Can Test-Time Training Remember What It Writes? Diagnosing Sequence-Internal Forgetting in Parameter-State Memory

Anonymous ACL submission

Abstract

Test-time training (TTT) can store long-context information by updating a parameter state during inference, avoiding the need to keep every past token in an explicit KV cache. This compact memory raises a stability question that standard long-context benchmarks do not isolate: after information has been written into the parameter state, does it remain recoverable after later tokens update the same state? We study this question as *sequence-internal forgetting*. To separate post-write degradation from initial encoding failure or cache-based retrieval, we introduce a shadow-stream evaluation framework: at selected anchors, we copy the current parameter state into a temporary stream with an empty KV cache and no further updates, then probe earlier content from that copied state alone. Across probes at multiple granularities, early content becomes progressively harder to recover from later TTT-style parameter states, whereas a Freeze Weights control shows no comparable degradation. A controlled write-scaling intervention changes this trajectory, indicating that continued online writes contribute to the degradation. Because forgetting can also be mitigated by suppressing new writes, we further introduce an anchor-grid protocol that evaluates parameter-state stability jointly with plasticity. Together, these results identify parameter-state stability as a distinct evaluation axis for TTT-style models. [1]

1 Introduction

Long-context language modeling is often framed as a question of how much history a model can store and retrieve. Full attention keeps history explicitly in a KV cache, but this storage grows with context length. This cost has motivated efficient attention (Vaswani et al., 2017; Dao et al., 2022; Liu et al., 2024; Grattafiori et al., 2024) and compressed-memory alternatives that replace token storage with recurrent, state-space, linear-attention, or parameterized states (Rae et al., 2019; Katharopoulos et al., 2020; Sun et al., 2023; Gu and Dao, 2023). Test-time training (TTT) provides a related but distinct form of compressed memory: during inference, the model performs online self-supervised updates, and the resulting parameter state carries contextual information (Sun et al., 2020, 2024; Tandon et al., 2025).

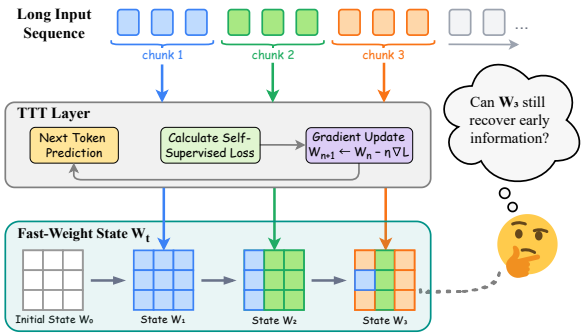

Figure 1: Parameter-state memory in TTT-style models. Online updates write new context into a parameter state, but each update also modifies the same state that stores earlier context. We ask whether information recoverable from an early state such as W_1 remains recoverable from later states such as W_3.

Unlike a KV cache, a parameter-state memory remains writable throughout the sequence: after early content is written into the state, later tokens update the same state to encode new information. These later writes may perturb parameter directions that previously supported recovery of early content. Figure 1 illustrates this question. Thus, we identify *sequence-internal forgetting*: **information written into an early TTT parameter state can become less recoverable after subsequent online writes to the same state.**

This failure mode requires a diagnostic separate from ordinary long-context accuracy. When a compressed-memory model cannot answer a ques-

[1] Our code is available at https://anonymous.4open.science/r/Sequence-Internal-Forgetting-9E83.

tion about earlier context, the information may never have been written into the finite state with sufficient fidelity, or it may have been written and later degraded by subsequent updates. End-task accuracy usually collapses these two bottlenecks into a single metric.

Recent work partially mitigates the first issue by routing hard-to-compress tokens to an auxiliary exact-attention residual cache (Swamynathan, 2026). Since the second issue has received less systematic study, we focus on parameter-state memory stability. The goal of this paper is diagnostic: to measure this failure mode and determine whether it appears in current TTT-style models.

We isolate sequence-internal forgetting with a *shadow-stream evaluation framework*. At a probing anchor, we copy the main stream's parameter state into a temporary stream, reset the KV cache, disable further TTT updates, and evaluate probes drawn from earlier regions of the sequence. Because the shadow stream has no cached access to the original context and cannot adapt to the probe, successful prediction must rely on the copied parameter state. Under this protocol, Vanilla TTT shows rising early-probe loss as later updates accumulate, while a Freeze Weights control remains flat. The pattern appears for token-level, answer-level, and entity-centric probes, and recurs in various TTT-style models.

We further test whether the trajectory is sensitive to the write rule using a controlled *write-scaling intervention*, which scales later writes along historically active parameter directions. Changing later writes changes early-content recoverability, indicating that sequence-internal forgetting is tied to write dynamics rather than being merely an artifact of the evaluation; the intervention is used as a mechanism test, not as a new memory architecture.

A useful parameter-state memory should not only preserve old information, but also continue absorbing new information. Thus, reducing forgetting alone is not sufficient: overly suppressing writes may improve stability while damaging plasticity. We therefore introduce an *anchor-grid protocol* that measures both properties within the same online trajectory, defining stability as whether earlier information remains recoverable and plasticity as whether newly observed information can still be written. Our contributions are as follows:

- We identify **sequence-internal forgetting** as a stability failure mode of TTT-style parameter-state memory.
- We introduce **shadow-stream evaluation** to isolate post-write recoverability from cache access and probe-induced adaptation.
- We use a controlled **write-scaling intervention** as a mechanism test and show that later write dynamics contribute to the degradation.
- We introduce an **anchor-grid protocol** for evaluating stability and plasticity within one online trajectory.

2 Parameter-State Memory and Sequence-Internal Forgetting

2.1 TTT as Online Parameter-State Memory

TTT-style models turn inference into an online adaptation process. Given a test sequence, the pretrained parameters θ remain fixed while a parameter state W_t is updated by a self-supervised loss on each token or chunk (Sun et al., 2020, 2024; Schlag et al., 2021):

$$W_t = W_{t-1} - \eta \nabla_W \ell_t(W_{t-1}; \theta).$$

The updated state is then used for subsequent inputs, serving both as an adaptation variable and as a carrier of sequence information.

2.2 The Stability Problem

Parameter-state memory differs from explicit token storage because reading and writing are coupled through the same mutable state. A KV cache can be read without changing its stored tokens, whereas a TTT parameter state remains writable throughout the sequence. Later tokens can update coordinates that helped encode earlier content, creating a write-interference risk: information recoverable from W_t may become less recoverable from W_{t+k}.

We refer to this degradation as **sequence-internal forgetting**. A model exhibits sequence-internal forgetting when content from an early region of a test sequence becomes harder to predict from later parameter states after cache-based access to the original context has been removed. Unlike catastrophic forgetting across tasks, the interference occurs within one inference sequence as online writes accumulate.

2.3 Why Standard Long-Context Scores Are Not Enough

End-task failures on long-context retrieval do not by themselves identify parameter-state instability.

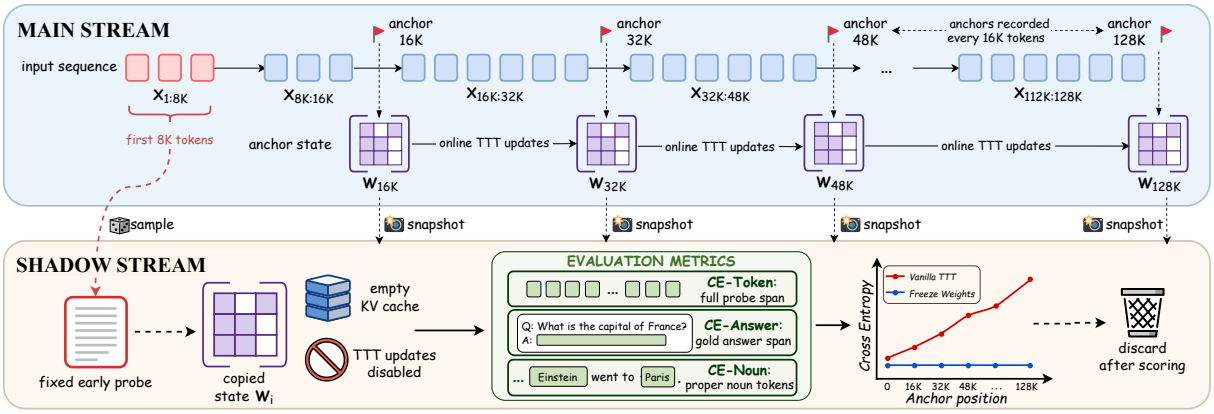

Figure 2: Shadow-stream evaluation isolates parameter-state memory. The main stream processes the sequence normally and performs TTT updates. At a probing anchor, we copy the current parameter state into a temporary shadow stream, reset the KV cache, disable further TTT updates, and evaluate probes drawn from earlier sequence regions. Because the shadow stream cannot access the main stream's cached context, successful probe prediction must come from the copied parameter state. The shadow stream is discarded after scoring, leaving the main trajectory unchanged.

A compressed-memory model may fail because target information was never written with sufficient fidelity, or because it was written and later degraded. These explanations point to different remedies: better capacity or compression in the first case, and more stable writing in the second.

Our research focuses on the second failure mode: whether information recoverable from the current parameter state remains recoverable after subsequent writes. Standard long-context evaluations measure end-to-end task accuracy, which conflates initial encoding failures with post-write degradation. To objectively evaluate the internal dynamics of TTT architectures, we must first isolate the parameter state.

3 Diagnosing Sequence-Internal Forgetting

We first test whether sequence-internal forgetting can be measured separately from other long-context failure modes.

3.1 Shadow-Stream Evaluation

Shadow-stream evaluation probes a copied parameter state in a temporary inference stream. Figure 2 illustrates the protocol. The main stream processes a sequence normally; at selected anchors, we copy its parameter state into a temporary stream with an empty KV cache and no further TTT updates. The shadow stream evaluates probes from earlier regions and is subsequently discarded, leaving the main trajectory intact.

This construction removes two confounds that would otherwise obscure the interpretation. The fresh cache prevents recovery from cached activations, and disabling updates prevents the probe from creating a new adaptation trajectory. The measured loss reflects what the copied parameter state already makes recoverable. If the same early probe becomes harder to predict at later anchors, holding the probe itself fixed, the degradation indicates loss of recoverability from the evolving parameter state.

3.2 Multi-Granularity Probes

We evaluate recoverability at three granularities, always sampling probes from earlier sequence regions and re-scoring them at later anchors with the shadow-stream evaluation framework. Lower cross-entropy indicates better retention.

CE-Token. CE-Token measures average negative log-likelihood on an early token span:

$$\mathrm{CE}_{\text{token}} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{1}{|T_d|} \sum_{t \in T_d} -\log P(t \mid c_{<t}). \tag{1}$$

where $\mathcal{D}$ is the set of probe sequences, T_d is the set of tokens in probe d, and $c_{<t}$ is the context preceding token t during shadow-stream evaluation. CE-Token gives a token-level view of retention.

CE-Answer. CE-Answer asks whether an answer remains predictable when the model is cued by its associated question:

$$\mathrm{CE}_{\text{answer}} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{|A_q|} \sum_{t \in A_q} - \log P(a_{q,t} \mid p_q,\, a_{q,<t}) \,. \quad (2)$$

Here, $\mathcal{Q}$ denotes question-answer probes, A_q the answer-token positions, a_q the gold answer, p_q the question-side prompt, and $a_{q,<t}$ the answer prefix. CE-Answer targets answer-level retention while normalizing for answer length.

CE-Noun. CE-Noun focuses on entity-centric tokens that anchor specific facts in the earlier context:

$$\mathrm{CE}_{\text{noun}} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{1}{|N_d|} \sum_{n \in N_d} - \log P(n \mid c_{<n}). \quad (3)$$

where N_d is the set of proper noun tokens in probe d, identified by a PoS tagger (Li et al., 2025). Proper nouns often carry factual or entity-specific information, making CE-Noun a focused recall probe.

The three probes should be interpreted as complementary rather than redundant. CE-Token provides a coarse, distributional measure of early-text recoverability and is therefore not a pure factual-forgetting metric. CE-Answer and CE-Noun are finer-grained: the former tests whether queried knowledge remains usable for answer generation, while the latter targets entity-level factual cues that are closer to changed or unchanged factual knowledge. Together, the three metrics offer a more complete and balanced quantification of sequence-internal forgetting.

3.3 Diagnostic Setup

The diagnostic setup holds each early probe fixed while later online writes accumulate. CE-Token uses Project Gutenberg, CE-Answer uses TiC-StackExchange, and CE-Noun uses TiC-WIKI with part-of-speech-tagged proper nouns (Li et al., 2025).

Our main experiments use TTT-E2E-3B. For each 128K-token sequence, we sample a fixed early probe from the first 8K tokens and re-score it at anchors from 16K to 128K in 16K increments. Each metric is averaged over 300 sequences. To localize the role of continued writes, we compare Vanilla TTT with a **Freeze Weights control** that freezes the state after 16K and disables later updates. The first scoring anchor is therefore placed after the early probe has had an opportunity to enter the parameter state. Holding the probe and sequence fixed across anchors makes the resulting curve a within-sequence retention trajectory rather than a comparison across different examples.

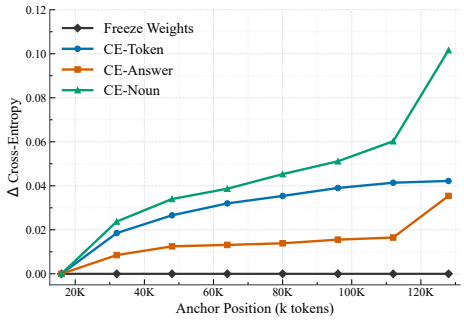

Figure 3: Aggregated forgetting trajectories across three probe granularities at TTT-E2E-3B. The plot shows the cross-entropy increase relative to the first anchor. CE-Token, CE-Answer, and CE-Noun all degrade as later online updates accumulate, whereas the Freeze Weights control remains flat.

3.4 Evidence from TTT-E2E

Figure 3 shows a consistent stability pattern: under Vanilla TTT, the same early probes become harder to recover as more tokens are processed. The effect appears for token spans, answer spans, and proper-noun tokens, so it is not tied to one probe construction.

The Freeze Weights control localizes the source of the effect. Once later online updates are removed, early-probe loss stays flat. The Vanilla rise is not explained by later anchors, longer prefixes, or the shadow-stream procedure; under this control, it is attributable to continued online updates to the parameter state. The TTT-style parameter state can carry earlier information, but continued updates make that information less recoverable.

3.5 Evidence Across TTT Variants

The TTT-E2E result could be checkpoint-specific, so we repeat the diagnostic on TTT-Linear-1.3B, TTT-MLP-1.3B, TTT-Linear-760M, and TTT-MLP-760M (Sun et al., 2024; Tandon et al., 2025). We use the released checkpoints and recommended hyperparameters, and evaluate using the same datasets and multi-granularity probes as the main diagnostic. The variants differ along two axes that matter for parameter-state memory: scale and the form of the online-updated TTT layer. TTT-Linear updates a single linear transformation, whereas TTT-MLP updates a two-layer MLP, test-

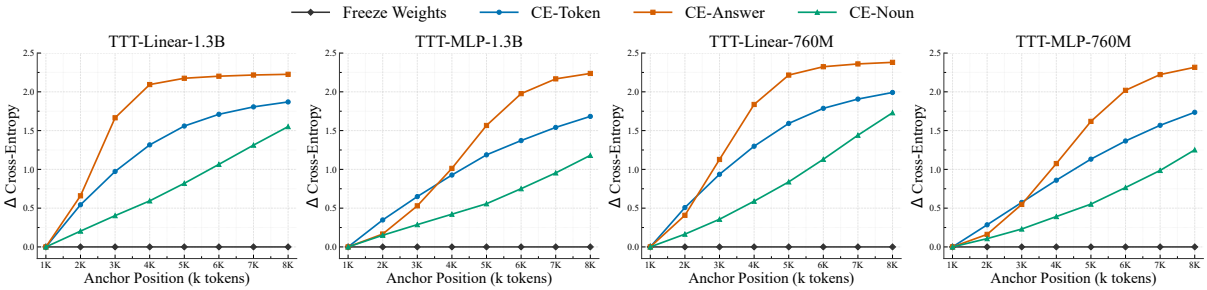

Figure 4: Cross-variant forgetting trajectories. The plots show the cross-entropy increase relative to the first anchor for four additional TTT variants. Consistent with the primary model, early-probe loss for all three metrics progressively degrades as later online updates accumulate, confirming that this instability generalizes across different architectures and scales.

ing whether degradation depends on the updated state form.

We follow the same shadow-stream design as Section 3.1. Adapting to the context limits of these released checkpoints, we evaluate all four models on 8K-token contexts, sample probes from the first 1K tokens, and score them at anchors from 1K to 8K. Because these settings follow different released hyperparameters and context limits, the experiment is designed as qualitative robustness evidence rather than an architecture ranking.

Figure 4 shows the same qualitative pattern. Across the evaluated variants and probe types, early-probe loss increases as later anchors accumulate more online updates, while the Freeze Weights control remains flat. Sequence-internal forgetting is therefore not specific to a single checkpoint; it recurs across the evaluated TTT-style family.

4 Testing Write Interference

Having localized the effect to continued writes, we test whether changing how later updates enter the parameter state changes the forgetting trajectory.

4.1 A Controlled Write-Scaling Intervention

We use a **write-scaling intervention** to probe write dynamics. The write-scaling intervention changes how later updates are written into the parameter state by scaling optimizer-produced updates according to historical update activity; it is not proposed as a new memory architecture.

Motivated by classic work in continual learning, we design the write-scaling rule to use historical update activity as a proxy for identifying parameter directions that are likely to carry previously written information (Zenke et al., 2017). At each step, it constructs a gated activity signal,

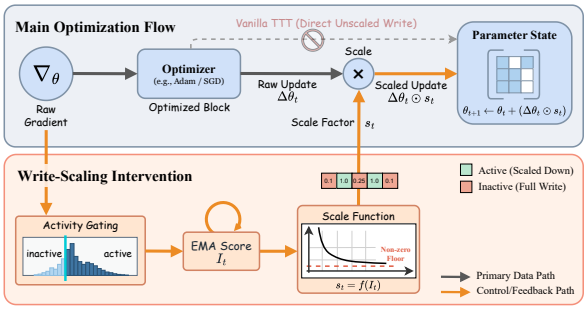

Figure 5: Controlled write-scaling intervention. The write-scaling intervention leaves the model, probes, anchors, and optimizer state fixed, but rescales optimizer-produced updates before they are written into the parameter state.

accumulates the signal into an exponential moving average (EMA) importance score, and converts the previous-step importance into a scale on the next optimizer-produced update. Historically active coordinates receive smaller future writes, while a nonzero floor prevents hard freezing.

The importance signal is deliberately tensor-local. The rule normalizes gradients within each tensor, forms an activity score, and retains only activity above a within-tensor quantile gate. The gate prevents diffuse low-level activity from protecting most coordinates and focuses the importance estimate on unusually active directions within each tensor.

As shown in Figure 5, the write-scaling intervention acts directly on the write into the parameter state while preserving the optimizer's internal dynamics. It rescales the optimizer-produced update rather than the raw gradient, making it compatible with optimizer state such as momentum while requiring no replay buffer, auxiliary memory module, extra backward pass, or second-order statistics. Appendix A gives the full tensor-normalization,

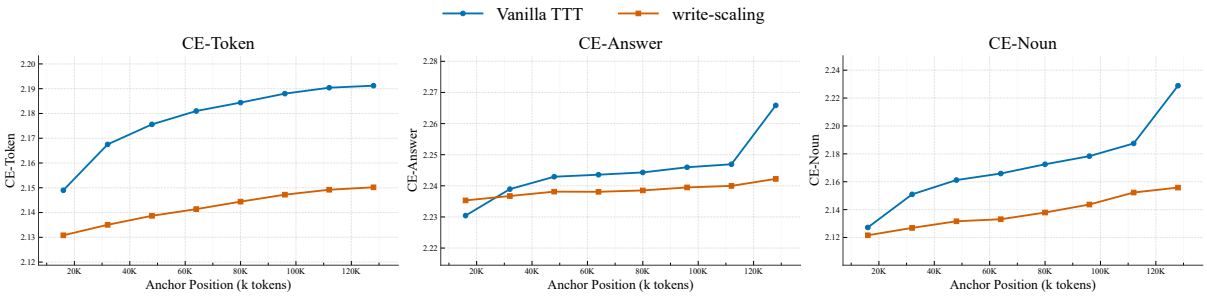

Figure 6: Effect of the write-scaling intervention. The plots show the cross-entropy at each anchor. The default model exhibits continuous degradation across all three metrics as sequence distance increases, whereas the write-scaled variant significantly flattens these trajectories, demonstrating improved sequence-internal retention.

activity, quantile-gating, and algorithmic details. Sections B, C, and D provide hyperparameters, ablations, and overhead measurements.

4.2 Effect on Forgetting Trajectories

We compare Vanilla TTT and the write-scaling intervention under identical checkpoint, sequence, probe, anchor, parameter-subset, and metric settings. The comparison changes only how optimizer-produced updates are scaled before they enter the parameter state.

For the primary TTT-E2E-3B model, the write-scaling panels of Figure 6 show that the write-scaling intervention shifts the forgetting trajectory. Across all three probe types, the write-scaling intervention yields lower early-probe loss at later anchors. The gap grows as the sequence progresses, suggesting that write-scaling intervention slows the cumulative degradation caused by later updates, rather than merely shifting the loss curve downward by a constant amount.

Importantly, this mitigating effect is not isolated to a single checkpoint. As detailed in Appendix E, applying the same write-scaling intervention to four additional TTT variants (TTT-Linear and TTT-MLP at both 1.3B and 760M scales) yields consistent reductions across architecture and scale.

The comparison is intended as a mechanism test, not as a claim that the write-scaling intervention solves long-context retrieval. However, the robust cross-variant evidence confirms that changing later writes changes the recoverability of earlier information, indicating that sequence-internal forgetting is fundamentally sensitive to write dynamics.

5 Stability–Plasticity Analysis

Reducing forgetting alone is not sufficient evidence of useful memory: freezing enough of the parameter state can protect old content while destroying adaptation to new inputs (Jung et al., 2023). We evaluate both *stability*, whether earlier information remains recoverable, and *plasticity*, whether newly observed information can still be written (Parisi et al., 2019; Lopez-Paz and Ranzato, 2017; Aljundi et al., 2018).

5.1 Anchor-Grid Protocol

The anchor-grid protocol measures stability and plasticity in a single online trajectory. We split each long sequence into intervals and sample one probe p_i from each interval using the probe forms in Section 3.2. At the anchor after interval i, the shadow stream evaluates p_i to test acquisition; at later anchors, it re-evaluates the same probe to test survival after additional updates.

This produces a lower-triangular matrix:

	$\boldsymbol{p}_1$	$\boldsymbol{p}_2$	$\cdots$	$\boldsymbol{p}_K$
a_1	$T_{1,1}$			
a_2	$T_{2,1}$	$T_{2,2}$		
$\vdots$	$\vdots$	$\vdots$	$\ddots$	
a_K	$T_{K,1}$	$T_{K,2}$	$\cdots$	$T_{K,K}$

where $T_{j,i}$ is the loss of probe p_i at anchor a_j. The diagonal entries $T_{i,i}$ define *plasticity*; for $j > i$, the degradation $T_{j,i} - T_{i,i}$ measures *stability*, averaged over the lower triangle. **Lower values are better for both metrics**. The grid separates writing new information from preserving old information.

5.2 Write-Control Mechanisms

We use this protocol to map the stability–plasticity profiles of several within-sequence write-control mechanisms. All methods share the same TTT-E2E-3B checkpoint, datasets, probes, anchors, update schedule, and updated parameter subset; runs

Method	CE-Token		CE-Answer		CE-Noun	
	Plasticity ↓	Stability ↓	Plasticity ↓	Stability ↓	Plasticity ↓	Stability ↓
Vanilla TTT†	**5.929**	3.047	**1.614**	1.434	**5.515**	4.335
write-scaling	6.548	1.418	2.779	0.238	6.058	1.724
L1	6.380	3.775	2.068	2.230	6.229	5.716
L2	5.991	3.181	1.677	1.591	5.629	4.588
SI	6.010	2.882	1.692	1.351	5.554	4.225
V-SoftMask	6.718	2.580	2.463	1.085	6.310	3.516
PackNet-Lite	7.973	**0.980**	3.627	**0.133**	6.358	**1.275**

Table 1: Stability–plasticity profiles of write-control mechanisms across three CE metrics. Plasticity columns report $10^2(P - 2.10)$, where P is the average diagonal loss and 2.10 is a fixed display offset used only to keep table entries compact; stability columns report $10^2 S$, where S is the average lower-triangular degradation. Lower values indicate better performance. **Bold** / underline: best / 2nd best per column. *Italics*: reference baseline (†).

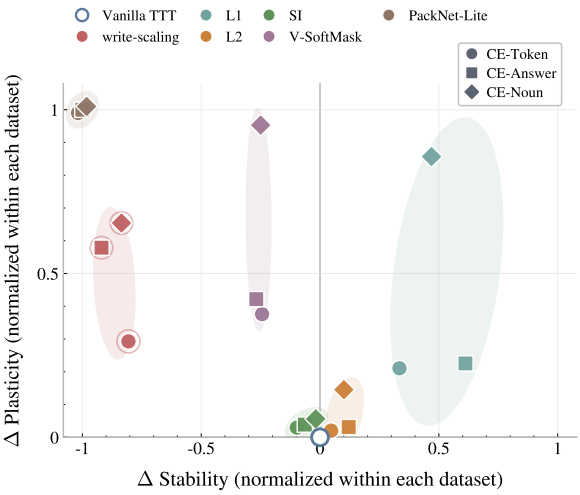

Figure 7: Baseline-relative stability–plasticity trade-off. Loss changes are measured relative to Vanilla TTT and normalized within each dataset, placing Vanilla TTT at the origin. Moving left indicates better stability and moving down indicates better plasticity, so **points in its lower-left region are more favorable**. Colors denote methods, marker shapes denote datasets, and ellipses summarize per-method spread.

use 128K-token sequences, 1024-token inner-loop chunks, 16K-spaced anchors, and 300-sequence averages.

The methods cover several write-control regimes: Vanilla TTT as the unprotected baseline (Tandon et al., 2025); L1/L2 drift penalties (Kirkpatrick et al., 2017); SI and V-SoftMask importance-based protection (Zenke et al., 2017; Ke et al., 2023); PackNet-Lite hard freezing (Mallya and Lazebnik, 2018); and the write-scaling intervention as a soft write-control mechanism. Here, the write-scaling intervention is included as a controlled operating point for testing write interference, rather than as a standalone memory architecture. Sections F and B describe the localized adaptations and hyperparameters.

5.3 Stability–Plasticity Profiles

Table 1 reports the raw scaled losses, while Figure 7 visualizes the same results relative to Vanilla TTT. For each dataset, we subtract the Vanilla TTT value from every method, placing Vanilla TTT at the origin. The x-axis shows the change in stability loss and the y-axis shows the change in plasticity loss; since lower is better for both, leftward movement means improved stability and downward movement means improved plasticity.

To account for different loss ranges across CE-Token, CE-Answer, and CE-Noun, we normalize these relative changes within each dataset by the maximum absolute change on each axis. Thus, the plot emphasizes operating regimes rather than raw cross-dataset magnitudes. Colors denote methods, marker shapes denote datasets, and translucent ellipses summarize each method's spread.

Plastic but unstable writes. Vanilla TTT and L2 preserve superior plasticity: their diagonal losses remain close to the lowest values in the comparison. They acquire new probes, but their stability is weak: the parameter state is writable but unstable.

Stable but rigid writes. PackNet-Lite occupies the opposite regime. It achieves the best stability across all three metrics, but at the largest plasticity cost. Hard protection reduces forgetting by making the state less able to absorb later inputs.

Soft write-scaling intervention. The write-scaling intervention occupies a soft write-control regime. Relative to Vanilla TTT, it reduces forgetting by 53.4%, 83.3%, and 60.2% on CE-Token, CE-Answer, and CE-Noun, respectively, while in-

curring only a small plasticity degradation. In the trade-off plane, this appears as a large leftward movement without the large upward shift of PackNet-Lite. The write-scaling intervention moves the model away from plastic-but-unstable behavior without collapsing into hard rigidity.

Overall. Forgetting mitigation must be interpreted together with plasticity. A useful parameter-state memory should preserve earlier writes while still admitting later ones.

6 Discussion

Clarification on "forgetting". Throughout this paper, we use "forgetting" in an operational, functional sense: an early context item is forgotten when its probe loss increases under a later parameter state, after removing KV-cache access and disabling further updates. This notion includes any mechanism by which later online updates reduce recoverability, including overwriting, readout drift, calibration changes, or shifts in local language-modeling preferences. We do not claim to identify or isolate a unique physical memory trace in the parameters.

Diagnostic interpretation. The shadow stream is not meant to predict downstream accuracy directly. It asks whether information once recoverable from the parameter state remains recoverable after subsequent writes, separating update-induced degradation from compression, retrieval, or decoding bottlenecks. The write-scaling intervention is also used only as a diagnostic intervention.

Interpreting CE-based recoverability. Since TTT-style memory is written through language-modeling-style online updates, CE-based probes are a task-native readout of parameter-state recoverability. Thus, sequence-internal forgetting in our setting is not limited to the loss of discrete facts, but refers to any decline in the ability of the evolving parameter state to support predictions about earlier sequence content.

7 Related Work

Long-context modeling and compressed memory. Long-context models reduce explicit attention cost with efficient attention (Dao et al., 2022; Liu et al., 2024) or replace token storage with recurrent, linear, state-space, or other compact memories (Rae et al., 2019; Katharopoulos et al., 2020; Sun et al., 2023; Gu and Dao, 2023). TTT-style models also compress context, but they write it through online parameter updates. We focus on whether this parameter-state memory remains stable after subsequent writes.

Test-time training and inference-time adaptation. Test-time training adapts a model at inference time using self-supervised objectives (Sun et al., 2020), with related work on entropy minimization, reconstruction, dynamic evaluation, and retrieved-text adaptation (Liu et al., 2021; Wang et al., 2020; Gandelsman et al., 2022; Krause et al., 2018; Hardt and Sun, 2024). Recent TTT-style long-context models repurpose such updates as memory (Sun et al., 2024; Tandon et al., 2025). We diagnose what happens after context has entered the parameter state.

Continual learning and catastrophic forgetting. Continual learning studies interference from sequential updates (McCloskey and Cohen, 1989; French, 1999; Goodfellow et al., 2013), using mechanisms such as importance regularization, masking, or freezing (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Mallya and Lazebnik, 2018). These mechanisms are useful reference points, but our setting differs: forgetting occurs within one test sequence, during inference, without task boundaries, replay, or offline consolidation.

8 Conclusion

This paper studies whether TTT-style parameter-state memory remains stable after storing earlier context. We identify sequence-internal forgetting, where information recoverable from an early parameter state becomes less recoverable after later online updates to the same state. Shadow-stream evaluation isolates this degradation from cache-based access and probe-induced adaptation, while Freeze Weights and the write-scaling intervention show that the measured degradation appears when later writes continue and changes when those writes are modulated. The anchor-grid protocol further shows why stability must be evaluated together with plasticity: a model should preserve earlier writes without losing the ability to absorb later inputs. These findings suggest that parameter-state stability should be treated as a core evaluation target for TTT-style models, alongside efficiency and context-length scalability.

Limitations

Model coverage. Our experiments cover several representative TTT-style variants, but they do not establish an impossibility result for all online-updating long-context models. Different update rules, larger parameter-state capacity, or hybrid memory designs may yield different stability–plasticity profiles. The contribution is diagnostic and empirical: we isolate a failure mode in the evaluated TTT-style family and provide tools for measuring it.

Probe coverage. CE-Token, CE-Answer, and CE-Noun measure recoverability from parameter-state memory under controlled conditions. They do not replace downstream benchmarks for reasoning, instruction following, or exact retrieval. The probes intentionally target a narrower question: after cache-based context is removed, does the current parameter state still make earlier content predictable?

Intervention scope. The write-scaling intervention addresses update-induced degradation after information has been written into the parameter state. It does not guarantee that rare strings, structured facts, or arbitrary evidence are faithfully compressed in the first place. End-to-end long-context retrieval likely requires complementary mechanisms, including stronger compression objectives, explicit memory, retrieval modules, or hybrid attention-state designs.

Trade-off scope. The anchor-grid protocol compares write-control mechanisms under matched conditions, but the reported operating points depend on the chosen models, probes, and hyperparameter budgets. The protocol should be used as a diagnostic profile rather than as a universal scalar measure of memory quality.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154.

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

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

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. 2022. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Moritz Hardt and Yu Sun. 2024. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations*, volume 2024, pages 54625–54640.

Dahuin Jung, Dongjin Lee, Sunwon Hong, Hyemi Jang, Ho Bae, and Sungroh Yoon. 2023. New insights for the stability-plasticity dilemma in online continual learning. *arXiv preprint arXiv:2302.08741*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pre-training of language models. *Preprint*, arXiv:2302.03241.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2018. Dynamic evaluation of neural sequence models. In *International Conference on Machine Learning*, pages 2766–2775. PMLR.

Jeffrey Li, Mohammadreza Armandpour, Seyed Iman Mirzadeh, Sachin Mehta, Vaishaal Shankar, Raviteja Vemulapalli, Samy Bengio, Oncel Tuzel, Mehrdad Farajtabar, Hadi Pouransari, and 1 others. 2025. Ticlm: A web-scale benchmark for time-continual llm pretraining. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32231–32273.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2024. Ringattention with blockwise transformers for near-infinite context. In *International Conference on Learning Representations*, volume 2024, pages 3992–4008.

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

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773.

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

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

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. 2021. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, and 1 others. 2024. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

VP Swamynathan. 2026. Sr-ttt: Surprisal-aware residual test-time training. *arXiv preprint arXiv:2603.06642*.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, and 1 others. 2025. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2020. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. Pmlr.

A Write-Scaling Intervention Details

This appendix specifies the write-scaling intervention used in Section 4. The write-scaling intervention damps future updates along parameter directions that have been repeatedly active during earlier online adaptation, while preserving nonzero update magnitude for continued plasticity. The rule consists of tensor-wise gradient normalization, activity construction, per-tensor quantile gating, importance-based scaling, update modulation, and an EMA update of the importance estimate. Algorithm 1 summarizes the complete update.

Formally, let $g_{t,i}^{(\ell)}$ denote the gradient of the i-th parameter in tensor ℓ at step t, let $u_{t,i}^{(\ell)}$ denote the raw update produced by the optimizer, and let $m_{t-1,i}^{(\ell)}$ denote the exponential moving average (EMA) of the historical importance score from the previous step. The rule computes a scaling factor from $m_{t-1,i}^{(\ell)}$ and applies it directly to the optimizer-produced update.

Step 1: Gradient normalization. When tensor-wise RMS normalization is enabled, we define

$$\hat{g}_{t,i}^{(\ell)} = \frac{g_{t,i}^{(\ell)}}{\sqrt{\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}(g_{t,j}^{(\ell)})^2 + \varepsilon}}, \tag{4}$$

where n_ℓ is the number of parameters in tensor ℓ. Without RMS normalization, we set

$$\hat{g}_{t,i}^{(\ell)} = g_{t,i}^{(\ell)}. \tag{5}$$

The normalization reduces scale differences across tensors before activity is computed.

Step 2: Activity construction. By default, the write-scaling rule uses squared normalized gradients as the activity signal:

$$a_{t,i}^{(\ell)} = \left(\hat{g}_{t,i}^{(\ell)}\right)^2. \tag{6}$$

More generally, the activity can be written as

$$a_{t,i}^{(\ell)} = \begin{cases} |\hat{g}_{t,i}^{(\ell)}|, & \text{if activity = abs_grad,} \\ (\hat{g}_{t,i}^{(\ell)})^2, & \text{if activity = sq_grad.} \end{cases} \tag{7}$$

The activity signal is used as a proxy for how strongly each parameter participates in the current update.

Step 3: Per-tensor quantile gating. For each tensor ℓ, we compute an activity threshold based on the within-tensor quantile:

$$\tau_t^{(\ell)} = Q_q\left(\{a_{t,j}^{(\ell)}\}_{j=1}^{n_\ell}\right), \tag{8}$$

where $q \in [0,1]$ is the `gate_quantile`. We then define the gated activity as

$$\tilde{a}_{t,i}^{(\ell)} = \max\left(\frac{a_{t,i}^{(\ell)} - \tau_t^{(\ell)}}{\tau_t^{(\ell)} + \varepsilon}, 0\right). \tag{9}$$

The gate restricts the importance estimate to activity substantially above the within-tensor background level, yielding a sparser importance pattern.

Step 4: Protection term and scaling. The scaling applied at the current step is determined solely by the previous-step importance:

$$p_{t,i}^{(\ell)} = \lambda_{\text{imp}}\left(m_{t-1,i}^{(\ell)}\right)^\gamma, \tag{10}$$

$$s_{t,i}^{(\ell),\text{pre}} = \frac{1}{1 + p_{t,i}^{(\ell)}}, \tag{11}$$

$$s_{t,i}^{(\ell)} = \text{clip}\left(s_{t,i}^{(\ell),\text{pre}},\, s_{\min},\, 1\right). \tag{12}$$

Here, λ_{imp} controls the overall protection strength, γ controls the nonlinearity of protection, and $s_{\min}$ preserves a minimum level of plasticity. When $\gamma > 1$, highly important parameters are suppressed more aggressively, while low-importance parameters are affected more mildly.

Step 5: Update modulation. The final modulated update is

$$u_{t,i}^{\prime(\ell)} = s_{t,i}^{(\ell)}\, u_{t,i}^{(\ell)}. \tag{13}$$

The rule rescales the optimizer-produced update rather than the raw gradient, making it compatible with optimizers that maintain internal state, such as momentum SGD or Adam.

Step 6: Importance EMA update. After the current update has been modulated, we update the importance EMA using the gated activity:

$$m_{t,i}^{(\ell)} = \beta\, m_{t-1,i}^{(\ell)} + (1-\beta)\, \tilde{a}_{t,i}^{(\ell)}. \tag{14}$$

Initialization. At the beginning of processing, the importance EMA is initialized as

$$m_{0,i}^{(\ell)} = 0. \tag{15}$$

Algorithm 1 summarizes the complete procedure at one TTT update step.

Algorithm 1 Write-Scaling intervention at TTT step t

Require: $\{g_t^{(\ell)}\}_{\ell=1}^{L}$ (gradients), $\{u_t^{(\ell)}\}_{\ell=1}^{L}$ (optimizer updates)
$\{m_{t-1}^{(\ell)}\}_{\ell=1}^{L}$ (EMA importance)
Require: $\beta,\ \lambda_{\text{imp}},\ \gamma,\ s_{\min},\ q,\ \varepsilon$

for each tensor $\ell = 1, \ldots, L$ **do**
  $\hat{g}_t^{(\ell)} \leftarrow \text{normalize}(g_t^{(\ell)})$ ▷ Eqs. (4), (5)
  $a_t^{(\ell)} \leftarrow \text{activity}(\hat{g}_t^{(\ell)})$ ▷ Eq. (7)
  $\tau_t^{(\ell)} \leftarrow \text{quantile}(a_t^{(\ell)},\ q)$ ▷ Eq. (8)
  $\tilde{a}_t^{(\ell)} \leftarrow \text{gate}(a_t^{(\ell)},\ \tau_t^{(\ell)})$ ▷ Eq. (9)
  $p_t^{(\ell)},\ s_t^{(\ell)} \leftarrow \text{protect}(m_{t-1}^{(\ell)})$ ▷ Eqs. (10)–(12)
  $u_t'^{(\ell)} \leftarrow s_t^{(\ell)} \odot u_t^{(\ell)}$ ▷ modulate update
  $m_t^{(\ell)} \leftarrow \beta\, m_{t-1}^{(\ell)} + (1-\beta)\, \tilde{a}_t^{(\ell)}$ ▷ EMA importance
return $\{u_t'^{(\ell)}\}_{\ell=1}^{L},\quad \{m_t^{(\ell)}\}_{\ell=1}^{L}$

B Hyperparameter and Model Configurations

This section summarizes the main hyperparameters and model configurations used in our experiments. Unless otherwise specified, all experiments are conducted on an RTX Pro 6000 GPU.

TTT-E2E. The TTT-E2E model has 3B parameters and consists of 32 layers, a hidden size of 2560, 32 attention heads, an intermediate size of 5632, and a vocabulary size of 128256. The maximum sequence length is 131072, the TTT micro-batch size is 1024, and the sliding-window attention length is 8192. The feed-forward network uses SwiGLU activation, and positional encoding uses RoPE with hyperparameter $\theta = 500000$. QK normalization is enabled, and both pre-normalization and post-normalization are used. Word embedding weights are tied. Model computation is performed in bf16, while parameters and optimizer states are stored in fp32.

For the sequence-internal forgetting evaluation, each evaluation is run on 300 sequences. Probing anchors start at 16384 tokens, with a stride of 16384 tokens, and continue up to 131072 tokens. For probe sampling, the CE-Token probe length is 128 tokens, and the local prefix length is 64 tokens. The maximum probe input length for CE-Answer is 1024 tokens. The CE-Noun probe window length is 128 tokens, and the local prefix length is 32 tokens. All three metrics are randomly sampled from the first 8192 tokens.

TTT-Linear. The TTT-Linear experiments use two model scales. The 1.3B model consists of 24 layers, a hidden size of 2048, 32 attention heads, an intermediate size of 5504, and a vocabulary size of 32000. The 760M model consists of 24 layers, a hidden size of 1536, 16 attention heads, an intermediate size of 4096, and a vocabulary size of 32000. Both models use TTT-Linear as the sequence-modeling module, with a TTT micro-batch size of 16 and a TTT base learning rate of 1.0. A pre-causal convolution with width 4 is used. The feed-forward network uses SwiGLU activation, normalization uses RMSNorm, and positional encoding uses RoPE with hyperparameter $\theta = 10000$. Word embedding weights are tied. Model computation is performed in bf16.

For the sequence-internal forgetting evaluation, each evaluation is run on 300 sequences. The maximum sequence length is 8192; probing anchors start at 1024 tokens, with a stride of 1024 tokens, and continue up to 8192 tokens. For probe sampling, the CE-Token probe length is 128 tokens, and the local prefix length is 64 tokens. The maximum probe input length for CE-Answer is 1024 tokens. The CE-Noun probe window length is 128 tokens, and the local prefix length is 64 tokens. All three metrics are randomly sampled from the first 1024 tokens.

TTT-MLP. The TTT-MLP experiments also use two model scales: 1.3B and 760M. The 1.3B model consists of 24 layers, a hidden size of 2048, 32 attention heads, an intermediate size of 5504, and a vocabulary size of 32000. The 760M model consists of 24 layers, a hidden size of 1536, 16 attention heads, an intermediate size of 4096, and a vocabulary size of 32000. Both models use TTT-MLP as the sequence-modeling module, with a TTT micro-batch size of 16, a TTT base learning rate of 0.1, and an initial learning rate of 0.01. The TTT learning-rate warmup is 5000 steps for the 1.3B model and 2900 steps for the 760M model. The models use a pre-causal convolution with width 4. The feed-forward network uses SwiGLU activation, normalization uses RMSNorm, and positional encoding uses RoPE with hyperparameter $\theta = 10000$. Word embedding weights are tied. Model computation is performed in bf16.

For the sequence-internal forgetting evaluation, each evaluation is run on 300 sequences. The maximum sequence length is 8192; probing anchors start at 1024 tokens, with a stride of 1024 tokens, and continue up to 8192 tokens. For probe sampling, the CE-Token probe length is 128 tokens, and the local prefix length is 64 tokens. The maximum probe input length for CE-Answer is 1024

tokens. The CE-Noun probe window length is 128 tokens, and the local prefix length is 64 tokens. All three metrics are randomly sampled from the first 1024 tokens.

Write-scaling intervention. The default write-scaling configuration is as follows: EMA decay $\beta = 0.99$, protection strength $\lambda_{\text{imp}} = 0.1$, nonlinearity $\gamma = 2$, minimum scale $s_{\min} = 0.1$, and $\varepsilon = 10^{-8}$. Activity is normalized by RMS for each tensor, and per-tensor quantile gating uses `gate_quantile`=0.99. The rule is applied to all updated parameters.

C Write-Modulation Ablations

Table 2 and Figure 8 ablate the factors that determine the write-scaling operating point. Starting from the default configuration–squared-gradient activity, EMA decay $\beta = 0.99$, nonlinear protection $\gamma = 2$, minimum scale $s_{\min} = 0.1$, and gate quantile 0.99–we test five variants: linear protection ($\gamma = 1$), no temporal accumulation ($\beta = 0$), no scaling floor (*No Floor*), a less sparse gate (*q=0.90*), and absolute-gradient activity (*Use abs-grad*).

The ablations separate mechanism-critical choices from operating-point controls. Replacing squared activity with absolute activity weakens forgetting loss reduction across all probe types, especially CE-Answer and CE-Noun. Removing temporal accumulation or nonlinear protection also increases forgetting loss. These results suggest that squared activity, EMA accumulation, and nonlinear scaling are the main ingredients that make the write-scaling rule targeted.

The floor and gate threshold behave differently. Removing the floor or using a less sparse gate can reduce forgetting further, but at the cost of worse plasticity. These choices control where the variant lands on the stability–plasticity plane rather than uniformly improving the rule. The default configuration is a balanced operating point: it does not maximize protection, but avoids drifting toward hard-freezing behavior.

D End-to-End Computational Overhead

Because TTT-style models are motivated by inference efficiency, write-control mechanisms should be evaluated not only by their retention–plasticity profile but also by their runtime and state overhead. We profile the end-to-end cost of all compared mechanisms, including boundary-specific operations such as V-SoftMask recomputation, PackNet-Lite freezing and sorting, and SI consolidation. We report mean step latency, average overhead relative to Vanilla TTT, boundary and non-boundary latency, and persistent adaptation state.

Table 3 summarizes the results. Vanilla TTT has nearly identical latency on boundary and non-boundary chunks (140.1 ms vs. 139.8 ms), indicating that the profiling protocol itself does not introduce a boundary artifact. The write-scaling intervention increases mean step latency from 139.9 ms to 165.2 ms, corresponding to an 18.1% overhead, and adds 1.32 GiB of persistent adaptation state: one EMA tensor over the adapted inner parameters.

The cost profile of the write-scaling intervention is smooth across the sequence. Boundary and non-boundary latencies are nearly identical (165.7 ms vs. 165.2 ms), unlike mechanisms whose maintenance work concentrates at anchors. SI has slightly lower average compute overhead but requires substantially larger persistent state. V-SoftMask and PackNet-Lite have modest non-boundary overhead but large boundary spikes, leading to worse tail latency.

The write-scaling intervention provides a lightweight soft write-control operating point in this comparison. Its retention gains do not require heavy state growth or expensive boundary-triggered computation; the cost is distributed across ordinary update steps.

E Extended Evaluation of the Write-Scaling Intervention on Additional TTT Variants

To verify the generalizability of the write-scaling intervention used in Section 4.2, we conduct extended evaluations on four open-source TTT variants beyond the primary model: TTT-Linear-1.3B, TTT-Linear-760M, TTT-MLP-1.3B, and TTT-MLP-760M.

This experiment strictly follows the identical controlled settings used in Section 3.5: For all four variants, we uniformly evaluate 8K-token contexts, sampling probes from the first 1K tokens and scoring them at evenly distributed anchors ranging from 1K to 8K. Within each variant, we compare the write-scaled TTT run against the Vanilla TTT baseline using the same checkpoint, dataset, sequences, probes, and metrics. The comparison changes only

Method	CE-Token		CE-Answer		CE-Noun	
	plasticity	stability	plasticity	stability	plasticity	stability
Default write-scaling	2.19894	0.01308	2.30221	0.00290	2.14602	0.02045
$\gamma = 1$	2.19444	0.01708	2.29550	0.00481	2.14357	0.02560
$\beta = 0$	2.19832	0.01709	2.30097	0.00412	2.14733	0.02587
No Floor	2.20221	0.01039	2.30511	0.00214	2.14622	0.01663
$q = 0.90$	2.21342	0.00819	2.31604	0.00001	2.15805	0.01583
Use absgrad	2.19317	0.02310	2.29091	0.00972	2.14774	0.03680

Table 2: Ablations of the write-scaling intervention under the stability–plasticity protocol. Lower values are better for both metrics. Removing temporal accumulation ($\beta = 0$), weakening nonlinear protection ($\gamma = 1$), or replacing squared activity with absolute gradients weakens forgetting loss reduction, while removing the floor or adopting a less sparse gate ($q = 0.90$) further reduces forgetting at the cost of worse plasticity.

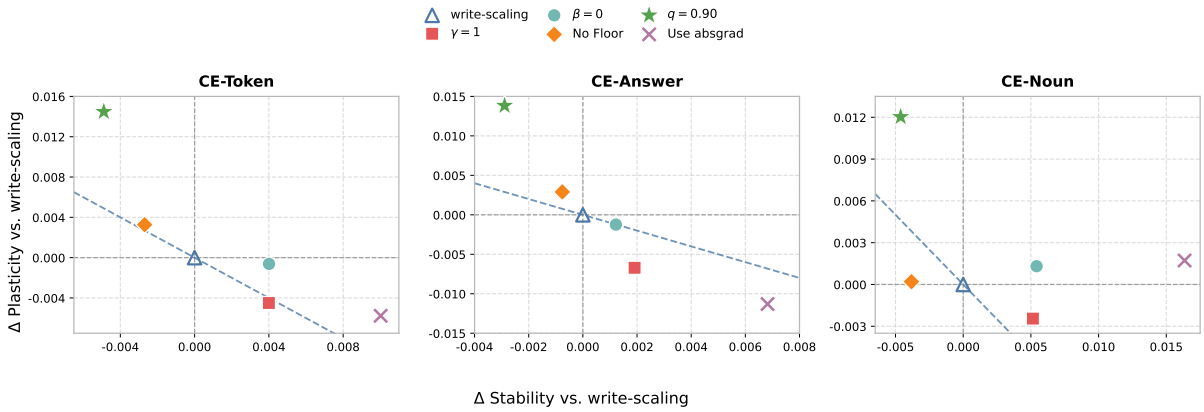

Figure 8: Ablation trade-off under the three probe metrics. Each point denotes one ablated variant, plotted by its change in forgetting loss and plasticity loss relative to the default write-scaling rule. The plot shows how individual design choices move the write-control operating point on the stability–plasticity plane.

how optimizer-produced updates are scaled before entering the parameter state, without altering any other hyperparameters.

The results of the extended evaluation are fully presented in Figure 9. The empirical data demonstrate that the write-scaling intervention exhibits robustness across architectures and scales.

Across all four additional variants, the write-scaling intervention successfully alters the forgetting trajectory of the baseline models under token-level (CE-Token), answer-level (CE-Answer), and noun-level (CE-Noun) probing metrics. As the sequence progresses to deeper anchors, the early-probe loss of Vanilla TTT rises continuously, whereas the write-scaled variants significantly flatten this trajectory. The widening performance gap between the two curves as sequence length increases confirms that the intervention actively slows the cumulative degradation caused by subsequent online updates, rather than merely shifting the loss curve downward.

Because this trend remains highly consistent across both Linear and MLP architectures at the 760M and 1.3B parameter scales, it provides strong empirical support for the mechanism hypothesis in Section 4.2, indicating that the sensitivity of sequence-internal forgetting to write dynamics is not an artifact of a single model.

F Localized Write-Control Baselines

The baselines in Section 5 were originally designed for continual-learning settings with explicit task boundaries. Sequence-internal forgetting has a different structure: interference occurs within one test sequence while the model performs online updates. To make these methods meaningful in our setting, we localize their core stability mechanisms to anchor-bounded sequence intervals.

These adaptations should be read as controlled within-sequence instantiations, not as faithful reproductions of the full task-level algorithms. Their purpose is not to establish an exhaustive ranking of continual-learning methods. Instead, they provide reference mechanisms that occupy different

Method	Mean (ms)	Avg. OH (%)	Boundary (ms)	Non-boundary (ms)	State (GiB)
Vanilla TTT	139.9	0.0	140.1	139.8	0.00
write-scaling	165.2	+18.1	165.7	165.2	1.32
L1	167.5	+19.8	265.2	156.5	1.32
L2	168.6	+20.5	268.1	157.3	1.32
SI	160.3	+14.6	160.1	160.3	5.28
V-SoftMask	164.8	+17.9	288.1	150.9	3.28
PackNet-Lite	216.4	+54.7	837.7	146.2	1.65

Table 3: End-to-end overhead profiling. We report average step latency, compute overhead relative to Vanilla TTT, boundary/non-boundary latency, and persistent adaptation state. The write-scaling intervention adds one EMA tensor over the adapted parameters and has no boundary-specific latency spike.

regions of the retention–plasticity landscape.

All baselines follow four shared rules. First, task boundaries are replaced by anchor boundaries. Second, each mechanism operates only on the inner-loop parameter subset updated by TTT. Third, all auxiliary states are reset at the beginning of each evaluation sequence, so no method accumulates information across sequences. Fourth, only one write-control mechanism is enabled in each run. These constraints keep the comparison focused on within-sequence stability rather than on mismatched assumptions from task-incremental learning.

Vanilla TTT. Vanilla TTT is the unregularized reference point. It uses the standard inner-loop online updates with the same checkpoint, probes, anchors, and updated parameter subset as all other methods.

L1 and L2 drift regularization. L1 and L2 are sequence-local drift controls. After a predefined activation point, we store a reference copy of the inner parameters and penalize subsequent deviation from that reference. The reference exists only within the current evaluation sequence and is reset before the next sequence.

Synaptic Intelligence (SI). SI estimates parameter importance from the interaction between gradients and parameter movement. We adapt it by treating each anchor interval as a pseudo-task. Importance statistics are accumulated within the interval, consolidated at the next anchor boundary, and then used to regularize later movement within the same sequence. All SI state is reset between sequences.

V-SoftMask. V-SoftMask is a soft importance-masking baseline. At anchor boundaries, it recomputes gradients on cached chunks from the recent interval, converts them into tensor-normalized importance scores, and accumulates a within-sequence importance map. Future gradients are attenuated according to this map, with a nonzero floor to avoid removing plasticity entirely.

PackNet-Lite. PackNet-Lite represents hard protection. At each anchor boundary, it scores inner parameters by how much they changed during the preceding interval and freezes a top-scoring subset subject to a maximum frozen budget. Unlike full PackNet, this variant performs no retraining, stores no task-specific masks, and carries no state across sequences.

Overall. These localized mechanisms span direct drift control, path-integral importance consolidation, soft masking, and hard freezing. They allow the anchor-grid protocol to distinguish whether an apparent retention gain reflects softer write control or reduced plasticity.

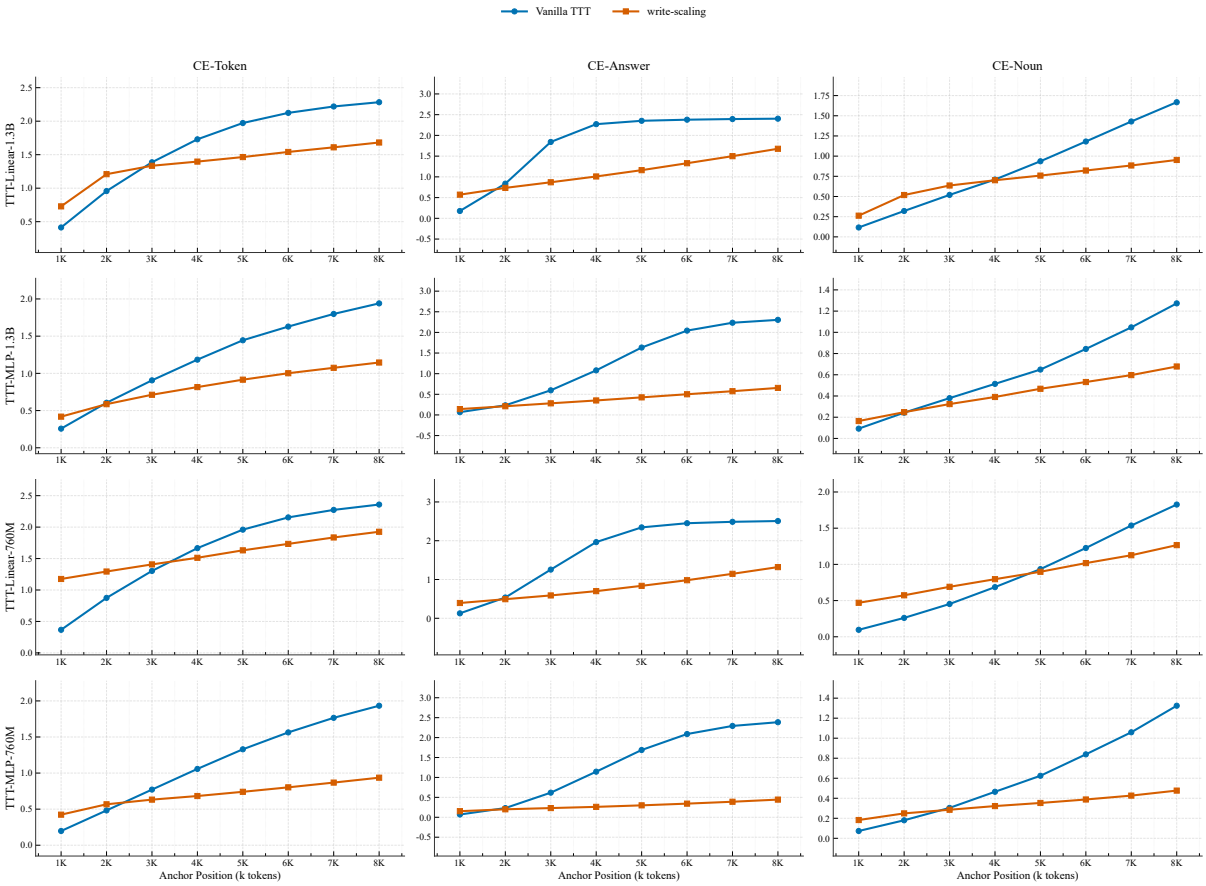

Figure 9: Effect of the write-scaling intervention across four additional TTT variants. The plots show the cross-entropy at each anchor. The results show that across different memory architectures (Linear and MLP) and parameter scales (760M and 1.3B), the write-scaling intervention significantly flattens the forgetting curves for all three probe granularities (CE-Token, CE-Answer, and CE-Noun), validating the general effectiveness of this mechanism in mitigating sequence-internal forgetting.