SELF-GENERATED FEEDBACK DESTABILIZES TEST-TIME TRAINING: A CAUSAL DECOMPOSITION OF LONG-HORIZON ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time training (TTT) gives a model writable state: information can persist in its weights after the source text leaves the attention window. The same mechanism creates feedback when the model learns from its own output: each update changes the model that generates the next training example. We study this feedback over 128K-token streams. Retaining generated-text writes worsens prediction of independent human-written text at three TTT-E2E scales (125M, 760M, and 3B), and the same failure occurs when Adam updates Qwen3-4B's existing weights. Three matched comparisons provide a causal decomposition of the result. First, *Fixed Generation*, in which a frozen model generates every training chunk, removes 97.5% of the measured damage even though the adapting model continues to update. Second, *Recorded Replay* of the same degraded text shows that reading it through attention causes one loss, while retaining its updates stores an additional loss in the weights. Third, branching from an identical state before one update reveals the local conflict: the update predicts its source chunk better but predicts new real text worse. This cost grows after closed-loop adaptation, and a few source trajectories account for most large failures. *Settlement* evaluates the complete candidate state on independent real text before commitment, reducing the endpoint gap to at most .07 nats at 125M and 760M while retaining real-text adaptation. Thus, optimizing the current self-generated example can make later, independent inputs harder to predict.

1 INTRODUCTION

Test-time training (TTT) lets part of a model continue to learn during inference (Sun et al., 2020; Liu et al., 2021; Sun et al., 2025). In long-context language models, these changing parameters—the *fast weights*—can retain information after the source text leaves attention (Schmidhuber, 1992; Ba et al., 2016; Schlag et al., 2021; Tandon et al., 2025; Behrouz et al., 2025). This makes TTT attractive for long documents, streams, and long-running agents, where retaining the full history in attention is costly (Tandon et al., 2025; Behrouz et al., 2025; Wang et al., 2026). We call processing text without changing weights *reading*, and retaining an update learned from it *writing*.

Writing becomes risky when the model also supplies its future training data. Fast weights W_t generate chunk x_t; learning from x_t produces W_{t+1}; and the new weights generate the next chunk. In this feedback loop, $W_t \to x_t \to W_{t+1}$, lower loss on x_t shows only that the update fits its source, not that it predicts other text better. We ask which part of this loop makes persistent self-writing fail.

We measure this failure over 128K-token streams. Matched models retain or discard each generated-text update and are evaluated on separate human-written passages; the comparison therefore asks whether writing helps beyond the model's own output. In End-to-End Test-Time Training (TTT-E2E) (Tandon et al., 2025), retaining generated-text writes worsens real-text prediction at 125M, 760M, and 3B. The effect emerges beyond the original evaluation horizon and also occurs when Adam updates Qwen3-4B's existing feed-forward weights (Kingma & Ba, 2015; Yang et al., 2025); it is not specific to TTT-E2E's update module.

We identify the cause through a causal decomposition built from three matched comparisons. **First, we remove feedback to future training text.** In *Fixed Generation*, a frozen model generates each

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.

chunk while a separate model learns from it. Because updates cannot change later training text, 97.5% of the damage disappears despite greater weight movement. **Second, we hold the text fixed.** In *Recorded Replay*, a new model processes a saved degraded stream. The tokens hurt through attention, and retaining their updates adds a persistent weight cost, including at 3B. **Third, we isolate one update.** From identical weights, attention, and text, one branch retains the update and the other discards it. The retained update predicts its source better but predicts the next real passage worse. The cost is larger after Closed Loop adaptation, and a few trajectories produce most large failures. Unlike a static synthetic-data comparison, each update changes both the generator supplying the next example and the state from which the next update begins. A canonical decoder sweep shows that a within-chunk repetition penalty can slow this global feedback trajectory, reducing the 128K endpoint gap from 3.01 to $.86$ nats. Yet repetition still does not identify which fixed passage is safe to write. We instead evaluate the complete candidate state on independent real text. We call this rule *Settlement*. It leaves at most $.07$ nats of endpoint gap at 125M and 760M while preserving real-text adaptation. Unlike source masking, it evaluates the candidate state and remains stable under noisy labels.

Prior work has studied recursive synthetic-data training across successive models (Briesch et al., 2023; Shumailov et al., 2024), downweighted repeated tokens (Wang et al., 2026), or validated candidate updates on later observations (Ji et al., 2026). We study a different feedback process: within one inference stream, every retained update changes the model that supplies the next training chunk. Our contributions are: **(1) long-horizon evidence** that this self-generated feedback damages independent real-text prediction across three TTT-E2E scales and under ordinary Adam, together with an external-text exposure boundary for the failure; **(2) a causal decomposition** using Fixed Generation, Recorded Replay, and paired one-update branches to separate changes to future training text, attention, and persistent weights, revealing a state-dependent failure concentrated in a few trajectories. The same feedback intervention improves held-out exact success in WebShop; and **(3) a transfer criterion** showing why source fit and visible repetition cannot by themselves justify retaining an update, and how evaluation on independent text can directly test the complete candidate state.

2 Preliminaries: Persistent Test-Time Training

TTT-E2E as persistent test-time learning. Standard pretraining optimizes a model before any test-time update, whereas TTT evaluates it after such updates. TTT-E2E removes this train–test mismatch: during training, it applies the same next-token updates used at inference and meta-learns an initialization that performs well after adaptation (Tandon et al., 2025). At inference, TTT-E2E updates the MLP blocks in the final quarter of its sliding-window Transformer (Tandon et al., 2025); the underlying Transformer architecture follows Vaswani et al. (2017). Following the fast-weight literature (Schmidhuber, 1992; Ba et al., 2016), we call these adaptable parameters *fast weights*, with initial state W_0 and state W_t after t chunks.

Generated Writes Off control. *Generated Writes Off* (hereafter *Writes Off*) still feeds every generated chunk through attention. The generated tokens can therefore affect the attention state and the model's next prediction, but the parameter update learned from each generated chunk is discarded. Updates learned from the initial real-text prefix are retained in both conditions. Comparing the closed loop with this control changes only whether generated-text writes persist; it does not compare reading generated text with seeing no generated text.

3 Experimental Setup

Models and updates. TTT-E2E pretrains on the DCLM corpus (Li et al., 2024), then extends its long-context models on Books and evaluates on held-out Books (Tandon et al., 2025). We retain this book-domain setting, using PG-19 in place of Books (Rae et al., 2020). We train the 125M and 760M models with the TTT-E2E objective and use the released 3B checkpoint for the large-scale comparison. The main analysis uses the 125M model, with an 8192-token attention window and adaptable feed-forward modules in its final three layers; each 1024-token chunk is followed by one clipped gradient step. We also study Adam updates (Kingma & Ba, 2015) to existing Qwen3-4B weights (Yang et al., 2025). The main cross-scale comparison uses the same six books, five seeds,

128K horizon, and batch width eight at every scale. Appendix A gives the complete configuration map.

Streams and independent evaluation. Each 128K stream begins with 8K real-text tokens, followed by generated chunks and periodic evaluation on separate real passages. We restore the preceding weights and attention after each evaluation, so it affects neither later generation nor learning. Throughout the paper, *independent real text* means human-written evaluation text that is outside the generated training stream and excluded from the online state transition; it need not come from a different corpus. The original schedule contains 105 generated chunks and 15 evaluations; the large-model comparison also evaluates the initial state. K means 1024 tokens. Generation uses temperature one and top-$p = 0.95$ nucleus sampling (Holtzman et al., 2020). Appendix A gives complete schedules and state-restoration checks.

Primary comparison and outcomes. Both conditions start from the same weights and real-text prefix. The *Closed Loop* keeps updates from generated chunks. *Writes Off* discards them. Both keep learning from the real prefix and keep generated tokens in attention. Here c indexes an experimental condition, such as Closed Loop or Writes Off. Let $L_{c,\text{first}}$ and $L_{c,\text{last}}$ denote its NLL on the first and last independent real-text evaluations. Lower NLL is better. We define

$$D_c = L_{c,\text{last}} - L_{c,\text{first}}, \qquad H_c = D_c - D_{\text{off}}. \tag{1}$$

D_c is the condition's first-to-last change: $D_c > 0$ means real-text prediction worsened, while $D_c < 0$ means it improved. Under Writes Off, the update is discarded after *every* generated chunk; generated tokens still enter attention, and learning from the initial real-text prefix is unchanged. D_{off} denotes the resulting first-to-last NLL change under Writes Off. Subtracting it gives H_c, the excess NLL change relative to Writes Off. Thus $H_c > 0$ means worse than Writes Off and $H_c < 0$ means better.

Our primary outcome is H_c. For identical prerecorded tokens, it isolates the effect of retaining updates. During online generation it measures the full feedback policy, including changes to later text. Intervention suites additionally report the final NLL difference from Writes Off, $E_c = L_{c,\text{last}} - L_{\text{off},\text{last}}$. Positive E_c means that damage remains at the end; zero means that the final NLL matches Writes Off. When initial losses match, $E_c = H_c$. Unless specified otherwise, 95% intervals resample books after averaging seeds within each book. Appendix A lists each comparison and its baseline.

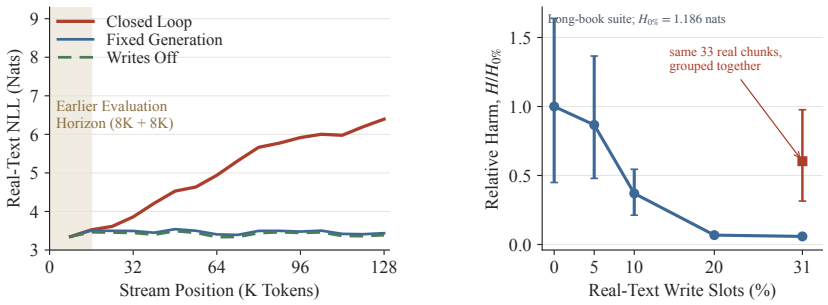
4 Long-Horizon Persistent Self-Writing Degrades Prediction

We begin with the simplest comparison. Both models generate text for 128K tokens and keep that text in attention. The only difference is what happens after a generated chunk: Closed Loop keeps the resulting weight update, whereas Writes Off discards it. We periodically pause the stream and measure both models on the same independent real passages. This design asks whether the write itself damages predictions beyond the effect of reading generated text. To hold evaluation books constant across scales, Table 1 evaluates all three models on the same six books defined in Appendix A. Retaining generated-text writes adds 3.01, 6.00, and .40 nats over Writes Off at 125M, 760M, and 3B, respectively. Every interval excludes zero, establishing the same directional failure at all three evaluated scales. The magnitude differs across checkpoints; the result is a cross-scale replication of direction rather than a monotone relation with parameter count.

Table 1: **Generated-text writes worsen real-text prediction.** The same six books, five seeds, and 128K horizon are used at every scale.

Model	H (↑ worse)	95% Interval
125M TTT-E2E	$+3.01$	$[+1.86, +4.20]$
760M TTT-E2E	$+6.00$	$[+4.28, +7.74]$
3B TTT-E2E	$+0.40$	$[+0.30, +0.51]$

Figure 1a shows when the two policies separate. Inside the shaded 8K+8K horizon, their real-text losses remain close. After many more generated-text writes, the Closed Loop loss rises while Writes Off stays nearly flat. The experiment therefore identifies a long-horizon accumulation effect, not a universal penalty that is already large after the first update. The effect is not an artifact of nucleus truncation. On the same six books, full-support sampling reduces the 128K endpoint from



(a) Prediction Degrades Beyond the Earlier Horizon

(b) Frequent External Text Interrupts the Loop

Figure 1: **Long horizons and sparse real-text exposure expose the failure.** (a) Real-text NLL across a 128K stream; shading marks the earlier 8K+8K horizon (Tandon et al., 2025). (b) Harm relative to a separate long-book suite's 0% baseline. Even spacing interrupts the loop; grouping the same 31% real-text budget into bursts restores much of the harm. Panel (b) is normalized within a separate matched suite (Appendix B.3).

$H = 3.010$ to 1.452 $[.708, 2.180]$, but its loss is still rising by .197 nats per late evaluation. Lower temperature and typical sampling likewise change how soon the curves separate without removing the endpoint gap. A within-chunk repetition penalty reduces H to .879 and the late slope to .0027 nats per evaluation. Periodic reset is a stronger but less selective baseline: resetting fast weights every eight chunks leaves $H = .175$ $[.084, .321]$, but preserves only 44.1% of the real-text adaptation benefit. Resetting every 32 chunks instead leaves $H = 1.922$ $[.949, 3.158]$. The complete comparison appears in Appendix B.2.

Frequent external text interrupts the feedback loop. An independently matched long-book suite replaces 0, 5, 11, 21, or 33 of 105 generated write slots with real text. Figure 1b reports each result relative to this suite's 0% baseline. Evenly spacing 5, 11, 21, and 33 real chunks leaves .866, .370, .068, and .059 of that baseline harm. With the same 33 real chunks grouped into a few bursts, the model can process 12 generated chunks in a row instead of at most three, and .604 of the harm remains. How often real text interrupts self-generation therefore matters in addition to how much real text the stream contains (Appendix B.3).

Beyond native TTT: Qwen3-4B shows the same failure. We use Adam (Kingma & Ba, 2015) to update Qwen3-4B's existing feed-forward weights (Yang et al., 2025), without adding a TTT module. At learning rate 10^{-4}, persistent self-writing raises real-text NLL by 1.231 nats relative to Writes Off; at 10^{-5}, the interval includes zero (Table 2). Yet the same 10^{-4} update lowers NLL by .168 nats when it learns from real text. Ordinary Adam updates can therefore help on real text and fail in a self-generated loop: the failure is not specific to TTT-E2E's learned update mechanism.

Table 2: **Qwen3-4B with Adam.** The same weights benefit from real text but fail under persistent self-writing. Sampling units and complete settings: Appendix C.5.

Adam LR	Persistent self-writing H 95% CI; ↑ worse	Real-text B state range; ↑ better
10^{-5}	.026 $[-.071, .122]$	.139 [.115, .163]
10^{-4}	**1.231** $[\mathbf{.580}, \mathbf{2.350}]$	.168 [.132, .204]

Real-text writes can be useful. Why not disable every update? In a separate *Real-Text Learning* condition on eleven identical teacher-forced streams, keeping updates lowers mean NLL by .0340 nats relative to *No Updates* (Appendix B.4). Because the text is fixed, this gain comes from learning rather than from changing future inputs. Writing itself is not the problem; the next section asks what changes when the learner supplies its own future training text.

5 A Causal Decomposition of Generation, Reading, and Weight Updates

Why does the Closed Loop condition fail? Generated text might be intrinsically poor training data; degraded text might hurt while it remains in attention; or its updates might store additional harm in the weights. Figure 2 provides this causal decomposition. In **Fixed Generation** (a), the initial state W_0 supplies every chunk to a separate learner, so the learner cannot change its future training text. **Recorded Replay** (b) gives a receiver a saved degraded sequence, allowing the same tokens to be read with or without updates. In the **Closed Loop** (c), W_t generates x_t and learning from x_t produces W_{t+1}, which can change the next generated chunk. All three comparisons evaluate the adapted learner on independent real text.

5.1 Keep learning, but fix the generator

We first ask whether synthetic origin is sufficient for harm. In panel (a), the learner still updates after every chunk, but an independently cached W_0 generates all future chunks; evaluation uses the learner.

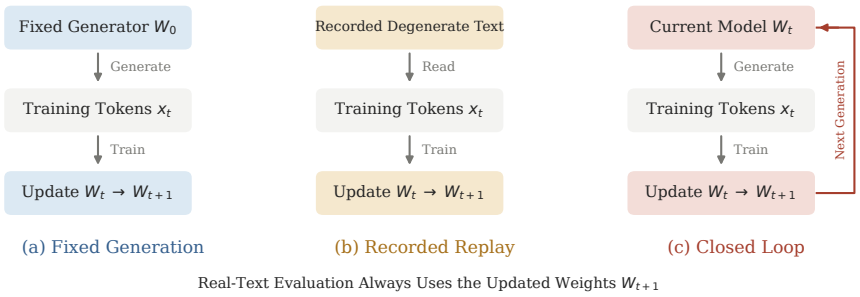


Figure 2: **Three controlled comparisons separate generation, content, and updating.** (a) The initial state W_0 generates while a separate W_t learns, breaking feedback. (b) Saved tokens are processed read-only or with updates, holding content fixed. (c) Current W_t both generates and learns, closing the loop. Every write-enabled condition evaluates the adapted W_t; Appendix A.1 verifies generator isolation.

Figure 3a compares four 128K policies in an audited eight-book, five-seed suite. Relative to Writes Off, Closed Loop raises final real-text NLL by 2.8308 nats $[1.8171, 3.8598]$, whereas Fixed Generation leaves only .0707 $[.0510, .0924]$—a 97.5% reduction. Table 3 confirms the result at 760M. The important point is what remains unchanged: the learner still sees generated text and still updates after every generated chunk. What is removed is only the learner's ability to alter the source of its next update.

Table 3: **Breaking feedback removes most of the excess change at both evaluated scales.** Fixed Generation residuals and removal fractions use matched within-scale comparisons on the same six canonical books. Complete endpoints are in Appendix B.

Model	Fixed Generation H	Closed Loop harm removed
125M	.052	98.3%
760M	.073	98.8%

Figure 3 uses an eight-book mechanism comparison, while Table 3 gives the same six-book comparison at 125M and 760M. The 3B causal confirmation instead holds recorded text fixed in Table 4.

The reduction is not explained by taking smaller steps. Fixed Generation moves farther from the initial weights than Closed Loop (.01387 versus .00857), yet causes far less damage. Real-Text Learning moves farthest (.01759) and improves NLL by .3544 nats $[.1236, .6048]$. Nor is generated origin by itself sufficient: Fixed Generation still uses generated text. These comparisons point specifically to the recurrence in which updated weights determine the text used for later updates. Closed Loop also ends almost entirely repetitive (repeated-4 = .9902), unlike Writes Off (.0217) or Fixed Generation (.0082), but this comparison alone does not tell us whether the degraded text is harmful to another model. Recorded Replay answers that question while holding the tokens fixed.

Fixing the generator also changes the generated text. We therefore next hold the text fixed and measure the additional effect of learning from it.

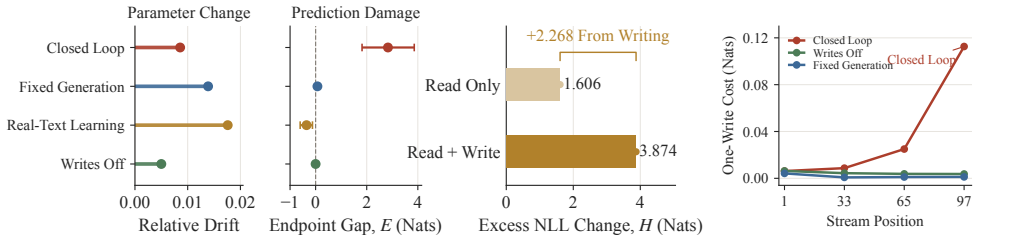
5.2 Recorded Replay: Process the Same Text With and Without Writing

We next hold content fixed. A receiver adapted on different books processes a saved degraded stream that it did not generate. Starting from the same initial state, the receiver first processes the recording without updating. We then restore that state and process the identical tokens while retaining their updates. Because the text is fixed, the difference isolates the effect of writing the recording into the weights.

Table 4: **Degraded content harms through both attention and weights.** The 125M read-only excess shows that the recording is harmful in attention; the paired read-and-write difference measures the additional persistent cost.

Scale	Receiver operation on the same recorded text	Real-text NLL above control
125M	Read only	1.606
125M	Read and write	3.874
125M	**Passage-level write cost**	**2.268**
3B	**Passage-level write cost**	**.193** [**.104**, **.327**]

At 125M, reading alone adds 1.606 nats over the receiver's control, so the degraded content is already harmful while it remains in attention. Reading and writing adds 3.874 nats; the extra 2.268 nats is the cost stored by the weight updates. A prespecified 3B replication measures the same write-minus-read contrast over 120 paired trajectories spanning six source and six disjoint receiver books. Writing adds .1927 [.1043, .3272] nats, and every source- and receiver-book aggregate is positive (Appendix C.2).



(a) Drift Does Not Rank Damage (b) Same Text, Different Writes (c) Feedback Amplifies One-Write Cost

Figure 3: **Feedback, not parameter drift, amplifies the cost of writing.** (a) Audited 125M control: the same four policies are ordered across relative drift and final clean-text NLL; horizontal bars are 95% book intervals. (b) Identical recorded tokens with updates disabled or retained; only this panel holds text fixed. Both panels use eight books and five seeds. (c) In a separate paired one-update comparison, the real-text cost of retaining one update rises sharply over a Closed Loop history, but stays near zero when earlier generated writes were discarded or came from Fixed Generation.

Together, the causal decomposition identifies two distinct paths. First, feedback changes which text the model will later learn from: freezing the generator removes most damage. Second, once a degraded passage exists, it affects prediction in two ways: reading it changes the active context, and learning from it stores an additional persistent cost. The write effect is confirmed at both 125M and 3B.

The same feedback effect appears in WebShop. We repeat the comparison with persistent LoRA on Qwen3-4B. Fixed Generation keeps the same update rule but uses a frozen actor to select later trajectories. It raises held-out exact success from .1053 to .1800 in all five paired seeds; task-metric Settlement reaches .1853. Thus the feedback path changes task completion as well as text prediction (Figure 4c; Appendix G). We next isolate one update.

5.3 One update can fit its source and hurt another input

To isolate one update, we branch from the same weights, attention cache, and training chunk, then keep the update in one branch and discard it in the other. Table 5 reports *keep minus skip* for source-chunk NLL, next-real-text NLL, and later repetition.

The three rows describe the state before this paired update. *Writes Off History* means all earlier generated-text writes were discarded; *Closed Loop* means they were retained; and *Fixed Generation* means earlier updates were learned from chunks generated by frozen W_0. Negative training-chunk NLL means that keeping the update fits its source better; positive next-real-text NLL means that the same update harms the new passage.

Table 5: **An update can improve source fit while harming new text.** 125M; keep minus discard, with 95% book intervals.

Preceding History	Training-Chunk NLL (↓ better fit)	Next Real-Text NLL (↑ worse)	Future Repeated-4 (↑ more repetition)
Writes Off History	$-.1374$ $[-.1453, -.1285]$	$+.0045$ $[.0037, .0053]$	$+.0049$ $[-.0079, .0248]$
Closed Loop	$-.0354$ $[-.0384, -.0317]$	$+.0381$ $[.0098, .0718]$	$+.0086$ $[.0020, .0184]$
Fixed Generation	$-.1449$ $[-.1482, -.1407]$	$+.0018$ $[.0014, .0023]$	$+.0017$ $[-.0034, .0063]$

In every row, keeping the update lowers loss on its generated chunk but raises loss on the next real passage: source optimization does not ensure transfer. The magnitudes are not monotone either. Fixed Generation gives the largest source-fit gain (0.1449) and smallest real-text cost (0.0018), while Closed Loop history gives a smaller gain (0.0354) and larger cost (0.0381). Source fit therefore cannot rank updates by transfer. Within Closed Loop history, the average cost also rises from 0.006 nats at the earliest measured position to 0.113 at the latest, showing that the preceding state changes the effect of the next update.

Table 6: **Two local diagnostics predict measured one-update transfer cost.** (a) More-negative gradient cosine indicates greater conflict and harm. (b) The first-order estimate $g_{\text{real}}^{\top}\Delta W$ increases with measured harm. Spearman ρ; 96 paired updates per model-history setting.

(a) Gradient cosine

History	125M ρ	760M ρ
Closed Loop	$-.502$ $[-.642, -.305]$	$-.633$ $[-.808, -.383]$
Writes Off	$-.788$ $[-.826, -.720]$	$-.731$ $[-.810, -.666]$

(b) First-order prediction

History	125M ρ	760M ρ
Closed Loop	$+.704$	$+.632$
Writes Off	$+.794$	$+.751$

Gradient conflict identifies a local signature of poor transfer. Can we identify this conflict before applying the update? At the same state, g_{self} is the generated-chunk gradient and g_{real} is the gradient of the next real passage. A negative cosine means that reducing the first loss tends to increase the second. Across four positions, eight books, and three seeds, a more negative cosine predicts a larger one-update transfer cost at both 125M and 760M (Table 6a). The first-order quantity $g_{\text{real}}^{\top}\Delta W$ gives the same ranking in panel (b). Thus gradient conflict provides a local explanation of failed transfer. Because it requires an independent real-text gradient, it is a causal signature rather than an update-selection rule (Appendix C.4).

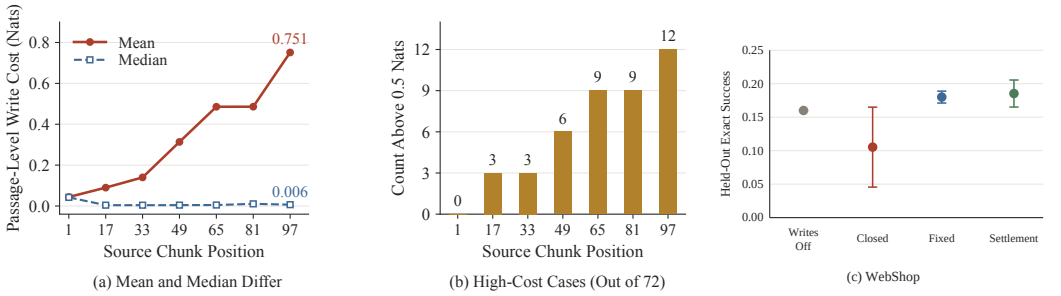
What the causal decomposition establishes. Generated text is not automatically harmful; feedback changes future content; degraded content affects prediction through attention; and retaining its update can store additional harm. Gradient conflict gives a local signature of this failed transfer. We next ask whether visible repetition can identify writes to avoid.

6 Which Updates Should Be Retained?

6.1 Damage amplifies in degraded states but remains heterogeneous

Table 5 shows that retaining one generated-text update costs more after Closed Loop adaptation. We ask whether every late passage becomes harmful or whether a few passages cause large failures.

We extract eight-chunk passages from seven positions in 24 recorded sequences. From the same receiver state, each passage is processed read-only and again with its updates retained; the subsequent



(a) Mean and Median Differ (b) High-Cost Cases (Out of 72) (c) WebShop

Figure 4: **Feedback produces concentrated text damage and lower agent success.** (a) Mean and median passage-level write cost; (b) cases above 0.5 nats, with 72 fixed-receiver comparisons per position. (c) WebShop held-out exact success for Writes Off, Closed Loop, Fixed Generation, and task-metric Settlement over the same five seeds.

real-text NLL difference is its passage-level write cost. Each position has 72 paired comparisons across three receiver configurations. Figure 4 reports the mean, median, and count above 0.5 nats. If passage-level write cost rose uniformly, both mean and median would rise; rare failures would raise only the mean.

Mean passage-level write cost rises from 0.0439 to 0.7512 nats, while the median falls from 0.0422 to 0.0064. Thus the average rises because a minority of passages become extremely harmful, not because every late passage becomes moderately harmful. The number above 0.5 nats rises from 0 to 12 of 72, and all such cases come from only four of the 24 recorded source sequences. Moreover, once a source-receiver pair crosses this threshold, all its later sampled passages remain above it (0 recoveries in 504 comparisons). Most passages therefore stay near zero while a few trajectories enter a persistently damaging state. Repetition does not reliably identify them: repetition bins give an unresolved early-to-late change of $+.0587$ $[-.0382, .3286]$ (Appendix D).

A natural repair is to learn less from repeated output, but the heterogeneous cost distribution above explains why this is incomplete. Repetition-aware weighting lowers endpoint gap from 2.89 to 1.56 nats, and read-only real context lowers it to 1.40 nats; neither reaches Writes Off. Uniform dose reduction traces the same trade-off; an eight-chunk reset removes 94.8% of damage but retains 44.1% of real-text adaptation. Full comparisons appear in Appendices B.2 and D.2. This motivates a direct question: does an update help beyond its source text?

6.2 External validation measures whether an update transfers

Neither repetition nor source fit directly measures whether an update helps another input. We therefore evaluate that effect before keeping the update. Let W be the current fast weights, δ the proposed change, and q an independent real-text passage. Holding attention fixed, we compare

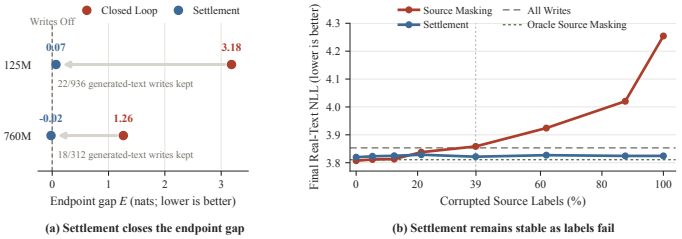
$$A(\delta; W, q) = L(q; W) - L(q; W + \delta). \tag{2}$$

Positive A means the candidate improves prediction of q. *Settlement* keeps the candidate temporary until this comparison is available, committing it only when $A \geq 0$. Candidates are checked sequentially against the state containing earlier accepted updates. This matters: separately approved changes can harm prediction when added together (Appendix E.5). Settlement uses the next arriving real passage, or retained real text when no new real input arrives. Neither evaluation changes attention, and separate passages measure the final prediction outcome. *The rule does not use the candidate's source label*, but it still requires identifiable real text for validation. Algorithm 1 in Appendix E.1 specifies the sequential validation and commitment rule.

Settlement reduces E from 3.18 to .07 nats at 125M and from 1.26 to $-.02$ at 760M (Figure 5a). This selective commitment is not reproduced by simply weakening every write: quarter-sized updates, read-only real text, and both together leave .90, 1.04, and .28 nats without validation; with Settlement, every variant finishes within .04 nats of Writes Off (Appendix E.3). On a matched all-real stream, Settlement accepts 89/113 candidates and gives a .0534-nat benefit $[.0299, .0721]$, versus .0450 for Write All; it is not a reject-all policy (Appendix E.4).

When is external validation useful beyond Source Masking? Source Masking keeps real-text writes and rejects writes labeled as generated; Settlement instead evaluates the proposed state. Their

predictions are close with correct labels (3.8072 and 3.8196 NLL). As labels are corrupted, Settlement remains stable and becomes better beyond 38.9% error (Figure 5b). Full results appear in Appendices F–E.5.



(a) Settlement closes the endpoint gap

(b) Settlement remains stable as labels fail

Figure 5: **Independent evidence evaluates transfer rather than source.** (a) Settlement nearly closes the gap to Writes Off. (b) Source Masking deteriorates as labels are corrupted, whereas Settlement remains stable. Lower is better; full protocols and intervals are in Appendices E–F.

7 Related Work

Writable inference-time state. Fast weights provide writable state in attention and linear sequence models (Schmidhuber, 1992; Ba et al., 2016; Schlag et al., 2021); TTT layers, TTT-E2E, Titans, and In-Place TTT extend it to language models (Sun et al., 2025; Tandon et al., 2025; Behrouz et al., 2025; Feng et al., 2026). We study what happens when this state also changes the text supplying its next update.

Test-time adaptation and stability. Test-time adaptation learns from deployment inputs or retrieved neighbors (Sun et al., 2020; Liu et al., 2021; Wang et al., 2021; Hardt & Sun, 2024); continual methods use averaging, restoration, or reset (Wang et al., 2022; Niu et al., 2023; Press et al., 2023). Self-writing is performative because updates change later examples (Perdomo et al., 2020).

Generated-data feedback and update selection. Recursive synthetic-data training can reduce quality or diversity, while real data can slow collapse (Briesch et al., 2023; Shumailov et al., 2024; Alemohammad et al., 2024; Marchi et al., 2024; Gerstgrasser et al., 2024). Unlike this across-model setting, we separate generation, attention, and updates within one stream. Prior methods constrain gradients or reweight examples (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Ren et al., 2018); aTTT downweights repetition (Wang et al., 2026), while VANE evaluates later observations (Ji et al., 2026). We instead evaluate whether the accumulated state transfers beyond its update text.

8 Discussion and Conclusion

What causes the failure? Frozen generation is stable, and real-text updates improve prediction, so generated text alone is not harmful. Damage appears when adapted weights generate chunks that supply later updates. This changes model state and future training data, and four of 24 sequences produce every loss above 0.5 nats.

How does the causal decomposition establish this explanation? Fixed Generation keeps all updates but stops them changing later training chunks, removing 97.5% of the 125M damage. Recorded Replay fixes the tokens: degraded text hurts through attention, and learning from it stores additional harm. Paired single-update branches show that source fit can hurt the next real passage; gradient conflict predicts the largest costs. Fixed Generation also improves WebShop exact success. The failure is poor transfer inside the feedback loop, not weight movement or repetition alone.

What reduces the damage? Frequent real text interrupts the loop: with the same 31% allocation, even spacing leaves .0694 nats of excess damage whereas bursts leave .7155. Settlement evaluates the proposed state before commitment and preserves real-text adaptation; under corrupted labels it is more stable than Source Masking. For agents, validation should use success or reward; an inner-loop update should persist only when the combined state preserves that capability.

AI USE STATEMENT

In this work, we used generative AI tools to refine hypotheses and experimental plans, provide feedback on research methodology, assist with the interpretation of experimental results, and revise the manuscript's structure and presentation. We additionally used these tools to edit prose, LaTeX, tables, and scientific figures, and to assist with literature search and reference formatting. Generative AI was not used to generate experimental data or to make final decisions about scientific claims. The authors ran the experiments, checked the reported results against the underlying experimental records, reviewed all AI-assisted text and artifacts, and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Appendix A reports the model configurations, checkpoint and data hashes, book identifiers, random seeds, update schedules, evaluation procedures, and statistical units used in the experiments. The remaining appendices provide the complete controlled-comparison definitions, additional results, and pseudocode for the validation procedure. Code and configuration files will be made publicly available.

REFERENCES

Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard G Baraniuk. Self-consuming generative models go MAD. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2025.

Martin Briesch, Dominik Sobania, and Franz Rothlauf. Large language models suffer from their own output: An analysis of the self-consuming training loop. *arXiv preprint arXiv:2311.16822*, 2023.

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

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

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, et al. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. *arXiv preprint arXiv:2404.01413*, 2024.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2020.

Hongjin Ji, Guoyang Xia, Luoyang Sun, Fangxiang Feng, and Lei Ren. VANE: Reliable test-time training for vision-language-action models via future visual representation prediction. *arXiv preprint arXiv:2608.09448*, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Jeffrey Li et al. Datacomp-lm: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, volume 37, 2024.

Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When does self-supervised test-time training fail or thrive? In *Advances in Neural Information Processing Systems*, volume 34, pp. 21808–21820, 2021.

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

Matteo Marchi, Stefano Soatto, Pratik Chaudhari, and Paulo Tabuada. Heat death of generative models in closed-loop learning. In *2024 IEEE 63rd Conference on Decision and Control*, pp. 1524–1530, 2024. doi: 10.1109/CDC56724.2024.10886816.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations (ICLR)*, 2023.

Juan Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative prediction. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 7599–7609, 2020.

Ori Press, Steffen Schneider, Matthias Kümmerer, and Matthias Bethge. RDumb: A simple approach that questions our progress in continual test-time adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning (ICML)*, 2018.

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

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

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

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9229–9248, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57503–57522, 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, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations (ICLR)*, 2021.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Yanbo Wang, Jinhua Hao, Yuze Shi, Kun Yuan, and Ming Sun. No time like the present: Agentic test-time training for llm agents. *arXiv preprint arXiv:2607.03441*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

A Experimental scope and implementation

The experiments answer three questions: whether persistent self-writing causes long-horizon damage, which part of the feedback loop causes it, and whether independent evidence can screen a proposed update. Every conclusion comes from a paired treatment and control that share the model, books, seeds, horizon, and evaluation schedule. Unless stated otherwise, language-model intervals are 95% bootstrap intervals over books after averaging seeds within each book.

Stable book identities. The cross-scale, Fixed Generation, and decoder experiments use the same six PG-19 documents, denoted *canonical books 2–7*, with five seeds and a 128K horizon. Exposure and validation require longer uninterrupted documents, so each uses one fixed long-book set shared by every policy in that comparison. The manifests record document hashes, selection thresholds, and scripts.

Protocol map. Table 7 maps each paper claim to its shared configuration and controlled comparison. All harm measures use Writes Off as the reference. H compares first-to-last changes; E compares final endpoints and is used for intervention policies. They coincide when the initial losses match.

Table 7: Main experiment configurations and why they differ. Each row compares policies under one shared configuration; books are the independent units for language-model intervals.

Question	Shared configuration	Controlled comparison
Long horizon	125M, 760M, 3B; canonical six books; five seeds; 128K; width 8	Closed Loop versus Writes Off
Feedback path	125M and 760M; canonical six books; five seeds; 128K; Fixed Generation added	Closed Loop versus Fixed Generation
Read vs. write	Fixed recorded tokens; 125M: eight receiver books, five seeds; 3B: 120 paired trajectories	Read Only versus Read + Write
Single update	Eight books and four stream positions; identical state and source text in each branch	Keep versus discard one update
Decoder/exposure	Decoder: canonical setup; exposure: one eight-book long-stream set, five seeds	Decoder pairs; evenly spaced versus bursty real text
Validation	Scale-specific long-book sets; each scale has its own paired Writes Off trajectory	Closed Loop versus Settlement
Agents	Qwen3-4B LoRA; five paired seeds; fixed held-out WebShop evaluation	Closed Loop, Fixed Generation, Settlement, Writes Off

Model and training. Our 125M TTT-E2E reproduction uses a 12-layer transformer, an 8192-token sliding attention window in the first nine layers, and a SwiGLU fast-weight MLP in each of the final three layers. Each sequence has its own fast weights. One clipped inner-loop SGD step follows each 1024-token chunk. Unless a decoder control is specified, temperature is one and top-p is .95. Training follows the official 4800-step recipe on 2.5B DCLM tokens (Li et al., 2024), followed by the 32K length-extension schedule on PG-19 (Rae et al., 2020) in place of Books. The "125M" name follows the model configuration; the converted model contains 184,363,776 parameters in

total, including its TTT parameterization. The experiment manifests record complete SHA-256 digests for the 125M checkpoint and the PG-19 validation array.

Conversion and numerical checks. The PyTorch and official-format models have identical parameter counts. Conversion changes the sequence loss by 3.6×10^{-5} over eight inner-loop chunks; every chunk loss differs by less than 10^{-4}. A rematerialized inner-loop backward agrees with a naive second-order implementation to 6×10^{-16} in fp64. For the released 3B/128K checkpoint, the converted training forward changes sequence loss by 1.05×10^{-4} and gives token-level NLL correlation 0.9996. This verifies the converted training forward, not exact parity of the 3B streaming decoder with the official decoder.

Streaming evaluation. The standard 128K schedule starts with eight real-text chunks, followed by generated chunks and periodic real-text evaluations. The original schedule has 15 branch evaluations and 105 generated chunks; the new cross-scale suite reports 16 NLL values including its initial reference. Evaluation never updates weights. They snapshot and restore the complete carried state, so evaluation text does not become later generation context. The state contains fast weights, attention caches, positions, held updates, and partial-chunk buffers. We distinguish a condition's first-to-last NLL change D_c from the paired excess $H_c = D_c - D_{\text{off}}$. When initial losses are matched exactly, this equals the final-evaluation difference. Other tables explicitly use the endpoint gap $E_c = L_{c,\text{last}} - L_{\text{off},\text{last}}$.

Write strength, retention, and state restoration. The audited write-strength multiplier scales the clipped update, not the loss before clipping. For $w \in \{1/8, 1/4, 1/2, 1, 2, 4\}$, a single-step norm obeys $\|\Delta W(w)\|/(w\|\Delta W(1)\|) = 1$ to a worst relative error of 1.15×10^{-7}; $w = 0$ gives exactly zero update. Loss scaling before a saturated norm clip would not implement the same controlled operation. Over 12 chunks, net drift is not linear in w because update directions partly cancel. Retention applies $W \leftarrow W_0 + \lambda(W - W_0)$ once per chunk, including chunks whose proposals are rejected. With no accepted updates, six chunks at $\lambda = .9$ give exactly the expected displacement factor $.531441$. Snapshot–restore returns all 39 carried tensors bitwise identical and four scalars equal. A teacher-forced 12-chunk regression check with interleaved evaluations matches the losses obtained without evaluation exactly. These checks do not establish bitwise reproducibility of all stochastic generation trajectories.

A.1 Generator isolation and chunk boundaries

The audited Fixed Generation control uses a separate cache advanced only by the frozen generator. On two books, perturbing receiver weights, KV, positions, and buffers leaves this generator unchanged. Keyed per-row sampling is invariant to batch reordering, and a writes-off null comparison is bitwise identical.

The canonical implementation starts each generated chunk with a real boundary token, and every paired policy uses the same rule. Teacher-forcing recorded tokens is not bitwise equivalent to online decoding; the audit finds maximum logit difference $.1875$ and mean KL $.000287$.

A.2 Feedback and displacement control

This suite uses the 125M checkpoint, eight receiver books, five seeds, 128 chunks, and the canonical boundary rule. All four policies use the same code path. Real-text evaluations snapshot and restore state. The analysis first averages seeds within book, then uses a 20,000-draw paired bootstrap over the eight books. CUDA sampling and the disabled token-level entropy observer are recorded in the experiment manifest.

Paired endpoint contrasts are: Closed Loop minus Writes Off 2.8308 $[1.8171, 3.8598]$; Fixed Generation minus Writes Off $.0707$ $[.0510, .0924]$; Real-Text Learning minus Writes Off $-.3544$ $[-.6048, -.1236]$; and Closed Loop minus Fixed Generation 2.7601 $[1.7555, 3.7881]$. Fixed Generation and Real-Text Learning move farther from initialization than Closed Loop while causing far less harm. Parameter displacement is therefore not a sufficient explanation of the closed-loop loss. The independent Fixed Generation suite in Appendix A.1 remains a replication under a separate protocol.

Table 8: **Audited 125M feedback and displacement control.** Drift is $\|W_t - W_0\|/\|W_0\|$. Real-Text Learning contains no generated tail, so token-diversity statistics are not applicable.

Policy	Final clean NLL	Final relative drift	Tail Distinct-2	Tail Repeated-4
Closed Loop	6.5084	.00857	.0084	.9902
Writes Off	3.6776	.00500	.8493	.0217
Fixed Generation	3.7483	.01387	.8779	.0082
Real-Text Learning	3.3232	.01759	—	—

A.3 Checkpoint-specific contamination screening

The first two candidate books have anomalously low loss under the larger released checkpoints. The audit compares unadapted first-chunk loss against the 125M reproduction. The larger ratios on books 0–1 are consistent with checkpoint-specific familiarity, but cannot establish the exact training membership of any book. Table 9 defines the ratio explicitly. The screened 3B set uses canonical books 2–7. Screening removes a visible anomaly; it does not prove absence of all contamination. The 760M and 3B training lineages also differ, so their damage magnitudes do not establish a scaling law.

Table 9: Unadapted loss ratio $L_{125M}/L_{\text{checkpoint}}$ on the same book. A large ratio means unusually low loss under the larger checkpoint.

Checkpoint	0	1	2	3	4	5	6	7
760M, DCLM	3.69	1.44	1.15	1.23	1.18	1.19	1.14	1.18
3B, DCLM	10.35	4.12	1.30	1.33	1.29	1.32	1.30	1.28
3B, Books finetune	8.83	2.67	1.44	1.69	1.44	1.41	1.37	1.46

B Replication and boundaries of the failure regime

This appendix follows Section 4. It first confirms Fixed Generation at a second model scale, then asks which properties of the stream change the severity of the failure. The evidence gives two clear boundaries: decoder choice changes both when harm becomes visible and how fast it grows, whereas regular real-text writes sharply suppress it.

B.1 Fixed Generation confirmation at 125M and 760M

The three-condition suite compares Closed Loop, Writes Off, and Fixed Generation. Table 10 evaluates both scales on the same six canonical books with five seeds per book. Initial evaluations agree across policies within each scale, so the endpoint contrasts are directly comparable within a row and use the same first-to-last definition as Table 1.

Table 10: Fixed Generation confirmation on the same six receiver books. Intervals resample books after averaging seeds.

Model	Condition	Initial NLL	Final NLL	Final excess over Writes Off [95% CI]
125M	Writes Off	3.3469	3.3845	0
	Fixed Generation	3.3469	3.4361	.0516 [.0248,.0772]
	Closed Loop	3.3469	6.3942	3.0097 [1.8575,4.1972]
760M	Writes Off	2.7913	2.9383	0
	Fixed Generation	2.7913	3.0116	.0733 [.0538,.0939]
	Closed Loop	2.7913	8.9405	6.0021 [4.2695,7.7347]

At both scales, fixing the generator leaves only a small fraction of the Closed Loop endpoint gap. The independent 3B confirmation in Appendix C.2 instead holds the received text fixed and measures the additional cost of retaining its updates.

B.2 Decoder and reset controls change the rate of damage

Six decoder policies and two reset schedules use the same 125M checkpoint, canonical books 2–7, five seeds, and a 128K schedule with 105 generated chunks. All rows use the same book-blocked estimator as Table 1. The repetition penalty is applied to tokens already generated within the current chunk; typical-$p = .95$ replaces rather than composes with top-p sampling. A reset restores fast weights to their initial values every K chunks but leaves the attention cache intact.

Table 11: Canonical decoder and reset controls. Endpoint H uses the common Writes Off reference; late slope is the mean change per evaluation over the final four intervals. Lower is better in both columns.

Control	Endpoint H	Late slope (nats/evaluation)
top-$p = .95$	3.0097	+.1339
top-$p = .99$	3.3531	+.1497
typical-$p = .95$	2.9292	+.0948
full support	1.4520	+.1979
$T = .7$, top-$p = .95$	2.2246	+.1704
repetition penalty 1.1	**.8794**	**+.0027**
reset every 8 chunks	**.1749**	**+.0056**
reset every 32 chunks	1.9218	+.0623

Endpoint and late slope answer different questions. Full-support sampling has a smaller 128K endpoint than the default decoder, but its late slope is the largest in the table: it delays the visible failure without flattening the curve. Fixed Generation, reset every eight chunks, and the repetition penalty all have late slopes below .006; the remaining continuing controls have late slopes above .062. Reset frequency matters sharply. Resetting every eight chunks leaves $H = .1749$ $[.084, .321]$ (5.2% of Closed Loop damage), whereas resetting every 32 leaves $H = 1.9218$ $[.949, 3.158]$ (56.8%). Thus a short reset window is an effective safety baseline, but it also limits how long adaptation can persist.

We measure that trade-off on real-text streams. Ordinary persistent writing improves NLL by .0850 $[.0444, .1260]$ over Writes Off. Resetting every eight chunks preserves a .0375 $[.0195, .0526]$ benefit, or 44.1% of the persistent adaptation gain. Reset therefore controls runaway accumulation by shortening the lifetime of every update, including useful ones. The repetition penalty also changes the entire future stream; neither control decides whether a particular fixed passage should be written.

For intuition, consider an idealized SGD step at a fixed context. Sampling from $q_W(x) \propto p_W(x)^{1/T}$ on fixed support A and applying $\eta \nabla_W \log p_W(x)$ gives expected increment $\eta T \nabla_W \log \sum_{x \in A} p_W(x)^{1/T}$. At $T = 1$ with full support, this mean is zero; truncation instead increases retained probability mass to first order. The derivation assumes an infinitesimal unclipped step and a fixed context and therefore does not predict a zero finite-horizon effect. The paired experiment above supplies the relevant empirical result: truncation amplifies damage but is not required for it.

B.3 External-text exposure bounds the failure regime

We vary the number and placement of real-text write slots while keeping the 125M checkpoint, the dedicated long-book set $[0, 1, 27, 28, 29, 43, 44, 46]$, five seeds, 8K real prefix, 105 live slots, temperature one, and top-$p = .95$ fixed. *All Writes* retains updates from both real and generated slots. *Writes Off* retains real-text updates but discards generated-text writes while still reading those tokens. Thus H is the excess NLL change caused by retaining generated-text writes at each exposure density. The exposure sweep needs longer uninterrupted documents than the canonical comparison, so every exposure policy instead uses the same eight long books. The main-text panel reports $H/H_{0\%}$ within this set; the table below gives the corresponding absolute values.

Increasing evenly spaced real-text exposure reduces excess NLL change while preserving a positive real-text adaptation benefit. Density is not the whole story: at 31%, grouping the same 33 real slots together increases the longest uninterrupted stretch of self-generated chunks from 3 to 12 and increases H by an order of magnitude. The controlled quantity is therefore real-text cadence: both how much real text arrives and the longest interval without it.

Table 12: Real-text cadence controls excess NLL change. H is the excess NLL change of retaining generated writes; B is the benefit of retaining real-text writes. Intervals use a paired book bootstrap after averaging seeds.

Real fraction	Schedule	Real slots	Max self burst	H [95% CI]	B [95% CI]
0%	Even	0	105	$1.1858; [.5326, 1.9430]$	$.0000; [.0000, .0000]$
5%	Even	5	17	$1.0275; [.5679, 1.6195]$	$.1350; [.0557, .2578]$
10%	Even	11	8	$.4389; [.2517, .6465]$	$.0933; [.0729, .1155]$
20%	Even	21	4	$.0806; [.0657, .0981]$	$.0802; [.0585, .1052]$
31%	Even	33	3	$.0694; [.0534, .0843]$	$.1051; [.0668, .1431]$
31%	Bursty	33	12	$.7155; [.3730, 1.1578]$	$.1040; [.0502, .1527]$

B.4 Teacher-forced real-text payoff

The matched 125M payoff experiment uses eleven fixed teacher-forced real-text books. Benefit is the loss under *No Updates* minus the loss under the evaluated writing policy; positive values mean that adaptation helps. Ordinary sequential writing gives $.0340$ nats of benefit. Because the text is fixed, the improvement cannot come from changing future training tokens. The same suite evaluates whether validation preserves this gain, including the failure of individually validated but jointly committed updates; the full comparison appears in Table 26.

C Causal decomposition: feedback, attention, and persistent weights

This appendix follows the causal order of Section 5. We first separate the generator from the learner, then replay identical text with and without updates, and finally branch before a single update. Each step removes one source of variation, moving from the full online loop to a paired local comparison.

C.1 Fixed Generation and Recorded Replay

The primary audited 125M generator control is reported in Appendix A.2: Closed Loop minus Writes Off is 2.8308 $[1.8171, 3.8598]$, while Fixed Generation minus Writes Off is $.0707$ $[.0510, .0924]$. The matched 760M result is reported in Appendix B. These controls show that continuing to update on generated text need not reproduce the large closed-loop loss. They change subsequent text as well as its dependence on adapting weights, so they do not hold content fixed.

The Recorded Replay comparison instead supplies the same recorded tokens to the receiver with writing enabled or disabled. Relative to the suite's Writes Off control, Recorded Replay without writes gives 1.606 nats of excess and replay with writes gives 3.874, an additional 2.268 nats. This paired comparison isolates the contribution of writing that replayed text. Comparing live closed generation with replay does not isolate an additional feedback effect, because they contain different text. The prespecified 3B experiment below is the large-model confirmation.

C.2 Prespecified 3B fixed-text replay confirmation

This experiment evaluates whether retaining updates from a fixed recorded stream adds cost beyond reading exactly the same tokens. Six source books are disjoint from six receiver books. Two prespecified cyclic source–receiver mappings and five independently sampled recordings produce 120 paired read/write trajectories. For each pair,

$$J = (L_{\text{write,last}} - L_{\text{write,first}}) - (L_{\text{read,last}} - L_{\text{read,first}}).$$

The paired trajectories have exactly equal initial NLL and identical received token hashes. A 20,000-draw two-way source/receiver cluster bootstrap gives $J = 0.1927$ $[0.1043, 0.3272]$; clustering only by source gives 0.1927 $[0.1181, 0.2941]$. Thus the conclusion does not depend on treating the 120 dependent trajectories as independent samples.

Table 13: **The 3B passage-level write cost is positive across every source and receiver aggregate.** The statistical interval is clustered by source and receiver, not by individual trajectory.

Source book	Mean J	Leave-one-out J
12	.1862	.1940
13	.1953	.1922
14	.0617	.2189
15	.1331	.2046
16	.1602	.1992
17	.4196	.1473

Receiver book	Mean J
2	.2872
3	.1824
4	.1302
5	.1042
6	.1343
7	.3178

C.3 Paired one-update comparison

The 125M experiment branches at generated positions 1, 33, 65, and 97 from Closed Loop, Writes Off, or Fixed Generation histories. The branches share the starting state and current text; one retains that text's update and the other skips it. Four draws of four future chunks use paired random numbers. All 15 history trajectories and 60 paired configurations completed, giving 480 book-level observations. We average seeds and positions within each book before a 20,000-draw paired book bootstrap. Each row in Table 14 is a paired keep-minus-discard comparison within its starting states; differences between rows also change the text and history, and do not isolate receiver susceptibility alone.

Table 14: Keep minus discard for one update: negative source NLL improves fitting; positive real NLL harms independent prediction. Intervals are 95% book-bootstrap intervals.

History	Δ source NLL	Δ real NLL	Δ future repeated-4
Closed Loop	$-.0354$ $[-.0384, -.0317]$	.0381 [.0098, .0718]	.0086 [.0020, .0184]
Writes Off	$-.1374$ $[-.1453, -.1285]$	.0045 [.0037, .0053]	.0049 $[-.0079, .0248]$
Fixed Generation	$-.1449$ $[-.1482, -.1407]$	.0018 [.0014, .0023]	.0017 $[-.0034, .0063]$

Table 15: Mean one-update real-text NLL effects by position, 125M. Positions differ in both current text and receiver state.

History	Position 1	33	65	97
Closed Loop	.0062	.0087	.0250	.1126
Writes Off	.0063	.0044	.0038	.0037
Fixed Generation	.0042	.0008	.0011	.0012

The paired comparison uses matched random numbers and measures update norms from the applied parameter increment. Reimplementation checks reproduce the reported aggregate outcomes; the analysis concerns paired outcome differences rather than bitwise execution traces.

C.4 Gradient conflict and one-update transfer cost

We evaluate whether the geometry of a candidate generated-text update predicts its measured one-update transfer cost. From the same 125M and 760M pilot snapshots, we form a generated-chunk gradient g_{self}, a gradient g_{real} on the next independent real passage, and a second real-text gradient as a reference for attainable real–real agreement. Both gradients use the same deferred clipping and normalization path as the candidate update. For each model and history, eight books, three seeds, and four positions give 96 branches. We report Spearman correlations because the one-update transfer costs are heavy-tailed; intervals resample books after averaging seed/position branches.

All eight correlation intervals exclude zero. The generated and real-text gradients are nearly orthogonal in writes-off histories (mean cosine $-.005$ and $-.004$) and more opposing after closed-loop history ($-.093$ and $-.103$). In contrast, two real-text gradients have positive mean cosines ($.125/+.481$ at 125M and $.097/+.481$ at 760M, writes-off/closed), so the signal is distinguishable from a general lack of gradient agreement. The alignment term is therefore a local diagnostic of transfer direction. It explains the paired one-update cost; Settlement separately measures the consequence of committing the complete accumulated state.

Table 16: Gradient conflict predicts the measured cost of retaining one generated-text update. ΔL_{real} is keep minus discard on the next real passage; negative cosine means the source and real-text gradients oppose one another. Negative ΔL_{self} confirms that each update improves its source chunk.

Model / history	$\rho(\cos, \Delta L_{\text{real}})$	$\rho(g_{\text{real}}^{\top}\Delta W, \Delta L_{\text{real}})$	ΔL_{self}	Mean cos
125M / Closed Loop	$-.502$ $[-.642, -.305]$	.704 [.521, .829]	$-.033$	$-.093$
125M / Writes Off	$-.788$ $[-.826, -.720]$	.794 [.721, .838]	$-.136$	$-.005$
760M / Closed Loop	$-.633$ $[-.808, -.383]$	.632 [.449, .781]	$-.051$	$-.103$
760M / Writes Off	$-.731$ $[-.810, -.666]$	.751 [.685, .828]	$-.166$	$-.004$

C.5 In-place Adam: dose and history are separate considerations

Qwen3-4B freezes all parameters except the final four `mlp.down_proj` matrices and takes one Adam step per chunk, with gradient norm clipped at one. It uses no outer meta-learning or added modules. Eight books, three seeds, batches of four, eight real warm-up chunks, and 128 total chunks supply Table 17. The latest 8192 tokens are re-encoded each chunk; naive KV truncation raised evaluation loss from about 3.04 to 8.85, while rebuilding restored it to 3.02. This audit motivates the reported window handling. The comparison isolates ordinary Adam updates to existing model parameters rather than an added TTT module.

Table 17: In-place Adam long-horizon control. The larger-rate excess is 1.231 [.580,2.350]; the smaller-rate excess is .026 [-.071,.122].

Condition	NLL change D	Relative drift	Distinct-2	Repeated-4
Writes Off	-.060	.00000	.140	.831
Adam 10^{-5}	-.034	.00627	.187	.775
Adam 10^{-4}	1.171	.02399	.008	.991

To verify that the larger learning rate is not simply unusable, we also adapt the same parameter subset on teacher-forced real text. The future tokens are fixed and therefore cannot be changed by the update. Positive $B = L_{\text{no update,last}} - L_{\text{real update,last}}$ means that writing improves prediction on independent real text.

Table 18: **The Qwen3-4B update configuration also learns useful real text.** Two independently adapted states each contain four books; the bracket is their range, not a book-level confidence interval.

Adam LR	B [range of two state means]	State 0	State 1
10^{-5}	.1392 [.1150, .1634]	.1634	.1150
10^{-4}	.1680 [.1317, .2043]	.2043	.1317

Both learning rates improve independent real-text prediction in both adapted states. Together with Table 17, this shows that the 10^{-4} configuration can help on real text while harming prediction when its future training stream is self-generated.

D From heterogeneous harm to partial repairs

Section 6 shows that late damage is heavy-tailed rather than uniform. This appendix first gives the underlying distribution, then evaluates three natural responses: weighting repeated tokens less, shrinking every write, and inserting real text without writing it. Each response reduces harm, but none directly determines whether a particular update transfers.

D.1 A minority of source passages drives the late mean

The fixed-receiver sweep replays eight-chunk passages from 24 source rows into three receiver offsets at seven stream positions. For each passage, write damage is subsequent real-text NLL after read-and-write minus NLL after read-only processing from the same receiver state. Receiver offsets show whether a source effect transfers across books; they do not create additional independent source trajectories.

Table 19: Fixed-receiver stage replay. The increasing mean is produced by a growing right tail; the median remains near zero.

Source stage	Median	Mean	Standard deviation	Cases above .5 nats
1	.0422	.0439	.025	0/72
17	.0040	.0898	.385	3/72
33	.0039	.1399	.620	3/72
49	.0042	.3135	1.009	6/72
65	.0047	.4855	1.274	9/72
81	.0105	.4858	1.267	9/72
97	.0064	.7512	1.731	12/72

All cases above .5 nats come from four of the 24 source rows. The same rows produce large effects at all three receiver offsets. A separate repetition-bin comparison gives an unresolved early-to-late change of $.0587$ $[-.0382, .3286]$. Stream position and repetition therefore indicate risk but do not identify every harmful passage.

D.2 Repetition weighting mitigates but does not remove damage

The expanded policy suite uses sixteen books and five seeds. Random and uniform controls reduce update weight without using token identity; the comparison asks whether repetition-aware weighting helps beyond simply learning less.

Table 20: Repetition weighting and real context reduce the endpoint gap but do not reach Writes Off. Lower E is better.

Policy	Final NLL	E [95% interval]	Distinct-2
Writes Off	3.5767	0	.8425
Closed Loop	6.4620	2.8853 [2.0072,3.8310]	.0082
Repetition weighting	5.1404	1.5637 [1.1858,1.9825]	.0469
Random dose	5.6325	2.0558 [1.6339,2.4946]	.0086
Uniform dose	6.0628	2.4861 [1.9510,3.0939]	.0155
Read-only real context	4.9746	1.3979 [1.0988,1.7140]	.0730

Repetition weighting improves on random and uniform dose assignment by $.4921$ and $.9224$ nats, respectively, while leaving 1.5637 nats above Writes Off. The result supports repetition as a useful symptom, not a sufficient admission rule. Because these are online policies, later generated text also differs.

D.3 Smaller writes expose a stability–adaptation trade-off

We scale each post-clipping parameter increment by $\alpha \in \{0, 1/16, 1/8, 1/4, 1/2, 1, 2\}$. The same scale is applied to teacher-forced real-text writes. Generated-stream harm H and real-text benefit B are both measured relative to $\alpha = 0$.

Table 21: Uniformly smaller writes reduce both harm and useful adaptation. Every evaluated nonzero dose has positive values for both quantities.

α	Excess NLL change H [95% CI]	Real-text benefit B [95% CI]
0	.0000; [.0000, .0000]	.0000; [.0000, .0000]
$1/16$	.3856; [.1737, .6432]	.0412; [.0249, .0579]
$1/8$	.8583; [.5480, 1.2276]	.0543; [.0328, .0758]
$1/4$	.9248; [.6525, 1.3374]	.0682; [.0413, .0942]
$1/2$	1.0319; [.6424, 1.4540]	.0812; [.0507, .1106]
1	1.1867; [.8512, 1.5056]	.0922; [.0579, .1248]
2	1.9374; [1.0941, 2.9775]	.1009; [.0626, .1373]

Reducing α delays and reduces damage, but it also reduces the payoff on real text. Uniform scaling therefore moves along a stability–adaptation frontier rather than separating useful and harmful writes.

D.4 Real read-only context helps because of its content

Every eighth generation slot is replaced by a read-only slot, leaving 92 generated-text writes in every condition. The model receives no update in that slot. A matched no-op condition skips the same slots, so the comparison separates the content read in attention from the pause in generation.

Table 22: Read-only content at a matched write schedule. Negative differences improve prediction relative to skipping the slot.

Read-only content	Final NLL	Difference from no-op [95% CI]
No-op	6.4042	0
Self-generated	6.7214	.3172 [-.4775,1.1963]
Random tokens	7.3969	.9927 [-.2699,2.0992]
Shuffled real	5.5680	-.8362 [-1.7643,.0324]
Fixed Generation text	5.7912	-.6130 [-1.5127,.2200]
Real text	4.8349	-1.5693 [-2.3011,-.8433]

Real text lowers final NLL by 1.5693 nats relative to the no-op schedule and outperforms random, shuffled-real, and Fixed Generation anchors in direct paired comparisons. Thus real text can interrupt the loop through attention without being stored in weights. The policy still ends 1.3979 nats above Writes Off (Table 20), so it is a partial repair rather than a complete selection rule.

E Testing transfer with independent evidence

The mechanism experiments show that source fit does not certify transfer. This appendix turns that observation into a controlled question: if a candidate state predicts independent real text better, can it be retained without recreating the closed loop? We define the sequential rule, report its evidence requirements and scale results, and then separate validation from weaker updates and better context. The hybrid policy evaluates a pending candidate on arriving real text before learning from that text, and uses retained real text when no new external text arrives. Each candidate is compared with the currently accepted state using the same validation context. Accepted candidates update that reference before the next candidate is evaluated. The decision does not inspect the candidate's source label, but selecting trusted real validation text does require knowing which evidence is real. Evaluation passages are separate from admission text. The policy uses external validation; it does not claim that validation itself is a new learning principle.

E.1 Sequential validation and commitment

Algorithm 1 describes the decision rule for an ordered list of pending fast-weight increments. Each increment was computed from its source chunk before the validation passage became eligible; the passage must not be the text used to produce that increment. In the hybrid policy, new external text is preferred and retained real text supplies a fallback. Outcome-evaluation passages are not used to choose updates.

ReadOnlyNLL restores the same attention context, positions, buffers, and other carried state for both evaluations and leaves the live stream unchanged. It does not train on q. Only accepted increments change the committed fast weights; ordinary reading of arriving input occurs outside this procedure. The next increment is evaluated against the accumulated accepted state, not the original W. Candidate increments are not jointly summed after separate approval against an unchanged reference.

For Quarter Dose, the proposed increment is one quarter of the clipped generated-text update before it enters P. The validation comparison thus judges the same increment that would be committed. This differs from scaling the loss before clipping, accepting a full update and scaling it afterwards, or decaying previously accepted weights. Retention is not part of this ablation's validation rule.

Evidence scheduling and pending-buffer capacity are implementation choices; the reported experiments use the schedules stated with each comparison.

Algorithm 1 Settlement processes pending fast-weight updates sequentially.

Require: Accepted fast weights W; ordered pending increments P; eligible new real text q_{new}; retained real-text bank B; fixed validation context C

$q \leftarrow q_{\text{new}}$ if available; otherwise eligible text from B
if no eligible q is available **then**
  return (W, P) ▷ Wait; buffer capacity is configuration-specific
end if
$S \leftarrow W$
for each δ in P, in proposal order **do**
  $\ell_0 \leftarrow \text{READONLYNLL}(q, S, C)$
  $\ell_1 \leftarrow \text{READONLYNLL}(q, S + \delta, C)$
  if $\ell_1 \leq \ell_0$ **then**
    $S \leftarrow S + \delta$ ▷ Next candidate is judged against accepted changes
  end if
  Remove this candidate from P; record losses and decision
end for
Commit $W \leftarrow S$
return (W, P)

Table 23: Completed Settlement outcomes and later admission-log extraction. The two scales use separate matched suites and Writes Off trajectories.

Model / books	Closed Loop minus Writes Off	Settlement minus Writes Off	Generated admitted
125M / 12	3.183 [2.106,4.253]	.067 [-.013,.169]	22/936
760M / 4	1.263 [.799,1.618]	-.021 [-.058,.014]	18/312

E.2 Cross-scale outcomes

At 125M, Settlement improves on Closed Loop on 11 of 12 books; at 760M, it improves all four books. The residual intervals include zero but are not equivalence bounds and do not establish superiority to Writes Off. Real-text acceptance is 83% and 100% in the respective log summaries.

E.3 Component ablation

This four-book, three-seed comparison separates external validation from two weaker controls: quarter-sized updates and periodic read-only real context. Table 24 reports final NLL above Writes Off. Without validation, each control leaves a visible gap. With Settlement, every combination finishes within .04 nats of Writes Off. Adding read-only real context to Settlement lowers NLL by a further .057 nats $[.024, .089]$; the incremental effect of Quarter Dose is not resolved.

Table 24: Completed component ablation, 125M, four books and three seeds. Rounded interval endpoints at zero are not equivalence statements.

Without validation	Excess [95% interval]	With Settlement	Excess [95% interval]
Closed Loop	3.06 [.67,7.02]	Settlement	.04 [.00,.09]
Quarter Dose	.90 [.32,1.97]	+ Quarter Dose	.02 [-.01,.04]
Read-Only Real Context	1.04 [.19,2.59]	+ Real Context	-.02 [-.03,.00]
Both controls	.28 [.13,.53]	+ Both controls	.00 [-.01,.02]

E.4 Admission and adaptation in one all-real stream

A rule can match Writes Off by rejecting every proposal. We therefore measure admission and adaptation together on an all-real stream. The three policies process the same six long-stream books with three seeds. *Reject All* supplies the no-write baseline, *Write All* retains every real-text proposal, and Settlement applies the same sequential rule used above. Because the first evaluation occurs after adaptation has begun, benefit is the policy's first-to-last NLL improvement relative to Reject All.

Settlement admits 78.8% of the real-text proposals. Its paired benefit over Write All is .0083 $[.0020, .0161]$. The low admission rate on self-generated streams is therefore a property of those candidate updates, not a fixed tendency to reject all learning.

Table 25: Settlement admits most real-text candidates and preserves their adaptation benefit in the same matched experiment. Positive benefit is better; intervals resample books after averaging seeds.

Policy	Real admitted	Benefit [95% interval]
Reject All	0/113	0
Write All	113/113	.0450 [.0235, .0609]
Settlement	**89/113**	**.0534** [**.0299**, **.0721**]

E.5 Preserving adaptation and validating the committed state

The scheduled gates below alternate admission and evaluation passages so the same passage does not both authorize and evaluate an update. Harm and benefit in Table 26 come from separate streams, not a joint experiment such as Table 25. Harm uses four books (five seeds for ordinary/scheduled gates; three for Settlement). Real-text benefit uses eleven teacher-forced books and *No Updates* as its baseline.

Table 26: Validate the state that will be committed. Joint commit checks each candidate at W but deploys their unevaluated sum; sequential policies judge the accumulated state S_i.

Policy	Candidate judged against	State committed	H (↓)	B (↑)
Ordinary writing	—	$W + \delta_t$	$+1.56$	$+.034$
Joint commit	W (each δ)	$W + \sum_i \delta_i$	≈ 0	-1.093
Sequential	S_{i-1}	S_i	$+.10$	$+.019$
Settlement	S_{i-1} on next q	S_i	≈ 0	$+.037$

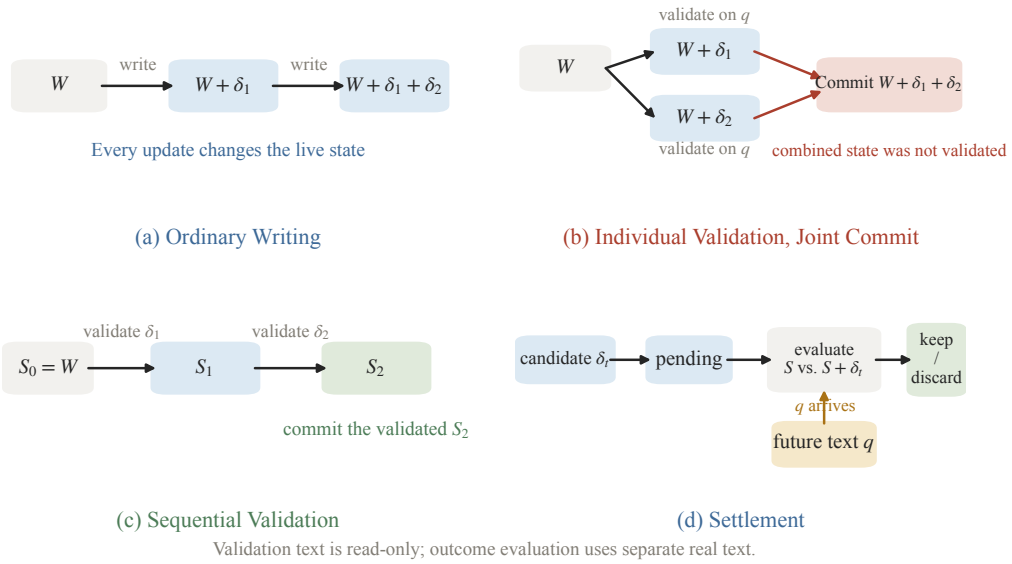


Figure 6: **Four ways to apply persistent updates.** (a) Ordinary writing immediately changes the live state. (b) Joint commit validates candidates separately at W but deploys their unevaluated sum. (c) Sequential validation judges each candidate against the accumulated accepted state. (d) Settlement keeps a candidate pending until arriving external text q can evaluate the state that would be committed.

Table 26 reports the numerical comparison. Positive benefit improves on No Updates; missing intervals are not inferred from other rows. The individual gate accepts 113/113 real candidates computed against a shared base, then commits their combination. This committed state matches unconditional deferred commitment and has terminal drift .0862, versus .0176 for ordinary sequential writing. Sequential validation evaluates each next increment against the accumulated accepted state, retains about 56% of the ordinary real-text benefit, and avoids that large loss. This identifies a failure of this joint application; it does not rule out alternative batch normalization, increment recomputation, or learning-rate corrections. Settlement accepts 98–101 of 112 real candidates. Its point benefit is close to ordinary writing. Table 25 provides the matched admission–benefit comparison.

F EXTERNAL VALIDATION WHEN SOURCE LABELS ARE CORRUPTED

Source Masking uses a real/generated label to decide whether a write should be retained. Settlement uses the observed labels only to locate real validation text; admission is determined by the candidate state's validation advantage A. We keep the underlying 125M mixed stream fixed at 31% real-text slots and corrupt an increasing fraction of the observed labels. Each condition uses three seeds. All Writes retains every update, and Oracle Source Masking uses the uncorrupted labels.

Table 27: Corrupting source labels harms Source Masking but has little effect on Settlement. Lower final NLL is better. The middle columns show the incorrect decisions made by Source Masking.

Observed error	Generated kept	Real discarded	Source Masking NLL	Settlement NLL
.000	0	0	3.8072	3.8196
.053	4	2	3.8116	3.8230
.124	9	5	3.8129	3.8248
.212	15	9	3.8377	3.8286
.389	26	18	3.8584	**3.8215**
.619	41	29	3.9242	**3.8270**
.876	64	35	4.0204	**3.8243**
1.000	73	40	4.2544	**3.8244**

With accurate labels, Source Masking has the lower endpoint NLL. As errors increase, it retains generated writes and discards real writes; its NLL rises from 3.8072 to 4.2544. Settlement stays between 3.8196 and 3.8286 and becomes better once the realized error rate reaches 38.9%. The paired Settlement-minus- Masking intervals exclude zero at error rates .389, .619, .876, and 1.000.

The stable result is not caused by a constant admission policy. At 100% observed-label error, regrouping Settlement decisions by true source gives 120/120 real candidates admitted and 0/219 self-generated candidates admitted; regrouping the same decisions by the corrupted labels reverses these counts. The validation advantage A therefore continues to separate the candidate states even when the labels used to locate validation text are wrong.

G WEBSHOP FEEDBACK AND TASK-METRIC VALIDATION

The main experiments evaluate independent real-text prediction. WebShop asks whether the same feedback path also changes task completion. The experiment uses persistent rank-16 LoRA updates to Qwen3-4B rather than TTT-E2E. *Writes Off* uses the initial model and retains no action updates. *Closed Loop* retains updates, so the adapted model chooses the actions that form later training episodes. *Fixed Generation* retains the same updates, but a frozen copy W_0 supplies every action for the learner. *Settlement* retains a proposed state only when it improves task reward. All held-out evaluations use the learner with writing disabled.

The base model is frozen, and LoRA is applied to `down_proj` in the final quarter of layers (about 1.8M trainable parameters). Each written episode takes two Adam steps at 10^{-5} on at most 16 prompt–action pairs. Each policy processes 900 episodes and is evaluated on the same 150 held-out goals (Yao et al., 2022). Closed Loop, Fixed Generation, and Settlement use the same five seeds; Writes Off is the same deterministic reference in each comparison.

Table 28: WebShop outcomes across five paired seeds. Writes Off is the same deterministic reference in every comparison.

Policy	Reward	Exact success	Retained
Writes Off	.3309	.1600	0
Closed Loop	.4356	$.1053 \pm .0597$	180/180
Fixed Generation	.3724	$.1800 \pm .0089$	180/180
Settlement	.3823	$\mathbf{.1853 \pm .0202}$	10/180

Fixed Generation raises exact success from $.1053 \pm .0597$ to $.1800 \pm .0089$ and beats Closed Loop in every paired seed. Settlement evaluates a candidate every 25 training episodes on 10 disjoint validation episodes. It accepts 10 of 180 candidates and reaches exact success $.1853 \pm .0202$, again beating Closed Loop in every paired seed. The feedback path therefore changes task behavior as well as language-model loss, and validation is most informative when it measures the capability to be preserved.