Deployment-Memory LLM Test-Time Training Should Require Behavioral Evidence Beyond Perplexity

Xiangchen Song[1] **Zhenhao Chen**[2] **Lingjing Kong**[1] **Shaoan Xie**[1]
Xinshuai Dong[1] **Guangyi Chen**[1,2] **Kun Zhang**[1,2]
[1]Carnegie Mellon University [2]MBZUAI
{xiangchensong,kunz1}@cmu.edu

Abstract

Large language model test-time training (TTT) is often evaluated through local proxy metrics: models are updated on recent tokens, retrieved context, target-domain data, or verifiable task attempts, and then judged by perplexity, future-token loss, long-context performance, or reward. These metrics are well matched to claims about stream adaptation, domain adaptation, context compression, and reward-backed test-time improvement. This position paper argues that TTT has a distinct claim-calibration problem: proxy evidence for local adaptation can migrate into stronger claims about deployed assistant memory, personalization, or sparse post-deployment learning. Such claims require behavioral evidence: later recall, paraphrase robustness, retention, locality, conflict handling, and use in downstream actions after the original support context is removed. We propose a claim-calibrated evidence ladder for LLM TTT, audit recent work through this lens, and provide a diagnostic counterexample in which proxy losses improve without behavioral recall. Our goal is to give authors and reviewers a standard for aligning TTT memory claims with the evidence actually reported.

1 Introduction

Test-time training (TTT) challenges the conventional "train, then deploy" boundary by allowing model states or parameters to change during inference. In large language models, recent work has made this idea technically concrete: models may update from retrieved neighbors [11], learn online hidden-state updates through fast weights [25], perform large-chunk updates for throughput and state capacity [35], or align online updates with next-token prediction [9]. Related work studies targeted context-specific updates [4], meta-learned long-context learning [28], parameter-efficient context memories [7], locally supported parametric memories [17], input-perplexity minimization [13], label-free uncertainty signals [32], unlabeled reinforcement learning [40], and self-directed update data [41, 1].

Across these settings, the evaluation recipe is relatively stable. A model is updated at test time on recently observed tokens, retrieved examples, task attempts, or generated data, and performance is then reported through lower perplexity, better future-token prediction, improved long-context accuracy, or higher reward. These results show real progress on in-sequence adaptation and compact use of recent context, often addressing practical limits of transformer-based systems such as finite context windows or static parameters. They also explain why TTT is attractive for LLM systems: online updates may allow a model to adapt to new evidence rather than relying only on fixed parameters or the current prompt.

This paper argues that this evidence is not always calibrated to the claims it is used to support. In real-world assistant settings, the relevant question is often not whether an update improves prediction on a nearby continuation. It is whether a deployed model can hear a sparse user preference, acquire a

Preprint.

project-specific fact, revise a stale belief, or learn a procedure, and then use that information later after the original support context is removed.

Our position is that LLM test-time training claims about memory, personalization, or sparse post-deployment learning require behavioral evidence beyond perplexity.

The fact that perplexity is an imperfect proxy for downstream behavior is not new; similar lessons appear in model editing, retrieval, and memory evaluation. The distinctive issue for LLM TTT is that inference-time parameter or state updates make local likelihood gains especially easy to reinterpret as evidence of learning, memory, or personalization. We call this failure mode *evidence migration*: evidence that directly supports one evaluation regime is carried into a stronger deployment narrative without the behavioral tests required for that narrative.

The key distinction is between *in-sequence adaptation* and *deployment-time behavioral learning*. In in-sequence adaptation, a model observes a token stream, updates its state or parameters on a support chunk, and is evaluated on future tokens from the same stream. In deployment-time behavioral learning, a deployed model receives sparse but high-value interactions—facts, preferences, corrections, or procedures—and must later use them in a personalized and stable way. Local loss reductions can strongly support the first setting while leaving the second only partially tested. Figure 1 illustrates this split with a sparse user-stated preference that should influence a later response.

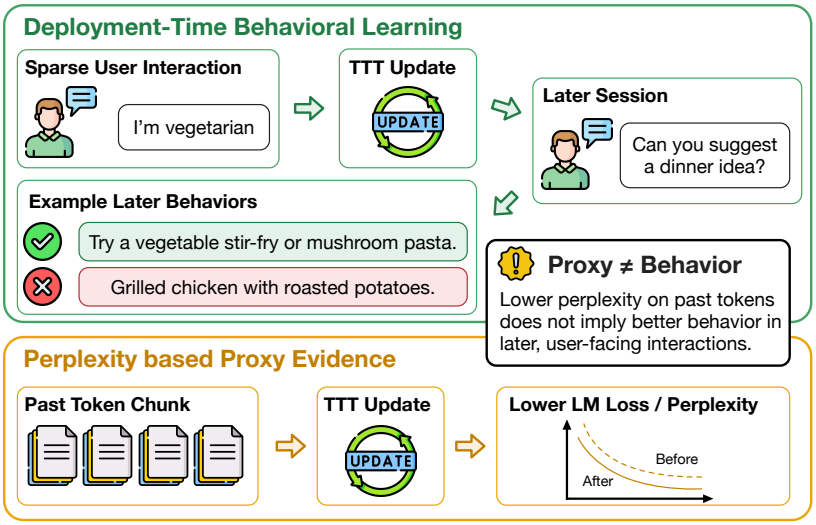

Figure 1: Two evaluation paradigms. Top: deployment-time behavioral learning from a sparse user interaction, illustrated with a user-stated dietary preference and evaluated by whether that preference changes later user-facing behavior. Bottom: perplexity-based evidence, where the model updates on a past support token chunk and achieves lower loss on a future chunk.

This distinction matters because downstream systems are beginning to draw on recent TTT work for memory-like capabilities. Many long-context, label-free adaptation, and reward-backed TTT methods primarily report perplexity, likelihood, uncertainty, throughput, long-context accuracy, or task reward [25, 35, 9, 4, 28, 13, 32, 40]. A closer frontier studies parameter-efficient context memories, self-adaptation, and context internalization [7, 30, 17, 37]. These are important steps toward memory-like systems, but deployment-time memory adds a further requirement: sparse user information should remain usable after delay and should be evaluated against retrieval or long-context baselines.

Many existing TTT papers are properly grounded in proxy evaluation and do not themselves claim to solve deployment-time memory. The problem arises when such results are later cited, summarized, or motivated as evidence for stronger claims about memory, personalization, or self-updating assistants. Table 1 summarizes common evidence migration patterns and the behavioral tests needed to support the stronger deployment-time narratives.

We use TTT as an umbrella term for inference-time methods that update model state, especially parameters or fast weights, and treat LLM *test-time learning* (TTL) as a neighboring adaptation formulation. We use *memory*, *continual learning*, and *deployment-time behavioral learning* for the stronger claim that new information encountered after deployment can be acquired and later used.

Table 1: Evidence migration patterns. Evidence that is well grounded for one claim can be used to motivate stronger deployment-time narratives that require additional behavioral tests.

Pattern	Grounded evidence	Deployment-time claim	Missing behavioral evidence
Token-stream TTT	lower future-token loss or long-context task gain	the model learns user facts online	sparse fact recall after context removal, paraphrase, and delay
Context compression / internalization	needle recall, context ablation, or parametric-memory evidence	forms persistent assistant memory	cross-session recall, locality, and conflict behavior
Self-adaptation from generated data	task improvement under generated updates	personalized continual learning	weak user evidence, user-specific retention, and correction
Reward-rich test-time discovery	verifiable reward improvement	broad deployment self-improvement	reward-poor user facts, preferences, and procedures

When we discuss *future-token gain*, we mean improvement on nearby held-out continuations after a test-time update.

Concretely, this paper makes four contributions:

1. We identify *evidence migration*: the risk that TTT results grounded in perplexity, reward, or same-stream adaptation are used to support stronger claims about memory, personalization, or post-deployment learning.
2. We formalize a claim-calibrated evidence ladder, ranging from proxy evidence for stream adaptation, to evidence for context internalization, to behavioral evidence for deployment-time memory.
3. We audit recent TTT and memory-adjacent work through this ladder, distinguishing what current evaluations directly support from what additional tests would be needed for stronger deployment-time claims.
4. We provide a sanity-check counterexample and a modular reviewer protocol for matching memory, personalization, correction, and agent-memory claims to appropriate behavioral evidence.

2 Perplexity is an Incomplete Proxy

Perplexity remains valuable. It is often the cleanest early signal that a test-time update has changed the model in a nontrivial way. The problem is not that perplexity is uninformative, but that it answers a narrower question than many deployment-time claims require. Lower loss can arise from mechanisms that are useful for stream adaptation but insufficient for post-deployment learning.

Continuous text is a dense-evidence regime. When support and evaluation chunks come from the same passage, they share topic, entities, lexical choices, and local discourse structure. Lower future-token loss may therefore reflect continuation fitting, topic priming, or memorization of nearby regularities, which is precisely what dynamic evaluation [15, 22] was designed to exploit. This is a legitimate stream-adaptation benefit. However, it does not require the model to form a stable, queryable representation of the underlying fact. Dense same-stream evidence is therefore weaker evidence for sparse deployment-time learning.

Associative retrieval differs from semantic acquisition. Long-context TTT can also succeed by building efficient associative state for the current input. Some long-context TTT results are consistent with the view that TTT-style state acts as a compressed associative memory over the current context. For example, TTT-E2E [28] connects TTT-style updates to key-value binding mechanisms and reports that full attention remains substantially stronger on needle-in-a-haystack recall, partly because compressed TTT-style state can discard details needed for exact retrieval. This supports context compression. It is weaker evidence for acquiring a reusable fact, rule, preference, or procedure that should survive paraphrase and delay.

Teacher forcing is easier than behavioral extraction. A model can assign higher probability to the gold continuation of a support fact without later producing the correct answer to a free-form query about that fact. This gap is especially important under open-ended decoding: a small increase in the probability of the right answer tokens may still be insufficient for greedy generation, beam search, or robust answer selection under paraphrase. Lower teacher-forced loss is therefore evidence that

the update moved probability mass in a useful direction, but it does not guarantee that the learned information is accessible during open-ended generation. A system can improve language-model loss while still failing to retrieve the new knowledge behaviorally.

Cumulative stream adaptation can be mistaken for sparse-interaction learning. In standard continuous-text TTT protocols, updates often accumulate across a long input stream. This is appropriate when the goal is online adaptation to that stream. It becomes harder to interpret when the same result is used as evidence that a single sparse user interaction can be internalized and reused later. A stream protocol can conflate immediate adaptation to the current support item with cumulative adaptation from all preceding chunks. Deployment-time claims therefore need reset-vs-stream comparisons, or equivalent controls, when the protocol permits them.

Local LM metrics do not test generalization, locality, or conflict. Deployment-time learning requires more than improved prediction of nearby text. The update should persist under paraphrase, delay, mild distribution shift, and conflicting later evidence, while leaving unrelated behavior intact. The model editing literature provides a useful cautionary parallel: ROME [19] and MEMIT [20] made efficacy, generalization, and specificity explicit, while later evaluations found that short-form edit tests can miss specificity failures [12], multi-hop failures [39], long-form failures [23], gradual and catastrophic forgetting under scale [10], broader evaluation mismatches for edited models [16], and deployment-style failures in the wild [33]. The same lesson carries over to TTT memory claims: a locally valid success metric can overstate the practical capability of interest when the claim concerns delayed, user-facing behavior.

Takeaway: Support reconstruction, future-token loss, answer NLL, and, when used, candidate ranking should remain in TTT reports because they are informative mechanism-level signals. For deployment-time TTT, however, they should be labeled as *proxy metrics* and kept separate from headline behavioral evidence.

3 Deployment-Time Memory Requires Behavioral Evidence

This limitation does not invalidate existing TTT evaluations for stream adaptation, long-context compression, domain adaptation, or reward-backed test-time improvement. The issue is claim calibration: these evaluations do not by themselves establish the stronger deployment-time capabilities implied by memory, personalization, or self-updating assistant narratives.

The regime split matters because memory-style deployment is no longer an abstract desideratum. It is already the target of direct behavioral evaluation. LoCoMo [18] evaluates very long-term conversational memory; LongMemEval [31] benchmarks long-term interactive memory in chat assistants; MemoryAgentBench [14] evaluates memory through incremental multi-turn interactions; MemoryBench [2] targets memory and continual learning in LLM systems; MemoryCD [36] studies lifelong cross-domain personalization; and Mem2ActBench [24] evaluates long-term memory utilization in task-oriented agents. These benchmarks ask whether information from one or a few interactions can be reused after delay, paraphrase, abstention pressure, or conflict. Improving nearby continuation loss after a dense support chunk answers a different question.

Deployment-time learning differs from pretraining and continuous-text adaptation in three ways. It is *low redundancy*: a fact, preference, correction, or procedure may be stated only once. It is *weak and heterogeneous*: user evidence often arrives as short, informal utterances rather than clean supervised examples. It is also *delayed and behavioral*: the model must later answer, revise, abstain, route, or act according to the learned information without damaging unrelated behavior. These properties make behavioral probes necessary: the central question is whether the update changes later behavior in the way required by the claimed deployment use case, not merely whether it improves local likelihood.

The distinction is especially clear in settings with stronger supervision. TTT-Discover [34] shows that test-time training can improve search when the environment supplies verifiable rewards and many attempts can be scored. Personalized assistants face a harder frontier: semantic internalization under weak evidence, as explored by SEAL [41] and Absorber LLM [37], and stressed by direct memory benchmarks such as MemoryBench [2]. Reward-rich discovery and dense stream adaptation are promising, but sparse deployment-time memory imposes a separate behavioral burden.

Matched explicit-memory baselines are therefore central. If the deployed task is to remember a user fact, preference, correction, or procedure, then retrieval, long-context prompting, and memory systems that store and reuse the same evidence are natural comparators. MemoryBank [38] stores long-term user memory, LongMem [29] augments language models with long-term memory, MemGPT [21] manages memory through an operating-system-like architecture, Mem0 [8] targets scalable production-ready long-term memory for agents, Dynamic Cheatsheet [27] maintains an adaptive test-time memory, and MEMORYLLM [30] studies self-updatable language models. These systems retain deployment-time information outside ordinary base-model weights and retrieve, page, curate, or pool it when needed. They need not be framed as competitors that TTT must always beat. Rather, they define the behavioral and systems-level tradeoff that a parametric update must justify.

Takeaway: Deployment-time memory claims require behavioral tests against matched explicit-memory baselines. These comparisons reveal whether parametric TTT adds value under privacy, latency, compression, or context-pressure constraints.

4 Audit

Claim levels. We audit recent TTT and memory-adjacent work using three claim levels. *S-level* evidence denotes stream or domain adaptation: the model is updated on a recent stream, target-domain data, retrieved examples, or task attempts, and evaluated by nearby loss, same-stream performance, domain accuracy, long-context performance, or reward. *B-level* evidence denotes bridge mechanisms such as internalization, parametric memory, context absorption, or self-adaptation: the evaluation suggests that information can be stored, compressed, or transformed inside model state, but does not yet establish sparse delayed user-facing behavior. *D-level* evidence denotes deployment-time behavioral learning: sparse post-deployment information changes later behavior under recall, paraphrase, delay, locality, conflict, or action-use tests after the original support context is unavailable. Figure 2 summarizes the ladder.

Operationally, we assign S when the primary evidence is same-stream, domain, or task-adaptation performance; B when the evaluation tests memory-like internalization, parametric storage, context absorption, or self-adaptation without fully testing sparse delayed user behavior; and D only when the evaluation directly probes later behavior from sparse post-deployment information after context removal. B-level is intentionally heterogeneous: it covers mechanisms that bridge proxy adaptation and deployment memory, but these mechanisms do not by themselves establish D-level memory.

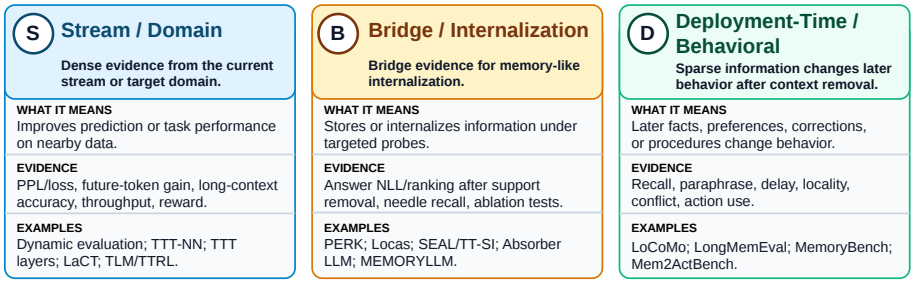

Figure 2: Three claim levels used to calibrate TTT claims. The level records the strongest evidence directly supported by the reported evaluation, not the broadest motivation or downstream narrative.

Audit protocol. We screened more than 40 targeted-search candidates available through April 2026 using bibliography seeding and targeted queries around LLM TTT/TTL, long-context TTT, context memory, parametric memory, personalization, continual learning, and assistant memory. A record was included if its title, abstract, introduction, or motivation explicitly invoked test-time learning or training, memory, persistence, context absorption, self-improvement, personalization, continual learning, or assistant memory. We coded 24 papers and treated the remaining records as background or exclusions, including retrieval-only or external-memory baselines, model-editing and safety background, non-LLM TTT, and architecture-adjacent context memory.

We therefore code claims by calibrating them to evidence rather than by inferring intent, assigning each paper the strongest claim level directly supported by its reported evaluations instead of the

level suggested by its broadest motivating statement. Assistant-memory benchmarks are labeled as D targets rather than D-level TTT evidence because they define the behavioral task family but do not themselves show that parametric TTT achieves it. This audit is not a prevalence estimate or systematic review. Appendix C gives the search fields, candidate accounting, background and exclusion summary, and boundary-case sensitivity worksheet.

The audit asks what each paper family's evidence directly supports, and what additional tests would be needed before using it in a deployment-memory claim. Table 2 gives representative examples.

Table 2: Claim-calibration exemplars. The level column records the strongest claim level directly supported by the reported evaluation, not the broadest motivation or downstream narrative. Full paper-level coding and boundary cases are in Appendix C.

Paper family	**Level**	**Evidence directly supported**	**Needed before D-level deployment-memory use**
TTT layers / LaCT / In-Place TTT [25, 35, 9]	S	Test-time state improves stream, domain, or long-context modeling under dense evidence	Sparse fact or preference behavior after context removal, paraphrase, delay, and locality probes
PERK / Locas / Absorber LLM [7, 17, 37]	B	Parameter-efficient memories, local parametric support, or context absorption provide bridge evidence	One-shot facts, preferences, corrections, conflict overwrite, locality, and matched retrieval baselines
SEAL / TT-SI / TTT-Discover [41, 1, 34]	B	Self-adaptation, agent improvement, or reward-backed discovery under generated or verifiable update data	Weak user evidence, user-specific retention, correction, and reward-poor deployment settings
Assistant-memory benchmarks [31, 2, 24]	D target	Direct behavioral tasks for long-term assistant memory, corrections, personalization, or actions	Matched TTT update budget, explicit-memory baseline, and no-context ablations before using benchmark success to justify parametric TTT claims

Audit pattern. The main pattern is not that TTT lacks progress; it is that most reported evidence remains below D-level deployment behavior. Dynamic evaluation, TTT-NN, TTT layers, LaCT, In-Place TTT, TTT-E2E, TLM, SyTTA, and TTRL primarily provide S-level evidence: adaptation to recent text, target-domain data, or verifiable tasks. PERK, Locas, SEAL, TT-SI, TTT-Discover, MEMORYLLM, and Absorber LLM move toward B-level evidence by studying parametric memories, self-adaptation, context internalization, or reward-backed discovery. Reasonable reclassifications of S/B boundary cases may change family counts, but they do not change the decision rule. A D-level deployment claim requires behavioral tests showing that a sparse user fact, preference, correction, or procedure changes later behavior after paraphrase, delay, conflict, and removal of the original context.

Takeaway: S- and B-level evidence can motivate deployment-memory hypotheses, but D-level language requires behavioral tests for sparse information after context removal, paraphrase, delay, locality, and conflict.

5 A Sanity-Check Counterexample

We use a narrow diagnostic experiment to test a common inference behind evidence migration. That is we test if improved support loss or answer likelihood is sufficient evidence for deployment recall. A test-time update is applied to a sparse factual support sentence, the support context is then removed, and the model is later queried for the injected fact. This setup demonstrated a single case where proxy metrics improve but later behavior does not is enough to show that proxy evidence and deployment recall should be reported separately.

Diagnostic setup. We use Qwen3 models at three scales, `Qwen/Qwen3-1.7B`, `Qwen/Qwen3-4B`, and `Qwen/Qwen3-8B`, with LoRA adaptation and a minimal online update on each support text. The factual setting introduces nonce access-code facts and then tests direct, paraphrased, and delayed recall after the support context is removed. We define $\Delta\mathrm{NLL} = \mathrm{NLL}_{\mathrm{after}} - \mathrm{NLL}_{\mathrm{before}}$, so negative values indicate improved teacher-forced loss after the update. Full hyperparameters, prompt counts,

scoring rules, explicit-memory baselines, conflict-overwrite tests, stress updates, and robustness checks are in Appendices B and D.

Evidence levels. The diagnostic keeps proxy, bridge, and target behavioral evidence separate. *Proxy* evidence consists of support reconstruction and local ΔNLL. *Bridge* evidence consists of answer ΔNLL. This metric tests whether probability mass moves toward the target answer, but not whether the model will produce that answer in open-ended use. *Target behavior* in the main factual probe is generated success under direct, paraphrased, and delayed prompting after the support context is removed. Appendix D adds locality, conflict-overwrite, retrieval, and replacement-memory checks as robustness and comparison evidence. This separation suggests lower loss is informative, but delayed free-form recall must be measured behaviorally.

Table 3: Loss improves, but generated free-form recall does not appear under the fixed greedy decoding protocol. Values are percentages except ΔNLL. Lower ΔNLL indicates improved teacher-forced loss after the update. Generated success is measured over 48 factual probes per model using normalized first-answer-line containment of the target answer.

	Loss improvement ΔNLL $\downarrow$			Greedy generated success (%) $\uparrow$		
Model	Support	Direct answer	Paraphrased answer	Direct	Paraphrased	Delayed
Qwen3-1.7B	-1.151 ± 0.051	-0.674 ± 0.085	-0.714 ± 0.071	0.0	0.0	0.0
Qwen3-4B	-1.218 ± 0.093	-0.529 ± 0.081	-0.531 ± 0.071	0.0	0.0	0.0
Qwen3-8B	-0.966 ± 0.026	-0.504 ± 0.079	-0.463 ± 0.070	0.0	0.0	0.0

Results. Across all three Qwen3 sizes, one-step LoRA updates improve support and answer loss, while generated free-form recall remains at 0.0% for direct, paraphrased, and delayed queries under the fixed greedy decoding and normalized-containment scoring protocol. Table 3 shows this proxy/behavior split directly, support ΔNLL improves substantially, and answer ΔNLL also improves, yet these gains do not translate into successful greedy recovery of the injected facts.

The zero generated-recall result is not the only signal we report. The same table shows that teacher-forced answer likelihood improves, yet greedy generation still omits the target fact. Appendix D gives representative cases where the likelihood signal does not translate into generated recall.

Explicit-memory controls. Matched explicit-memory controls serve as evidence-usability and deployment-comparison checks. The appendix specifies exact-context, BM25-style retrieval, and replacement-memory protocols, but the factual aggregate control reported here is the easy BM25 condition, in the 1.7B seed-replication checks, BM25 hit and memory answering remain at $48/48$. Harder retrieval checks in Appendix D have Hit@1 $0/48$ under paraphrased-support decoys and $0/24$ under stale/current conflicts. Thus, the point is not that retrieval trivially solves deployment memory, but that parametric TTT should be compared to explicit memory under matched evidence, query, scoring, and budget constraints.

Stress update and conflict checks. A stronger update can produce generated recall, but it exposes the missing locality dimension. In the Qwen3-1.7B factual setting, the appendix stress check raises direct and delayed recall to 72.9% $(35/48)$ and paraphrased recall to 54.2% $(26/48)$, while locality falls from 97.9% $(141/144)$ to 9.7% $(14/144)$ (Table A16). Conflict-overwrite checks show a different failure mode: the one-step LoRA update returns neither stale nor corrected code in all $72/72$ conflicts, while replacement memory is corrected-only in $68/72$ cases and both-corrected-and-stale in $4/72$ (Table A15). These checks show why D-level evidence should jointly report recall, paraphrase robustness, delay, locality or interference, conflict behavior, and matched baselines.

Robustness scope. Appendix D reports additional checks, the 1.7B seed replications keep direct/paraphrased/delayed recall at $0/48$, preference/correction and procedure mini-tasks also improve proxy loss while leaving greedy behavior at $0/24$, and the continuous-text check is reported only as proxy-regime context. We therefore use these checks to qualify the diagnostic, not to claim that the tested one-step LoRA update achieves deployment memory.

Takeaway: The diagnostic separates proxy improvement from deployment recall. Proxy and bridge gains can coexist with zero behavioral recall, and stronger recall can expose severe locality cost. D-level claims, therefore, require generated recall under paraphrase and delay, locality, or conflict reporting, and matched baselines.

6 Our Recommendations

For TTT papers that invoke post-deployment learning, memory, personalization, or self-updating assistants, our central recommendation is simple: claim language should track evidence level. Table 4 gives a claim-calibrated decision protocol. Stream-adaptation claims can headline stream loss, future-token loss, long-context accuracy, throughput, or reward. Deployment-memory claims should headline later behavior under a deployment-like update episode.

Table 4: Claim-calibrated decision protocol. Evidence can be positive and useful while still supporting only S- or B-level wording. D-level language requires behavioral improvement under the claimed use case, disclosed scoring, locality or failure reporting, and matched baseline comparison.

Evidence reported	Supported wording	Requires more evidence	Next evidence upgrade
Future-token loss, stream/domain metrics, or reward only	Improves stream adaptation, target-domain adaptation, or task performance under the stated objective	Remembers user facts, learns preferences, or performs post-deployment continual learning	Reset-vs-stream controls and direct behavioral probes for the claimed deployment target
Answer NLL / ranking, needle recall, context ablation	Provides bridge evidence for compression or internalization under the tested protocol	Robust deployment memory, personalization, or self-updating assistant behavior	Direct generation, paraphrase, delay, and no-support-context evaluation
Direct generated recall only	Immediate write-in under a narrow prompt format	Persistent or robust memory, broad personalization, or safe correction	Paraphrased and delayed recall, locality, conflict categories
Direct + paraphrase + delay + locality + conflict + matched baseline	Deployment-memory claim under the stated setting and budget	Broad general memory beyond the tested setting, user population, or update mechanism	Broader tasks, stronger baselines, more seeds/models, and stated deployment constraints

Match probes to claims. D-level evaluation should be claim-specific rather than maximalist. Factual memory requires no-context recall under paraphrase and delay; preference personalization requires later choices or response changes under plausible alternatives; correction claims require mutually exclusive stale/current scoring; procedural claims require later action or task transfer; and agent-memory claims require learned information to affect routing, tool calls, or arguments. Appendix B gives operational templates and evidence tiers. Such evaluations can be small: the support episode should introduce a sparse item absent from the evaluation prompt, and the later probe should test the failure modes relevant to the claim. When possible, reset-vs-stream controls should separate one-shot write-in from accumulated stream adaptation.

Keep proxy metrics separate. Support reconstruction, future-token loss, answer NLL, and, when reported, candidate ranking should remain in the report because they explain what the update is doing. But they should be presented as mechanism-level or intermediate signals, not as substitutes for the behavioral target. In particular, answer likelihood and generated answer success should be reported separately: the diagnostic above shows that a model can improve the likelihood of the target answer while still producing the same wrong free-form response.

Use matched baselines. A no-update model tests whether the update helps at all. Exact-context or long-context prompting tests whether the support evidence is sufficient when explicit. Retrieval or memory baselines test whether the same evidence can be stored outside the weights and reused under the same query and scoring rule. A *matched* baseline should use the same support information, query set, scoring rule, and disclosed budgets, including update tokens, trainable parameters, optimization steps, latency, memory footprint, context length, and retrieval index size when applicable.

Parametric TTT need not beat every explicit-memory baseline on every axis to be useful. Its value may appear under privacy, compression, latency, amortization, offline-operation, or context-pressure constraints. But those constraints should be stated and evaluated directly; otherwise, a memory or personalization claim risks conflating mechanism-level adaptation with deployment-time usefulness.

Takeaway: Claim language should track evidence level. Proxy and bridge metrics explain mechanisms, but memory, personalization, and post-deployment learning claims require behavioral evidence under the claimed use case, with failure categories and matched baselines visible.

7 Alternative Views

TTT papers should be evaluated by their stated objectives. Many TTT papers study stream adaptation, long-context scaling, or task reward under the objective they optimize, without claiming to implement deployment-time memory. Those papers should be evaluated on their stated objectives. The interpretive issue arises when such results are later used to motivate memory, personalization, or self-updating assistants. At that point, S- and B-level evidence needs additional behavioral probes before it supports D-level language.

Perplexity may already be a behavioral metric. One response is that perplexity is already behavioral for language models, because next-token prediction is the model's core behavior. This is reasonable in dense-stream settings, where better prediction may be the desired outcome. Personalized assistants require a different kind of behavior: recalling sparse facts, applying corrections, resolving conflicts, or abstaining when information is missing. In that regime, lower loss is useful evidence about the update mechanism, but deployed memory still has to be shown in later responses.

External memory may be the right deployment substrate. Assistant memory may often be better implemented explicitly rather than parametrically. Systems such as MemoryBank and MemGPT store, retrieve, page, or curate information outside the base-model weights, and they are aimed directly at long-term interaction and multi-session use [38, 21]. This view strengthens rather than weakens the evaluation standard. If explicit memory is the practical baseline, then parametric TTT should be evaluated against it under matched evidence, query, scoring, and budget constraints.

Context compression is itself a valid memory goal. Methods need not target sparse user facts to be memory-relevant. PERK, Locas, MEMORYLLM, and related systems study adapters, parametric memories, or persistent memory pools that encode context for later use [7, 17, 30]. We treat this as a legitimate B-level objective. The calibration point is narrower: compression or internalization results should be described as such unless they also show deployment behavior under paraphrase, delay, conflict, and locality.

Memory can be procedural rather than episodic. Test-time learning may be most useful for reusable strategies, code snippets, search heuristics, or reward-backed reasoning rather than user-fact recall. Dynamic Cheatsheet and TTRL are examples of this direction: they use accumulated solutions or reward-style signals to improve future problem solving [27, 40]. The same evidence standard applies, but the behavioral target changes. A procedural-memory claim should be tested through later task performance, transfer, and failure modes for the learned procedure, rather than through factual recall alone.

Stronger update mechanisms may succeed. Stronger TTT systems may pass behavioral tests that simple update mechanisms fail. Better objectives, routing, replay, constrained updates, verification, or hybrid parametric–external memory could produce useful deployment-time memory. Our position respects this possibility, and specifies what evidence would be needed to establish it: behavioral success under the claimed use case, together with update cost, locality, conflict, and matched baselines.

8 Conclusion

TTT has made real progress on long-context modeling, stream adaptation, domain adaptation, and reward-backed test-time improvement. This paper argues for a reviewable norm: TTT papers should headline behavioral memory evidence only when the claimed deployment behavior is directly tested. Lower perplexity, future-token loss, answer likelihood, candidate ranking when explicitly reported, and reward under a stated task are valuable evidence for the objectives they measure, but they should not be treated as sufficient evidence for deployment-time memory.

Showing that a model has acquired a user fact, preference, correction, or procedure for later use requires tests that resemble the claimed use case: direct, paraphrased, and delayed behavior; locality and conflict reporting; disclosed update budgets; mutually exclusive failure categories; and matched retrieval or long-context baselines. The aim is to keep claim and evidence at the same level. If future TTT systems can provide memory, the decisive evidence will appear where memory matters: in later behavior, under realistic constraints, with the interference cost visible.

References

[1] Emre Can Acikgoz, Cheng Qian, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. Self-improving llm agents at test-time. 2025.

[2] Qingyao Ai, Yichen Tang, Changyue Wang, Jianming Long, Weihang Su, and Yiqun LIU. Memorybench: A benchmark for memory and continual learning in LLM systems, 2026.

[3] Simone Antonelli, Mohammad Sadegh Akhondzadeh, and Aleksandar Bojchevski. Test-time training undermines existing safety guardrails. In *ICLR 2026 Workshop on Trustworthy AI*, 2026.

[4] Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Sai Surya Duvvuri, Fnu Devvrit, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Kale, and Samy Jelassi. Let's (not) just put things in context: Test-time training for long-context LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

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

[7] Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[8] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

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

[10] Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15202–15232, 2024.

[11] Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[12] Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. Detecting edit failures in large language models: An improved specificity benchmark. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11548–11559, Toronto, Canada, July 2023. Association for Computational Linguistics.

[13] Jinwu Hu, Zitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-time learning for large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[14] Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

[16] Qi Li, Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Xinglin Pan, and Xiaowen Chu. Should we really edit language models? on the evaluation of edited language models. *Advances in Neural Information Processing Systems*, 37:30850–30885, 2024.

[17] Sidi Lu, Zhenwen Liang, Dongyang Ma, Yan Wang, Haitao Mi, and Dong Yu. Locas: Your models are principled initializers of locally-supported parametric memories. *arXiv preprint arXiv:2602.05085*, 2026.

[18] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, 2024.

[19] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[20] Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[22] Amal Rannen-Triki, Jorg Bornschein, Razvan Pascanu, Marcus Hutter, Andras György, Alexandre Galashov, Yee Whye Teh, and Michalis K. Titsias. Revisiting dynamic evaluation: Online adaptation for large language models. *arXiv preprint arXiv:2403.01518*, 2024.

[23] Domenic Rosati, Robie Gonzales, Jinkun Chen, Xuemin Yu, Yahya Kayani, Frank Rudzicz, and Hassan Sajjad. Long-form evaluation of model editing. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3749–3780, 2024.

[24] Yiting Shen, Kun Li, Wei Zhou, and Songlin Hu. Mem2actbench: A benchmark for evaluating long-term memory utilization in task-oriented autonomous agents, 2026.

[25] 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 *Forty-second International Conference on Machine Learning*, 2025.

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

[27] Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In Vera Demberg, Kentaro Inui, and Lluís Marquez, editors, *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7080–7106, Rabat, Morocco, March 2026. Association for Computational Linguistics.

[28] 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, 2025.

[29] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[30] Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. MEMORYLLM: Towards self-updatable large language models. In *Forty-first International Conference on Machine Learning*, 2024.

[31] Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025.

[32] Yijie Xu, Huizai Yao, Zhiyu Guo, Weiyu Guo, Pengteng Li, Aiwei Liu, Xuming Hu, and Hui Xiong. You only need 4 extra tokens: Synergistic test-time adaptation for LLMs, 2026.

[33] Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Qi Cao, Dawei Yin, Huawei Shen, and Xueqi Cheng. The mirage of model editing: Revisiting evaluation in the wild. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15336–15354, Vienna, Austria, July 2025. Association for Computational Linguistics.

[34] Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time, 2026.

[35] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right. In *The Fourteenth International Conference on Learning Representations*, 2026.

[36] Weizhi Zhang, Xiaokai Wei, Wei-Chieh Huang, Zheng Hui, Chen Wang, Michelle Gong, and Philip S Yu. Memorycd: Benchmarking long-context user memory of llm agents for lifelong cross-domain personalization. *arXiv preprint arXiv:2603.25973*, 2026.

[37] Zhixin Zhang, Shabo Zhang, Chengcan Wu, Zeming Wei, and Meng Sun. Absorber llm: Harnessing causal synchronization for test-time training. *arXiv preprint arXiv:2604.20915*, 2026.

[38] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 19724–19731, 2024.

[39] Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[40] Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[41] Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

A Evidence migration patterns

This appendix makes the evidence-migration concern concrete without treating the cited papers as overclaiming. The migration risk is interpretive: a result that is well scoped as S- or B-level evidence can become misleading when reused as support for a D-level deployment-memory narrative.

Table A1: Concrete evidence-migration patterns.

Migration pattern	Grounded evidence	Tempting stronger narrative	Missing behavioral evidence
Dense-stream TTT → online user memory	Lower future-token loss or better long-context performance after updates on dense nearby text.	The model can learn new user facts online.	Sparse fact write-in, no-support-context recall, paraphrase, delay, and locality.
Context compression/internalization → persistent assistant memory	Needle recall, context ablation, parameter-efficient context state, or local parametric support.	The compressed state functions as durable assistant memory.	Cross-session or delayed behavior, conflict overwrite, unrelated-behavior preservation, and matched explicit memory.
Self-adaptation/generated updates → personalized continual learning	Task improvement after generated update data or self-improvement episodes.	The assistant can personalize from weak user evidence.	User-specific preference or correction probes, retention, overgeneralization checks, and safety-relevant conflict scoring.
Reward-backed discovery → broad deployment self-improvement	Test-time improvement in settings with verifiable rewards or many scored attempts.	The system can self-improve broadly after deployment.	Reward-poor user facts, preferences, procedures, held-out transfer, and failure categories for overfitting or reward hacking.

B Detailed evaluation agenda and limitations

Purpose of this appendix. This appendix expands the reviewer-facing protocol in Table 4. The goal is not to require every TTT paper to run every possible memory test. Rather, the evaluation should match the claim. A paper claiming stream adaptation can headline stream loss, future-token loss, long-context accuracy, throughput, or reward. A paper claiming deployment-time memory, personalization, correction, or agent memory should additionally report later behavior under a deployment-like update episode after the original support context is unavailable.

Diagnostic role of the pilot. The diagnostic in Section 5 partially follows this agenda. It specifies exact-context controls, reports BM25-style retrieval and replacement-memory checks, separates answer-likelihood and generated-answer reporting, discloses update-time and token-budget details, and adds a same-family Qwen3 scale axis, a 1.7B three-seed replication, a stress update, mutually exclusive conflict scoring, and small preference/correction and procedure/action probes. Full benchmark construction and representative method comparisons are left for future work.

Diagnostic role: The LoRA/Qwen pilot isolates one evidential question: whether support-loss and answer-likelihood gains are sufficient for generated deployment behavior. The result shows that these proxy and bridge metrics can improve without free-form recall, motivating separate reporting of mechanism-level and behavioral evidence.

Minimal D-level evidence by claim type. D-level evidence is claim-specific. The required behavioral bundle changes with the claim, but in every case the learned information should affect later behavior after the original support context is removed. Table A2 gives minimum evidence shapes for common deployment-memory claims.

Table A2: Minimal D-level evidence by deployment claim type. The required bundle changes with the claim, but in every case the later behavior must match the claimed use case after the original support context is removed.

Claim type	Minimal D-level behavioral evidence	Required failure categories	Matched baseline
Factual user memory	no-support-context answer; direct and paraphrased query; delayed query	correct / wrong / abstain; locality	exact context; retrieval memory
Preference personalization	later choice or response-style change under plausible alternatives	follows preference / ignores / overgeneralizes / unsafe conflict	profile prompt; retrieval profile
Correction / overwrite	corrected-only response after stale fact is contradicted	corrected-only / stale-only / both / neither	replacement memory with stale item removed
Procedure learning	held-out task where the learned rule changes action, not just string recall	correct transfer / memorized surface form / unrelated degradation	explicit instruction; tool memory
Agent/tool memory	learned fact changes tool call, route, or argument	correct action / wrong action / leakage / stale action	memory-augmented agent
Reward-backed discovery	later held-out reward or task gain under the same discovery budget	solved / overfit / reward hacking / unrelated damage	no-update; search or retrieval baseline

Operational behavioral templates. The same claim-specific idea can be made more operational by specifying the support episode, the no-context query, paraphrase or delay condition, locality or conflict check, and matched baseline. Table A3 gives templates for common deployment-memory settings. These are intended as minimal behavioral shapes, not as a universal benchmark.

Table A3: Claim-specific behavioral templates.

Claim	Support episode	No-context query	Paraphrase / delay	Locality / conflict	Matched baseline
Factual memory	One sparse user fact absent from the pre-update prompt.	Ask for the fact after removing support context.	Direct and semantic paraphrase; at least one intervening task or session break.	Unrelated facts or abilities should remain stable.	Exact context; retrieval memory.
Preference personalization	User states a stable preference under plausible alternatives.	Later choice or response style should reflect the preference.	Query with different surface form and scenario.	Check overgeneralization, unsafe preference conflicts, and unrelated preferences.	Profile prompt; retrieved profile.
Correction / overwrite	A stale fact is explicitly contradicted by a current correction.	Ask for the current answer without showing either support item.	Query correction with rewording or after unrelated turns.	Score corrected-only, stale-only, both, and neither.	Replacement memory with stale item removed.
Procedure learning	User gives a rule or procedure that changes task execution.	Held-out task requires applying the learned rule.	Rephrase the procedure or change surface variables.	Check unrelated procedures and mistaken transfer.	Explicit instruction; tool memory.
Agent/tool memory	User fact or rule should affect routing, tool choice, or arguments.	Later task requires the remembered item for action selection.	Change wording and separate the action from the support turn.	Score wrong tool, stale action, leakage, and unrelated route changes.	Memory-augmented agent.

Direct and paraphrased retrieval. A deployment-time update should be tested on direct questions and paraphrases that require extracting the newly introduced fact, preference, correction, or procedure after the support context is removed. Teacher-forced answer likelihood and free generation should be reported separately, because the gap between them is itself informative. A system can move probability mass toward the correct answer under teacher forcing while still failing to produce the answer under open-ended generation.

Retention, locality, and conflict. Updated knowledge should be queried after unrelated intervening turns or tasks. Evaluations should also include locality probes and conflict-resolution tests, because a useful update should not indiscriminately distort unrelated behavior or preserve stale information after a correction. If the claim concerns procedures or agents, the query should require the updated information to select a rule, route, tool, or action rather than merely repeat a string.

Matched explicit-memory baselines. If the intended use case is a personalized assistant or agent, prompt accumulation, long-context inference, retrieval-augmented memory, and other non-parametric memory systems are natural baselines. These baselines should use matched evidence and report context cost, latency, update budget, and failure categories, because a parametric update is most compelling when it improves behavior under constraints where explicit memory is costly or unavailable. Stream adaptation should also be separated from one-shot write-in through reset-vs-stream controls, since cumulative gains across a stream provide different evidence from learning from a single support item.

Table A4: Matched explicit-memory baseline protocol.

Baseline	What it controls	Matching requirement
No update	Whether the update helps relative to the base model or unmodified system.	Same query set, generation budget, scoring rule, and random seed policy when applicable.
Exact context	Whether the support evidence is sufficient when directly visible.	Same support item, query, scoring rule, and generation budget as the parametric-update condition.
Long-context prompting	Whether keeping the support evidence in context solves the task.	Same support information, same downstream query, disclosed context length, and comparable generation budget.
Retrieval memory	Whether the evidence can be stored outside weights and retrieved for the same task.	Same memory item, query set, answer scoring, retrieval index disclosure, and context/latency budget.
Replacement memory	Whether a correction can remove stale information before answering.	Same stale/current evidence, mutually exclusive corrected/stale/both/neither categories, and current-only memory state.
Memory-augmented agent	Whether learned information changes a tool call, route, or argument through explicit state.	Same support episode, tool set, action space, cost budget, and leakage/stale-action scoring.

In this paper, the easy BM25-style factual condition is a usability control: it verifies that the sparse support facts are answerable when explicit memory retrieves the intended sentence. The harder paraphrase and stale/current retrieval checks are failure-mode probes, showing why stronger explicit-memory baselines should include recency handling, semantic retrieval, reranking, or conflict resolution rather than naive lexical matching alone.

Evidence tiers for deployment-memory language. The strength of the claim should track the strength of the behavioral evidence. Table A5 gives three rough tiers. The tiers are not meant to define a benchmark leaderboard; they are a wording guide for authors and reviewers.

Table A5: Evidence tiers for deployment-memory language.

Tier	Appropriate evidence and wording
Minimum acceptable	A small no-support-context behavioral probe matched to the claim, with direct/paraphrased queries, disclosed scoring, and a no-update or exact-context control. Supports narrow deployment-memory wording under the tested setting.
Strong evidence	Adds delay, locality, conflict categories when relevant, multiple seeds or model sizes, and matched explicit-memory baselines. Supports robust claim wording within the tested update budget.
Deployment-grade	Adds realistic multi-session data, deletion/forgetting or user-isolation tests when personal data is stored, latency/privacy/context-cost accounting, and stronger external-memory comparators. Supports deployment-oriented system claims.

Wording examples. The decision protocol is meant to be operational. Table A6 gives examples of wording that stays within the evidence level and wording that calls for D-level behavioral evidence.

Table A6: Claim wording examples.

Better wording	Too strong unless D-level evidence exists
Improves future-token prediction after online updates	Learns user preferences at deployment time
Improves domain or task performance under the stated objective	Performs post-deployment continual learning
Compresses dense context into a test-time state	Forms persistent assistant memory
Improves answer likelihood or candidate rank for support facts	Can recall newly learned facts in deployment
Shows bridge evidence for internalization under this protocol	Enables self-updating assistant behavior

Claim discipline. Papers should title and frame their contributions at the level their evidence supports. Continuous-stream adaptation, target-domain TTL, formal test-time discovery, context compression, and deployment-time learning are all legitimate targets. Evidence from one regime should not inherit the behavioral expectations of another. In particular, proxy and bridge metrics should remain in the report, but they should be described as mechanism-level or intermediate evidence unless the later behavior required by the claim is directly tested.

Additional limitations. This diagnostic pass adds a same-family Qwen3 size axis, hard-negative retrieval controls, a 1.7B three-seed replication, conflict scoring, and small preference/correction and procedure/action probes. It leaves several directions open: cross-family model comparisons, a second parametric update mechanism, representative TTT/TTL systems, richer procedure-following tasks, and multi-seed runs for every model size. The confidence intervals are across examples unless explicitly marked as seed replication. Reward-rich settings such as math, code, kernels, and scientific optimization remain an important frontier. These limitations reinforce the central recommendation: specify the evaluation target, disclose the scoring rule, report failure categories, and avoid compressing all evidence into one loss number.

C Paper-level audit sheet

This appendix expands the claim-calibrated audit summarized in the main text. The level column records the strongest claim directly supported by the reported evidence as summarized in the paper, while the last column records the shortest behavioral test that would be needed before using the result as D-level deployment-time learning evidence.

Audit protocol. We screened papers available through April 2026 from the paper's bibliography plus targeted searches using terms such as *LLM test-time training*, *test-time learning*, *long-context TTT*, *context memory*, *self-adapting language models*, *parametric memory*, *personalization*, *continual learning*, and *assistant memory*. A paper enters the audit when its title, abstract, introduction, or motivation explicitly invokes test-time learning/training, memory, persistence, context absorption, self-improvement, personalization, continual learning, or assistant memory. We screened 41 candidate records: 24 are coded below, 5 retrieval-only or external-memory systems are retained as background baselines, 9 model-editing/safety/background papers are cited for context but excluded from claim-level coding, and 3 non-LLM or architecture-adjacent papers are excluded from the audit table. This is single-author coding with sensitivity recoding, intended as a position-paper calibration audit rather than a systematic review, prevalence estimate, or inter-coder reliability study. To make disagreements inspectable, we provide paper-level source phrases, boundary-case coding rationales, and sensitivity effects rather than asking readers to trust aggregate counts.

S-level coding means the reported evidence directly supports stream/domain/task adaptation; B-level coding means it supports a bridge mechanism such as internalization, compression, parametric memory, or self-adaptation; D-level coding means it directly tests sparse deployment information after paraphrase, delay, conflict, locality, or action use. Borderline cases are coded to the strongest level directly supported by the reported evaluation, not by the broadest motivation sentence. Eight papers are treated as boundary cases for sensitivity: TTT layers, LaCT, Not Just Context, TTT-E2E, PERK, Locas, MEMORYLLM, and Absorber LLM. Reclassifying these S/B cases changes the counts without changing the paper's decision rule, because none supplies the full D-level behavioral bundle for sparse user deployment claims.

Table A7: Audit field summary.

Field	Definition
Search source	Bibliography seeding plus targeted searches through April 2026.
Search terms	LLM test-time training, test-time learning, long-context TTT, context memory, self-adapting language models, parametric memory, personalization, continual learning, and assistant memory.
Inclusion rule	Title, abstract, introduction, or motivation invokes TTT/TTL, memory, persistence, context absorption, self-improvement, personalization, continual learning, or assistant memory.
Exclusion rule	Retrieval-only/external-memory systems, model-editing and safety background, non-LLM TTT, and architecture-adjacent memory papers are retained as context but not coded as LLM TTT evidence.
Coding target	Strongest claim level directly supported by reported evaluations, not by the broadest motivation sentence.

Table A8: Candidate accounting.

Category	Count	Treatment
Coded stream, long-context, TTL, or reward-rich task adaptation	10	Claim-level coded as S or S/B.
Coded bridge/internalization or parametric-memory evidence	8	Claim-level coded as B.
Coded D-level assistant-memory target benchmarks	6	Claim-level coded as D target behavior.
External-memory baselines	5	Retained as matched-baseline context, not LLM TTT claim-coded.
Model-editing and safety background	9	Retained for evaluation lessons and risk framing.
Non-LLM or architecture-adjacent exclusions	3	Retained only as boundary context.
Total screened	41	24 coded; 17 background or excluded from claim-level coding.

Boundary-case coding worksheet. We provide the following worksheet for inspecting the eight boundary cases used in the audit sensitivity discussion.

Table A9: Boundary-case coding worksheet.

Paper	Level	Boundary reason	Sensitivity effect
TTT layers	S/B	Fast weights improve sequence and long-context behavior; sparse delayed deployment memory remains untested	Recoding as S or B changes counts only; D-level still unsupported.
LaCT	S/B	Large-chunk TTT improves efficiency and long-context performance; D-level sparse behavior remains to be tested	Recoding as S or B leaves required sparse behavior unchanged.
Not just context	S/B	Context-specific updates are close to bridge evidence; matched delayed deployment behavior is missing	Recoding as S or B does not license personalization language.
TTT-E2E	B	Long-context compression and needle-style recall provide bridge evidence for deployment-memory evaluation	Treating as S/B would still require sparse semantic write-in.
PERK	B	Parameter-efficient test-time learning suggests internalization; sparse preference/correction behavior is missing	Boundary remains below D until user-facing behavior is tested.
Locas	B	Parametric memory framing gives bridge evidence; D-level conflict/locality behavior remains incomplete	Stronger memory framing raises, rather than removes, the D-level burden.
MEMORYLLM	B	Persistent memory pools support memory-like behavior and call for matched parametric TTT/no-context ablation for TTT claims	Counts could move toward D target, but ordinary parametric TTT claims still need ablation.
Absorber LLM	B	Context absorption and synchronization are bridge evidence; sparse user write-in with behavioral ablation is missing	Recoding as B or S/B leaves no-context deployment behavior untested.

Background and exclusion records. The following records were screened because they are useful comparators, cautionary evaluation background, or boundary cases, but they are not coded as direct LLM TTT evidence in Tables A11–A13.

Table A10: Background and exclusion records.

Record	Role	Evidence retained	Reason not claim-coded as LLM TTT
MemoryBank [38]	External memory	Long-term memory storage and retrieval.	Baseline context, not parametric TTT evidence.
LongMem [29]	External memory	Retrieval/cache memory for long context.	Baseline context, not TTT update evidence.
MemGPT [21]	External memory	Paged memory and context management.	Baseline context, not parametric TTT evidence.
Mem0 [8]	External memory	Scalable explicit memory system.	Baseline context for production memory.
Dynamic Cheatsheet [27]	External memory	Adaptive memory for reusable problem-solving snippets.	Explicit-memory comparator rather than ordinary weight update.
ROME [19]	Model editing	Efficacy, generalization, and specificity framing.	Background caution for evaluating state changes.
MEMIT [20]	Model editing	Mass editing and specificity.	Background caution for interference and scale.
Edit-failure benchmark [12]	Model editing	Specificity failures after edits.	Background evaluation lesson.
MQuAKE [39]	Model editing	Multi-hop editing evaluation.	Background evaluation lesson.
Long-form editing evaluation [23]	Model editing	Long-form behavior after edits.	Background evaluation lesson.
Editing at scale [10]	Model editing	Forgetting under many edits.	Background evaluation lesson.
Should we edit LMs? [16]	Model editing	Edit-necessity and evaluation critique.	Background evaluation lesson.
Mirage of model editing [33]	Model editing	In-the-wild edit evaluation failures.	Background evaluation lesson.
TTT safety guardrails [3]	Safety background	Safety impact of TTT.	Risk context rather than memory-claim evidence.
Original TTT [26]	Historical boundary	Self-supervised TTT for distribution shifts.	Non-LLM deployment-memory setting.
Titans [6]	Architecture-adjacent	Memory architecture for test-time memorization.	Boundary context, not LLM TTT claim coding.
ATLAS [5]	Architecture-adjacent	Context memorization architecture.	Boundary context, not LLM TTT claim coding.

Worked examples. *TTT-NN:* S pass, D no. It updates on nearest-neighbor text and reports local language-model improvement, so it supports adaptation to nearby retrieved evidence; deployment-memory use would require delayed no-context recall, locality, and matched memory baselines. *PERK/Locas:* B pass, D incomplete. They explicitly move toward parameter-efficient or locally supported memories, so their evidence is closer to internalization; D-level assistant-memory claims still need sparse user facts, preferences, and corrections under paraphrase, delay, conflict, and matched external-memory baselines. *LongMemEval/MemoryBench:* D target behavior yes. They evaluate multi-session or service-time memory behavior, so they instantiate the target capability; a TTT paper borrowing that motivation should report comparable behavior under a stated update budget. *This pilot:* proxy and bridge evidence yes, D behavior no for the tested one-step LoRA setup; matched retrieval and replacement-memory checks are reported, so the result supports the proxy-insufficiency point: proxy and bridge improvements should not be treated as deployment-memory evidence without matching behavioral success.

Table A11: Paper-level audit for stream and long-context TTT evidence.

Paper	Source phrase	Level	Coding rationale	Minimal evidence before D
Dynamic evaluation [15]	"Dynamic evaluation"	S	Updates on recent sequence context and evaluates sequence-model loss.	Delayed no-context recall and locality.
Revisiting dynamic evaluation [22]	"Online adaptation"	S	Studies online adaptation for LLMs with stream-style evidence.	Sparse user fact write-in after delay.
TTT-NN [11]	"Test-Time Training on Nearest Neighbors"	S	Adapts from retrieved neighbors and evaluates local LM improvement.	Paraphrased retrieval without neighbor context.
TTT layers [25]	"Learn at Test Time"	S/B	Fast-weight hidden states improve sequence modeling and long-context behavior.	Reset-vs-stream plus sparse delayed behavior.
LaCT [35]	"Test-Time Training Done Right"	S/B	Emphasizes efficient large-chunk TTT and long-context performance.	One-shot fact/preference retention and conflict.
In-Place TTT [9]	"In-Place Test-Time Training"	S	Aligns online updates with next-token prediction objectives.	Behavioral recall after removing support context.
Not just context [4]	"things in Context"	S/B	Tests context-specific updates for long-context LLMs.	Matched retrieval baseline plus delayed behavior.
TTT-E2E [28]	"End-to-End Test-Time Training for Long Context"	B	Targets long-context compression and needle-style recall.	Sparse semantic write-in under paraphrase and delay.

Table A12: Paper-level audit for bridge evidence toward memory-like claims.

Paper	Source phrase	Level	Coding rationale	Minimal evidence before D
PERK [7]	"Parameter-Efficient Test-Time Learning"	B	Uses parameter-efficient updates for long-context reasoning.	Sparse facts/preferences tested after delay.
Locas [17]	"Parametric Memories"	B	Frames models as initializers for locally supported memories.	Conflict overwrite and locality under user corrections.
MEMORYLLM [30]	"Self-Updatable"	B	Implements persistent memory pools for self-updating behavior.	Matched parametric TTT and no-context ablation.
TLM [13]	"Test-Time Learning"	S	Optimizes input likelihood at test time.	User-specific write-in and generated recall.
SyTTA [32]	"Test-Time Adaptation"	S	Uses label-free adaptation signals rather than new user memory.	Delayed paraphrase and locality tests.
TTRL [40]	"Test-Time Reinforcement Learning"	S	Improves tasks with reward-style test-time updates.	Reward-poor user facts and corrections.
SEAL [41]	"Self-Adapting Language Models"	B	Generates update data and reports self-adaptation gains.	Novel-information recall after context removal.
TT-SI [1]	"Self-Improving LLM Agents"	B	Studies self-improvement for agents at test time.	Personalization, conflict, and retention probes.
TTT-Discover [34]	"Discover at Test Time"	B	Uses verifiable rewards for active test-time discovery.	Weak-evidence assistant memory without rewards.
Absorber LLM [37]	"Test-Time Training"	B	Studies causal synchronization and context absorption.	Sparse user write-in with causal ablation.

Table A13: Paper-level audit for D-level assistant-memory targets.

Paper	Source phrase	Level	Coding rationale	Remaining gap for TTT claims
LoCoMo [18]	"Very Long-Term Conversational Memory"	D	Tests long-horizon conversational memory behavior.	Add matched parametric-update comparison.
LongMemEval [31]	"Long-Term Interactive Memory"	D	Evaluates memory in chat-assistant interactions.	Add TTT update budget and ablations.
MemoryAgentBench [14]	"Incremental Multi-Turn Interactions"	D	Tests memory through incremental agent interactions.	Compare against prompt/retrieval and parametric TTT.
MemoryBench [2]	"Memory and Continual Learning"	D	Targets service-time memory and continual learning.	Add explicit TTT method comparison if claimed.
MemoryCD [36]	"Lifelong Cross-Domain Personalization"	D	Tests long-context user memory for personalization.	Separate explicit memory from parametric update.
Mem2ActBench [24]	"Long-Term Memory Utilization"	D	Measures memory use in downstream agent actions.	Add matched TTT write-in and action ablation.

D Diagnostic robustness checks

Pilot experiment details. Unless otherwise noted, diagnostic runs use H100 GPUs, seed 13, LoRA rank 8, alpha 16, dropout 0.0, learning rate 5×10^{-4}, and a 24-token generation budget. The factual probes use 48 prompts per prompt type, the overwrite probes use 24 conflicts, and the locality probes use 144 unrelated prompts. We score generated answers as correct when the normalized first answer line contains the normalized target answer. We define $\Delta\mathrm{NLL} = \mathrm{NLL}_{\mathrm{after}} - \mathrm{NLL}_{\mathrm{before}}$, so negative deltas indicate improved teacher-forced loss after the update.

The matched external-memory baselines receive the same support evidence as the LoRA update. The *exact-context* baseline prepends the support sentence directly to the evaluation prompt. The *BM25-style retrieval* baseline builds a fixed memory bank whose retrieval unit is the support sentence. For each true fact, we add three hard negatives: the same object with a different prefix, the same prefix with a different object, and an answer-format distractor. We report both top-1 retrieval hit, counted as correct when the retrieved fact id matches the target id, and end-to-end answer success after prompting with `Retrieved memory: <support>` followed by the original query. For conflicts, the *replacement-memory* baseline discards the stale support, keeps only the correction sentence, and then answers the current-code query.

The supporting checks are deliberately narrow. On 1.7B only, we run a continuous-text proxy-regime check by splitting 32 passages into *support*, *future_1*, and *future_2*, updating on the support chunk, and comparing one-step, stream, and reset evaluation. We rerun the 1.7B factual/conflict core probes with two additional seeds. We also add small 24-item 1.7B preference/correction and procedure/action mini-tasks to test whether the proxy/behavior pattern is limited to nonce access-code strings.

Table A14: Robustness and stress checks for the 1.7B diagnostic setting.

Setting	Key result
Seeds 13 / 21 / 42	For the 1.7B model, the core pattern is unchanged across seeds: greedy direct/paraphrased/delayed recall remains 0.0% (0/48), BM25 hit remains 100.0% (48/48), memory answer remains 100.0% (48/48), LoRA conflict correction remains 0.0% (0/24), and locality ranges from 95.1%–97.9% (137–141/144).
Preference/correction mini-task	Preference updates improve proxy loss, with $\Delta\mathrm{NLL}(\mathrm{S/D/P}) = -1.524/-0.709/-0.629$, while greedy direct/paraphrased/delayed behavior remains 0.0% (0/24). Exact-context answers are 87.5%–91.7% (21–22/24), and replacement memory is corrected-only for most preference corrections.
Procedure mini-task	Procedure updates also improve proxy loss, with $\Delta\mathrm{NLL}(\mathrm{S/D/P}) = -1.219/-0.854/-0.785$, while greedy direct/paraphrased/delayed behavior remains 0.0% (0/24). Exact-context answers are 58.3% (14/24), 33.3% (8/24), and 16.7% (4/24) for the three procedure prompt types, and replacement memory resolves 87.5% (21/24) of conflicts.
Update-strength sweep	We use 1.7B LoRA sweep to cover rank-8 steps 1/4/16 and rank-32 steps 1/4. Rank-8 steps 1/4 have 0/48 direct/paraphrased/delayed recall with 141/144 and 135/144 locality; the rank-8 16-step stress point has 35/48 direct, 26/48 paraphrased, and 35/48 delayed recall with 14/144 locality. Rank-32 steps 1/4 also have 0/48 recall, with 143/144 and 131/144 locality.
Conflict categories	Mutually exclusive conflict scoring shows that one-step LoRA is neither-corrected-nor-stale in all 72 conflicts across Qwen3 1.7B/4B/8B. Replacement memory is corrected-only in 68/72 cases and both corrected-and-stale in 4/72 cases, all from the 1.7B run.
Continuous proxy check	For 1.7B, one-step support/F1/F2 deltas are -0.636 ± 0.042, -0.063 ± 0.012, and -0.058 ± 0.010, with stream F1 at -0.384 ± 0.059.
16-step stress update	As discussed in the main text, recall rises to 72.9% (35/48) on direct and delayed prompts and 54.2% (26/48) on paraphrased prompts, while locality drops sharply to 9.7% (14/144). We treat this setting as a locality stress test, not as a practical operating point.

Conflict-overwrite scoring. Correction claims require mutually exclusive categories. A response counts as successful only when it contains the corrected answer and excludes the stale answer. Responses containing both corrected and stale information are failures, even if a non-exclusive scorer would mark the corrected answer as present.

Table A15: Mutually exclusive conflict-overwrite categories. Counts are out of 24 conflicts per model. Corrected-only is the strict D-level success criterion; both corrected and stale is counted as failure.

Model	Method	Corrected-only	Both	Neither	Locality preserved
Qwen3-1.7B	LoRA update	0	0	24	68/72
Qwen3-1.7B	Replacement memory	20	4	0	–
Qwen3-4B	LoRA update	0	0	24	71/72
Qwen3-4B	Replacement memory	24	0	0	–
Qwen3-8B	LoRA update	0	0	24	72/72
Qwen3-8B	Replacement memory	24	0	0	–

The one-step LoRA update produces neither code in all conflict cases, indicating overwrite non-recall rather than successful correction. This is a failure of behavioral access, not merely a failure of conflict arbitration. Replacement memory is usually corrected-only, although the 1.7B model sometimes includes both corrected and stale codes in the same response.

Stress update tradeoff. A stronger update can produce generated recall, but recall alone is not sufficient for deployment-memory claims. Locality probes ask unrelated questions whose answers should remain unchanged after the update.

Table A16: Stress update tradeoff in the Qwen3-1.7B factual setting.

Setting	Direct Recall ↑	Paraphrased Recall ↑	Delayed Recall ↑	Locality ↑
1-step update	0/48	0/48	0/48	141/144
4-step update	0/48	0/48	0/48	135/144
16-step stress update	35/48	26/48	35/48	14/144

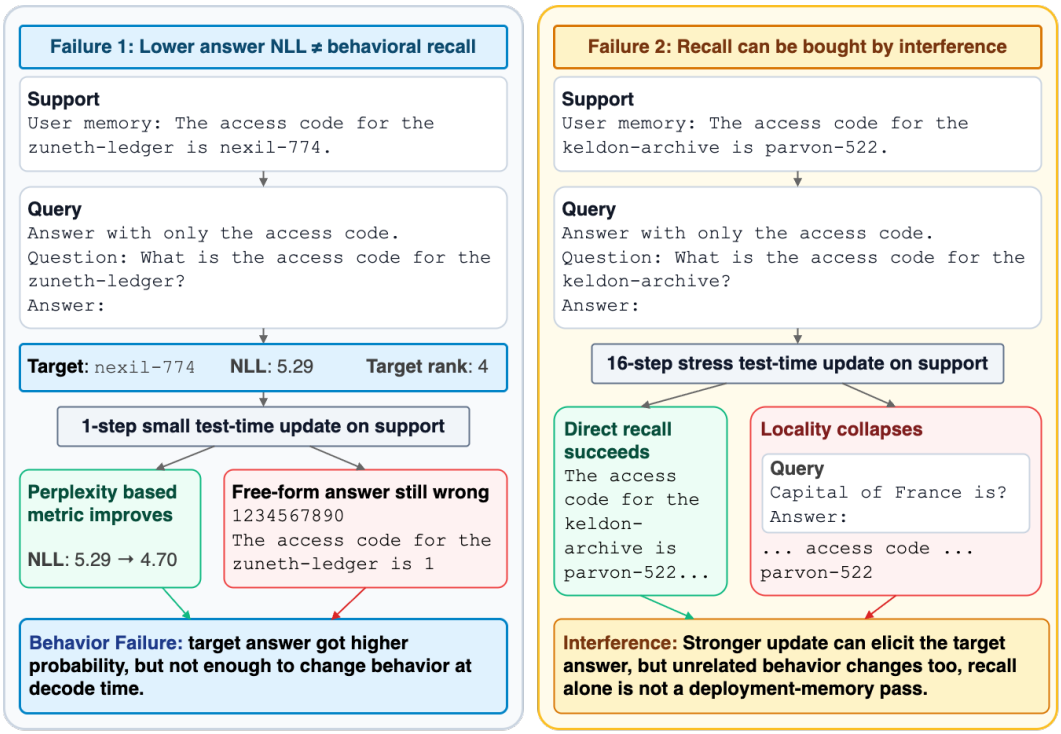

Figure A1: Qualitative diagnostic failure cases. Left: a one-step update lowers target-answer NLL without changing free-form generation. Right: a stronger stress update can elicit the target answer, but with severe locality interference on unrelated prompts.

The 16-step stress point raises direct and delayed recall to 72.9% and paraphrased recall to 54.2%, but locality falls from 97.9% to 9.7%. We treat this as evidence that stronger updates can move behavior, not as evidence that the update has achieved deployment memory. A D-level claim would need to report both the behavioral gain and the interference cost.

Retrieval hardness check. The original BM25-style factual condition is useful because it shows that the sparse evidence is answerable when explicit memory retrieves the intended sentence. To avoid making lexical retrieval look easier than it is, we add a no-generation lexical-retrieval stress check: one condition paraphrases the true memory snippets and adds same-subject backup-token distractors; the other puts stale and corrected snippets for the same subject in memory and asks for the current code. Table A17 reports retrieval hit rates and oracle top-1 answerability, with model generation left out of this check.

Table A17: Harder BM25 retrieval checks over the frozen fact set.

Condition	Queries	Hit@1	Hit@3	Interpretation
Paraphrased support + decoys	48	0/48	47/48	Top-1 is a same-subject backup-token decoy in all cases, even though the target is usually in the top three.
Same-subject stale/current	24	0/24	24/24	Top-1 is stale in all cases; the corrected snippet is recoverable in top three but requires recency/conflict-aware selection.

Qualitative diagnostic examples. The following examples make the failure mode inspectable. They are illustrative rather than a prevalence table, and show cases where answer likelihood improves while free-form behavior still does not change; the rank column is included only as case-level diagnostic context.

Table A18: Additional qualitative examples from the 1.7B factual diagnostic.

Fact	Support target	Proxy/rank movement	Generated answer	Interpretation
0	novari-archive → `lixume-235`	ΔNLL -0.56; rank $6 \to 6$	"1234567890..."	Lower target loss, unchanged wrong pattern.
1	keldon-archive → `parvon-522`	ΔNLL -0.81; rank $7 \to 7$	"1234567890..."	Larger proxy gain still does not change recall.
4	orvian-archive → `virex-188`	ΔNLL -0.40; rank $1 \to 1$	"1234567890..."	Case-level rank 1 still does not imply free generation.
6	zuneth-archive → `nexil-500`	ΔNLL -0.77; rank $3 \to 3$	"1234567890..."	Case-level top-3 rank is not behavioral recall.
7	belisar-archive → `sovem-657`	ΔNLL -1.39; rank $6 \to 6$	"1234567890..."	Strong NLL gain can coexist with the same wrong template.
30	zuneth-ledger → `nexil-774`	NLL $5.29 \to 4.70$; rank $4 \to 4$	"1234567890..."	Median-like proxy gain without generated recall.
stress 1	keldon-archive → `parvon-522`	16-step direct recall succeeds	"parvon-522..."	Stronger update can elicit recall while exposing locality cost.
conflict 0	novari-archive correction to `revok-247`	Replacement memory category: both	"revok-247, not lixume-235"	Non-exclusive scoring would overstate correction success.

E Scope and limitations

This paper is a position paper with a calibration audit and a narrow diagnostic experiment. The diagnostic shows that proxy improvement can be insufficient for generated deployment behavior; it does not map the full boundary of parametric deployment-time learning. The audit is structured and reproducible, but its purpose is claim calibration rather than prevalence estimation. These limitations reinforce the recommendation: specify the evaluation target, disclose the scoring rule, and report failure categories instead of compressing all evidence into one loss number.

F Deployment and Societal Risks

Deployment-time memory is not only a capability claim but also a governance claim. A system that stores user facts, preferences, corrections, or procedures through parametric updates may make information harder to inspect, delete, scope to a user, or audit than an explicit memory store. Prematurely validating such systems with proxy metrics could encourage deployment before consent, retention, deletion, cross-session isolation, and cross-user leakage are tested. Our behavioral standard therefore has an ethical dimension: D-level evidence should include not only recall and personalization success, but also locality, conflict handling, deletion or forgetting when relevant, and clear disclosure of where information is stored. Explicit-memory baselines are important partly because they often provide clearer audit and deletion surfaces. Parametric TTT methods that claim deployment memory should state what privacy, latency, compression, or offline-operation constraints justify storing information in model state.

Table A19: Governance-oriented behavioral tests for deployment-memory claims over user data.

Governance concern	**Behavioral test**	**When required**
Deletion / forgetting	Request deletion or expiry, then test that the fact, preference, or correction is no longer used while unrelated behavior remains intact.	Claims that user information can be removed, expired, or policy-scoped.
Cross-user isolation	Update on one user's information, then query under another user identity or session and score leakage.	Claims involving multi-user deployment or shared model state.
Stale/current conflict	Present stale information and a later correction, then require corrected-only behavior under mutually exclusive scoring.	Claims about corrections, updates, or self-updating assistants.
Consent and scope	Mark information as in-scope or out-of-scope for memory, then test whether only authorized information affects later behavior.	Claims that memory follows user consent, project boundaries, or session boundaries.
Auditability	Report where the information is stored or expose a behavioral audit that identifies which support item drove the response.	Claims that parametric memory is inspectable, controllable, or safer than explicit memory.