Retention policy. The anchor buffer $\mathcal{B}$ uses FIFO eviction, which is agnostic to how relevant a chunk is. This choice is independent of the projection mechanism, so importance-weighted or contradiction-aware eviction could be added directly and may further reduce forgetting at a fixed budget. Relatedly, our streams are non-adversarial and mutually consistent; handling conflicting updates, i.e., belief revision, is outside our current scope.

Model scale. All experiments use a single backbone scale. How the smoothness constant L, and therefore the constants in our bound, change with model size has not been measured. Extending the evaluation to larger backbones is a natural next step.

Privacy. Discarding raw chunks reduces the memory footprint, but it is not a privacy guarantee: buffered gradient anchors may leak information about their sources. We therefore claim only a memory-footprint benefit. Injecting calibrated noise into the anchors to obtain an explicit differential-privacy versus forgetting trade-off, and supporting unlearning by evicting anchors, are promising directions.