DeltaWrite: Selective Persistent Inference-Time Adaptation for Frozen Language Models

Author Name

April 17, 2026

Abstract

We present DeltaWrite, a system for selective, persistent, and reversible inference-time adaptation of frozen large language models. DeltaWrite targets a practical regime that is underserved by existing approaches: inserting a bounded piece of knowledge or behavior into a deployed model without running a standard fine-tuning job and without paying prompt-budget costs on every query. The system combines a compact runtime modification with a content-addressable routing mechanism that determines whether the modification should activate for a particular query. Across internal evaluations, the approach demonstrates strong paraphrase-robust recall on stock transformer hosts, practical scaling to large registries of inserted items under learned dispatch, and clean negative results showing that routing is necessary for composition. We also report a contrasting negative result on a purpose-built weight-resident memory architecture, which helps clarify where the current method is mature and where it remains an open research problem. Because patent activity is still ongoing, this manuscript intentionally omits the minimal implementation details required for direct reproduction of the core write mechanism. However, it does provide a precise problem formulation, system decomposition, evaluation protocol, empirical results, limitations, and falsifiable claims suitable for technical review.

1 Introduction

Large language models are increasingly deployed in settings where model behavior must evolve after pretraining. Operators may need to add a new fact, encode a customer-specific policy, override a default answer, or supply a narrow domain behavior without retraining an entire model family. Existing solutions each leave important gaps. Prompting is flexible but ephemeral and costly in tokens. Retrieval-augmented generation improves factual grounding but depends on an external memory system and repeated query-time evidence injection. Fine-tuning and adapter-based methods can be effective, but they introduce optimization infrastructure, training data requirements, and deployment friction that are poorly matched to small, fast updates.

This paper studies a different operating point: can a frozen transformer acquire selective persistent behavior through an inference-time mechanism that does not require conventional gradient-based adaptation during deployment? DeltaWrite is our answer to that question. It constructs a compact runtime intervention from a forward interaction with the host model, persists that intervention across prompts or sessions, and uses a routing mechanism to determine when the intervention should activate. The goal is not broad distributional change. The goal is a bounded write that is persistent, reversible, and mostly invisible to unrelated queries.

The central challenge is composition. A single inserted item is interesting, but not sufficient. A useful system must host many inserted items without letting them interfere destructively. Our experiments indicate that selective routing is therefore not a peripheral engineering detail; it is a core scientific component. With accurate routing, large collections of inserted items can coexist with strong recall and minimal leakage. Without routing, naive superposition fails even at small scales.

The contribution of this paper is threefold. First, we define a practical system architecture for selective persistent inference-time adaptation of frozen transformers. Second, we provide an empirical evaluation that separates write quality, routing quality, capacity, leakage, and architectural dependence. Third, we document both positive and negative results, including a contrast between a production-leading overlay architecture and a less mature weight- resident memory architecture. The paper is intentionally written to preserve IP position on the enabling construction while still exposing enough of the method and evidence for serious technical assessment.

2 Contributions and Disclosure Boundary

This manuscript makes the following claims.

1. A frozen transformer can support targeted persistent behavioral overlays created at inference time rather than through a standard training loop.
2. Selective routing enables practical multi-item composition, whereas naive additive superposition does not.
3. The resulting system can achieve strong paraphrase-robust recall and capacity scaling on stock open-weight model families.
4. A purpose-built weight-resident memory architecture remains an open problem despite its conceptual appeal.

At the same time, this manuscript does *not* disclose the exact minimal recipe required to reproduce the core write mechanism. Specifically, we omit the most enabling implementation choices around the write construction, detailed internal targeting recipe, and certain calibration details that are central to current patent activity. We instead provide a reproducibility boundary: enough for a reviewer or collaborator to understand the problem, architecture, evaluation logic, and strength of evidence, but not enough to trivially clone what appears to be the inventive core.

3 Related Work

DeltaWrite sits at the intersection of several literatures.

Prompting and in-context control. Prompt-based methods can induce new behavior without changing weights, but the effect is transient and consumes context budget. DeltaWrite instead aims for persistence across prompts and reuse without repeatedly reinserting the same evidence.

Retrieval-augmented generation. Retrieval systems improve access to mutable knowledge by injecting retrieved context at query time. DeltaWrite does not replace retrieval for large external corpora, but it occupies a distinct point in the design space: the inserted item behaves more like a persistent internal overlay than an external document snippet.

Fine-tuning and parameter-efficient adaptation. Adapter methods and related parameter-efficient tuning strategies can produce targeted behavior with small parameter updates, but they typically rely on offline optimization. DeltaWrite targets deployment-time adaptation without a standard training loop.

Model editing. Existing model-editing approaches demonstrate that local parameter changes can alter factual recall or behavior. DeltaWrite is closest in spirit to this family, but differs in operational emphasis: it is explicitly framed around inference-time construction, runtime persistence, and large-scale composition via routing.

Approach	**Persistence**	**Prompt budget**	**Deployment-time optimization**	**Best use case**
Prompting / in-context control	Per prompt	Yes	No	Fast temporary steering and task setup
Retrieval-augmented generation	External memory only	Yes	No	Large mutable corpora and citation-grounded answers
Fine-tuning / PEFT adapters	Persistent model or adapter state	No	Yes	Broad distributional adaptation and domain specialization
Classic model editing	Persistent parameter edit	No	Usually yes at edit time	Local factual or behavioral correction
DeltaWrite	Persistent runtime overlay or weight-resident module	No	No standard training loop at deployment	Selective bounded updates with reversible multi-item serving

Table 1: High-level comparison of DeltaWrite with adjacent adaptation paradigms.

Table 1 clarifies the main design-space distinction. The most important difference is not simply whether parameters are changed, but when adaptation cost is paid and how the resulting change is served. DeltaWrite aims for a regime in which registration is lightweight, persistence is available without repeated token injection, and multiple bounded updates can coexist under explicit selection logic.

The precise novelty claim of this paper is therefore not that models can ever be edited, but that a useful system for *selective persistent inference-time adaptation* can be built around a compact write mechanism plus a content-aware router, and that this combined system exhibits a particular empirical profile under scaling and control tests.

4 Problem Formulation

We study the following task. Let a frozen base model serve interactive queries. Given a target item of knowledge or behavior, the system should register an update that satisfies five properties: (i) it alters the model only in a narrow intended way, (ii) it persists across prompts or sessions, (iii) it can be removed cleanly, (iv) it remains largely dormant for unrelated inputs, and (v) it composes with many other registered items.

This differs from broad alignment, domain adaptation, or long-horizon continual learning. The target object is not a new global capability but a bounded overlay such as a factual insertion, policy override, terminology correction, or narrow behavioral mapping. The evaluation problem is therefore also specific. A system must be judged not only on whether the inserted item can be recalled, but also on whether it triggers under paraphrase, avoids false activation, and survives competition with many neighboring items.

5 Method Overview

DeltaWrite has two separable components: a *write path* and a *dispatch path*. The write path constructs a compact runtime intervention from a forward interaction with the frozen model and a desired target behavior. The dispatch path determines whether a new query should activate that intervention.

5.1 Write Path

At a high level, the write path observes internal structure associated with a desired completion and produces a compact directional modification in model computation. The resulting intervention is small relative to the host model and is designed to be causally meaningful only when the runtime activation resembles the intended trigger context. This distinguishes the method from unconditional parameter editing. The intervention is not meant to globally rewrite the model; it is meant to be locally potent under the correct conditions.

Several aspects matter for scientific interpretation even without disclosing the full recipe. First, the write path is closed-form at deployment time: it does not run a conventional optimizer, loss-minimization loop, or backpropagation procedure while registering the item. Second, the written object is compact and algebraically structured, which supports clean removal and persistence. Third, the write is cheap enough to support interactive registration workflows rather than only offline training pipelines.

5.2 Dispatch Path

The dispatch path converts a collection of inserted items into a practical memory system. Given an incoming query, the router chooses one target intervention, a small subset, or abstains entirely. This can be implemented through learned classifiers or semantic-routing mechanisms. The key scientific point is that the router is part of the method rather than a thin wrapper around it. The empirical

results show that accurate selection is what turns compact writes into a scalable system.

5.3 Persistence Modes

The broader program includes two persistence models. The first is an overlay model in which registered interventions persist as runtime objects that can be saved and reloaded across sessions while leaving the base checkpoint unchanged. The second is a weight-resident model in which inserted modules are accumulated into dedicated architectural components. The overlay model is currently the more mature path; the weight-resident model remains exploratory.

6 Experimental Setup

The evaluation program is designed to isolate four questions: does a registered item work at all, does it generalize beyond the exact registration prompt, does it remain selective among many neighbors, and how sensitive is it to host architecture?

Hosts. The main positive results come from stock transformer hosts, including a 7B instruction-tuned model and a 1.5B coder-family model. A separate research track evaluates a purpose-built approximately 215M-parameter memory- centric architecture with dedicated bank layers.

Task families. Experiments cover curated factual question-answer pairs, paraphrased query sets, capacity stress tests with large registries of inserted items, and architectural diagnostics. Internal knowledge-base evaluations use a seed benchmark with 60 items and an expanded benchmark with 560 items.

Metrics. We report recall under literal and paraphrased queries, routing accuracy, control leakage, and per-query latency. We also compare oracle routing with learned dispatch to separate limitations of the write path from limitations of the serving logic.

Controls and ablations. The study includes at least three important controls: oracle-vs-learned routing, raw superposition without dispatch, and a zeroed-layer diagnostic that tests whether the inserted overlay remains causally effective when the host layer is heavily ablated.

7 Results

7.1 Main Quantitative Results

Table 2 summarizes the core findings reported in the internal evaluation suite.

7.2 What the Numbers Show

The most important takeaway from the 7B experiment is the small gap between oracle and learned routing. Perfect oracle-routed paraphrase recall indicates that the inserted item can survive paraphrase and activate reliably when the system knows which item to select. The drop to 0.94

Evaluation	Value	Interpretation
Oracle-routed paraphrase recall on a 7B host with 50 inserted items	1.000	The write path succeeds when selection is ideal.
Learned-dispatch paraphrase recall on the same setup	0.940	Residual error is largely attributable to routing.
Capacity stress on a 1.5B host with 500 inserted items	499/500	Large registries are feasible under dispatch.
Control leakage in the 500-item stress test	0	Off-target activation is low in the tested regime.
Per-query latency at 500 registered items	~250 ms	Dispatch overhead remains operationally plausible.
Zeroed-layer diagnostic with oracle routing, 10 items	100% recall	The inserted overlay is causally active.
Raw superposition without dispatcher, 10 items	0/10	Routing is necessary for composition.
Purpose-built memory architecture on a 60-item benchmark	0/60	Weight-resident memory remains unresolved.

Table 2: Patent-safe summary of core internal results.

under learned dispatch is still strong, but it shifts the optimization target: the main bottleneck in that regime is classification quality in the router rather than failure of the write itself.

The capacity-stress result is important for product relevance. Achieving 499/500 correct routes with no reported control leaks at a registry size of 500 suggests that the system can operate as more than a toy memory demo. It can support a substantial catalog of inserted items while maintaining query selectivity. Latency around a quarter second per query at this scale implies a real but manageable serving cost.

The zeroed-layer diagnostic strengthens the mechanistic interpretation. If recall remains perfect when the relevant layer is zeroed and oracle routing is used, then the inserted overlay is not merely biasing an already-correct model toward a nearby answer. It is carrying causal signal that can substitute for heavily ablated host computation in the tested setting.

8 Ablations and Negative Results

Negative results are crucial here because they constrain overclaiming.

No-dispatch superposition fails. When multiple inserted items are combined without a router, the reported result is 0 successful recalls out of 10. This sharply limits any interpretation in which the system is simply adding more memory capacity to a model in a generic way. Instead, the data support a more specific thesis: compact writes are useful when paired with structured selection.

Weight-resident memory remains immature. A separate research line built an architecture with dedicated knowledge-bank layers intended to host persistent inserted modules directly in model parameters. Despite the conceptual appeal, reported Phase 1E results show 0 successful recalls out of 60 benchmark items. This negative finding is scientifically valuable. It indicates that making a model architecturally receptive to persistent memory is not straightforward, and that acceptance,

normalization, or downstream integration may be major barriers.

Architecture sensitivity is real. The briefing indicates that host families require materially different calibration and characterization. This means the approach appears portable across model lines, but not in a naive plug-and-play sense. A serious deployment would require per-family validation.

9 Discussion

The empirical picture suggests that DeltaWrite is best understood not as a replacement for all model adaptation methods, but as a new operating point between prompting and fine-tuning. It is stronger than prompting on persistence, stronger than retrieval on internalization of the inserted item, and lighter than standard adapter training in deployment complexity. At the same time, it is narrower than fine-tuning in scope and less appropriate than retrieval for large mutable corpora.

Scientifically, the most interesting claim is the division of labor between writing and routing. The experiments suggest that the system's ability to encode an item is already strong on stock hosts, whereas system-scale performance is mostly governed by how well the serving layer recognizes when to activate that item. This reframes the memory problem: scalable inference-time adaptation may be less about storing many items than about selecting them with high precision.

The contrast between the overlay system and the purpose-built memory model is also instructive. The more theoretically elegant architecture currently performs worse than the pragmatic overlay system. This is a valuable lesson for future research. It suggests that architectural accommodation for persistent memory is not sufficient on its own; the model must also accept, preserve, and use the inserted signal in a stable way.

10 Limitations

This paper has several limitations.

First, the disclosure boundary means the method section is less explicit than a fully open academic paper would be. We believe the remaining detail is adequate for technical assessment of the claims, but not for direct reproduction. Second, the evaluation suite is internally curated rather than a single canonical public benchmark. Third, the current evidence is strongest for targeted factual or behavioral overlays, not for broad domain adaptation. Fourth, portability across host families is promising but not yet reduced to a general theory. Finally, the weight-resident memory line remains negative, which means the paper supports a strong overlay-based claim but only a tentative architectural one.

11 Operational and Safety Considerations

A system that can write persistent behavioral overlays at inference time should ship with governance mechanisms. Registered items should be attributable, versioned, auditable, and removable. Routers

should be tested not only for accuracy but also for abstention behavior and off-target activation. Because the mechanism operates below the prompt surface, inspection and rollback are not optional product features; they are part of the safety case.

12 Conclusion

DeltaWrite provides evidence that frozen language models can support selective, persistent, and reversible inference-time adaptation through the combination of compact writes and content-aware routing. On stock transformer hosts, the method shows strong paraphrase-robust recall, practical scaling to hundreds of registered items, low observed leakage in the tested regime, and a clear separation between the success of the write path and the residual limitations of the router. Equally important, the negative results clarify the boundaries of the claim: routing is necessary for composition, and weight-resident memory remains an open research problem.

Taken together, these results are strong enough to motivate serious follow-up. The paper does not claim a universal substitute for retrieval or fine-tuning. Instead, it defines a narrower and potentially important category: selective persistent inference-time adaptation for frozen models. If subsequent work strengthens the theory, broadens the benchmarks, and continues to validate the results under guarded disclosure, this line of work may deserve a stable place alongside prompting, retrieval, and parameter-efficient adaptation in the modern LLM systems toolkit.