TEST-TIME TRAINING FOR PRETRAINED LLMS: AN EMPIRICAL STUDY OF LIMITED FAST-WEIGHT UTILIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time training (TTT) enables a model to learn from its input context during inference by updating a subset of its parameters, effectively turning them into a form of dynamic memory. In-Place TTT extends this approach to pretrained LLMs by repurposing existing MLP parameters as fast weights through continued pretraining. Unlike dedicated memory trained from scratch, these parameters already encode useful computation, creating a tension between contextual adaptation and pretrained capabilities. We study this tension through a detailed analysis across multiple fast-weight learning rules, varying the update objective as well as write and forget gating mechanisms. Our central finding is that performance gains can arise largely from changes learned by the underlying model during TTT-aware training, while the direct contribution of inference-time fast-weight updates is often surprisingly small. To understand how this behavior emerges, we examine training dynamics and find that TTT-aware training progressively limits how strongly fast-weight updates alter the model's computation, through mechanisms that depend on the learning rule. Amplifying updates at inference time can leave long-context retrieval intact while degrading pretrained capabilities, suggesting that capability preservation may partly explain this suppression. We further analyze how TTT-aware training shapes context utilization: the underlying model learns to attend more strongly to relevant evidence even without fast-weight updates, whereas the write gates governing those updates become more selective and focused on query-relevant tokens despite their limited direct contribution. Together, our results highlight the challenges of turning pretrained parameters into dynamic memory without disrupting the computation they already encode.

1 INTRODUCTION

Test-Time Training (TTT) (Sun et al., 2025; Zhang et al., 2025; Han et al., 2025; Behrouz et al., 2026; Tandon et al., 2025; Behrouz et al., 2025b) has recently gained increasing attention as a mechanism for models to continue learning from inputs encountered at inference time. In TTT, a subset of model parameters, often referred to as fast weights (Ba et al., 2016b), is updated as the context is processed, allowing information from the input to be internalized in the model parameters. Originally introduced for adapting predictive models to distribution shifts (Sun et al., 2020), recent work has repurposed TTT for long-context modeling, viewing online parameter updates as a form of memory or meta-learning (Liu et al., 2026; Zhang et al., 2025).

While many recent TTT approaches jointly train the model and its fast-weight update mechanism from scratch, applying TTT to an already pretrained LLM is an important practical setting. Recent work on hybrid State-Space Models has similarly reused pretrained Transformers, to preserve their capabilities and compare architectural variants without training each from scratch (Chattopadhyay et al., 2026; Bick et al., 2024; 2026). In-Place TTT (Feng et al., 2026) and its subsequent work (Ouyang et al., 2026; Wang et al., 2026) follows this direction by repurposing existing MLP down-projection parameters as fast weights without adding or replacing architectural components. Further, its chunk-wise updates enable efficient parallel computation. These properties make In-Place TTT a practical framework for studying fast-weight adaptation in pretrained LLMs.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

TTT formulates learning as a bi-level optimization problem. The *inner-loop* updates fast-weights to encode context-specific information using an inexpensive auxiliary objective, while the *outer-loop* optimizes the model end-to-end for next-token prediction, thereby shaping both the fast-weight update mechanism and how the model responds to the updated weights. However, when pretrained parameters themselves are repurposed as fast weights, they are not an empty memory state: they already implement useful computation. Updating fast-weights therefore creates a **tension between incorporating context-specific information and preserving pretrained capabilities.**

As the outer-loop is optimized in the presence of these inner-loop updates, it can respond to this tension by shaping both the magnitude of the updates and the model's reliance on them. The resulting performance gains may therefore reflect both the use of fast-weight updates at inference time and changes learned by the underlying model during training. This raises our central question: when TTT-aware training improves a pretrained language model, how much of the gain comes from each source? We disentangle these two contributions using the In-Place TTT framework. To better understand the mechanisms underlying these contributions, we analyze multiple inner-loop learning rules that vary in their update objective, token-wise gating, and decay mechanism—design choices introduced in recurrent models (Schlag et al., 2021; Yang et al., 2024; 2025b) to increase the expressivity of online state updates.

Our central finding is that the performance changes induced by TTT-aware training can arise largely from changes learned by the underlying model, while the direct contribution of inference-time fast-weight updates remains surprisingly small. Across learning rules, TTT-trained models often perform similarly on long-context benchmarks with the fast-weight updates enabled or disabled, despite being explicitly trained with these updates. We then examine how this behavior emerges during training. Outer-loop optimization progressively suppresses the functional magnitude of fast-weight updates, using different mechanisms depending on the form of the inner-loop learning rule. A controlled analysis of update magnitude suggests why this attenuation emerges: larger fast-weight updates can better incorporate information from the current context, but increasingly interfere with capabilities already encoded in the pretrained model. This trade-off provides a plausible explanation for why training drives the updates toward a small magnitude.

We further examine how TTT-aware training shapes context utilization, both in the underlying model and through the inner-loop updates. Under query-aware retrieval, TTT-trained models retain most of their performance improvement over standard continued pretraining even with fast-weight updates disabled, while showing stronger attention to query-relevant evidence. Meanwhile, token-wise gating also becomes increasingly selective throughout TTT-aware training and modestly more focused on relevant context, suggesting that inner-loop updates remain sensitive to context relevance despite their small direct contribution to performance.

Our main contributions include:

- We disentangle the contributions of outer-loop training and inference-time fast-weight updates, showing that much of the improvement persists even when the updates are disabled.
- We identify the need to preserve pretrained capabilities as a potential reason why outer-loop training suppresses fast-weight update magnitude, highlighting the tension between contextual adaptation and capability preservation.
- We systematically study these phenomena across multiple inner-loop learning rules and long-context retrieval, QA, and in-context learning tasks across two pretrained language models.

2 Preliminaries and Methods

This section presents In-Place TTT framework and the inner-loop learning rules evaluated in our study. We defer a broader discussion of related work to Appendix A

2.1 In-Place Test-time Training

In-Place TTT (Feng et al., 2026) repurposes the MLP down-projection matrix W_{down} as fast weights that evolve sequentially across input chunks, leveraging the MLP's existing role as a key–value memory (Geva et al., 2021; Dai et al., 2022).

In-Place TTT sequentially processes input chunks while updating the fast weights between chunks. The inner loop learning rule is formulated as associative key–value (KV) binding: the fast weights are updated to map each key to its associated value. Let $W_t \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ denote the fast weights before chunk t, and $X_t \in \mathbb{R}^{c \times d_{\text{model}}}$ its MLP input. The intermediate activations of the gated MLP (Shazeer, 2020) serve as keys: $K_t \equiv Z_t = \phi(X_t W_{\text{gate}}^\top) \odot (X_t W_{\text{up}}^\top) \in \mathbb{R}^{c \times d_{\text{ff}}}$, where ϕ is SiLU and $\odot$ denotes element-wise multiplication. The target values are constructed as $V_t = \text{Conv1D}(X_t) W_{\text{target}}^\top \in \mathbb{R}^{c \times d_{\text{model}}}$, allowing each target to incorporate neighboring-token information.

For each chunk t, the current fast weights W_t produce the MLP output $O_t = K_t W_t^\top$, and the chunk's key–value pairs are then used to compute a fast-weight update Δ_t for subsequent chunks. The inner-loop objective uses the negative Frobenius inner product:

$$\mathcal{L}_{\text{Frobenius}} = -\langle K_t W_t^\top, V_t \rangle_F, \qquad W_{t+1} = W_t + \underbrace{\eta V_t^\top K_t}_{\Delta_t}, \tag{1}$$

where η denotes the inner-loop learning rate. This yields a closed-form **additive** outer-product update.

2.2 Fast-Weight Learning Rule Variants

In-Place TTT uses an additive learning rule that accumulates key–value associations in the fast weights. Work on recurrent models has developed more expressive update rules through prediction-error corrections, gating, and decay (Schlag et al., 2021; Yang et al., 2024; 2025b), reporting a positive relationship between update-rule expressivity and downstream performance (Chattopadhyay et al., 2026). We refer readers to Appendix A.2 for further discussion. We ask whether this relationship also holds when the updated state consists of pretrained MLP parameters.

Prediction-error Learning Objective We consider a squared error objective:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2} \| K_t W_t^\top - V_t \|_F^2, \qquad W_{t+1} = W_t + \eta (V_t - K_t W_t^\top)^\top K_t. \tag{2}$$

Unlike the additive rule, this update writes the residuals between the value predicted by the current fast weights and the target value.

Write Gating To control which tokens contribute to the fast-weight update, we introduce token-wise write gates, $\boldsymbol{\beta}_t = (\beta_{t,1}, \ldots, \beta_{t,c}) \in [0,1]^c$ for chunk t:

$$\Delta_t = \eta \sum_{i=1}^{c} \beta_{t,i} \delta_{t,i} \tag{3}$$

where i indexes token positions within the chunk and $\delta_{t,i}$ is the corresponding token-level update.

Decay Gating Following Gated DeltaNet (GDN) (Yang et al., 2025b), we further introduce a token-wise decay gate (Gers et al., 1999) $\alpha_{i,t} \in [0,1]$, which is aggregated within each chunk($\alpha_t = \frac{1}{c} \sum_{i=1}^{c} \alpha_{i,t}$), and decays the existing fast-weight state:

$$W_{t+1} = \alpha_t W_t + \Delta_t. \tag{4}$$

Because W_0 contains pretrained parameters rather than an empty memory state, we additionally consider decaying only the accumulated residual:

$$H_{t+1} = \alpha_t H_t + \Delta_t, \qquad W_{t+1} = W_0 + H_{t+1}. \tag{5}$$

3 Experiment Setups

3.1 Model and Architectures

We use Qwen3-4B-Base (Yang et al., 2025a) and LLaMA-3.1-8B (Grattafiori et al., 2024) as the backbone model and perform continued pre-training on a long-context corpus. To closely match the training setup of In-Place TTT (Feng et al., 2026), we follow its continued pre-training configuration unless otherwise specified. Details are provided in Appendix B

Table 1: **Training and inner-loop configurations of the evaluated variants.** All trained variants use next-token prediction (NTP) as the outer-loop objective. Let $E_t = V_t - K_t W_t^\top$ denote the prediction error for chunk t; $\boldsymbol{\beta}_t$ is broadcast across the feature dimension.

Variant	Objective	Write	Decay	Fast-Weight Update
`standard-CPT`	–	–	–	–
`Frobenius`	Frobenius	–	–	$W_{t+1} = W_t + \eta V_t^\top K_t$
`Frobenius+`β	Frobenius	$\boldsymbol{\beta}$	–	$W_{t+1} = W_t + \eta(\boldsymbol{\beta}_t \odot V_t)^\top K_t$
`Delta`	MSE	–	–	$W_{t+1} = W_t + \eta E_t^\top K_t$
`Delta+`β	MSE	$\boldsymbol{\beta}$	–	$W_{t+1} = W_t + \eta(\boldsymbol{\beta}_t \odot E_t)^\top K_t$
`GDN`	MSE	$\boldsymbol{\beta}$	Full State	$W_{t+1} = \alpha_t W_t + \eta(\boldsymbol{\beta}_t \odot E_t)^\top K_t$
`GDN-Residual`	MSE	$\boldsymbol{\beta}$	Residual	$W_{t+1} = W_0 + \alpha_t(W_t - W_0) + \eta(\boldsymbol{\beta}_t \odot E_t)^\top K_t$

Baseline We first establish a standard continued-pretraining baseline, denoted `standard-CPT`, trained on the same corpus with the next-token prediction (NTP) objective. All TTT variants retain this NTP objective for outer-loop training, while introducing an additional inner-loop fast-weight update during sequence processing.

Inner-Loop Variants We consider two families of inner-loop updates, summarized in Table 1. The first uses In-Place TTT's Frobenius objective to produce additive updates (`Frobenius`) (Feng et al., 2026); the second uses an MSE objective to produce prediction-error updates (`Delta`) (Schlag et al., 2021; Yang et al., 2024; 2025b). `Frobenius+`β and `Delta+`β add a learned token-wise write gate $\beta_{t,i} \in [0, 1]$ to `Frobenius` and `Delta`, respectively. `GDN` further introduces a learned decay gate $\alpha_t \in [0, 1]$, while `GDN-Residual` applies decay only to the residual fast-weight state relative to the pretrained initialization.

3.2 Training

Following In-Place TTT (Feng et al., 2026), we use a two-stage continued pretraining schedule. We first train with a context length of 32K for 23B tokens, followed by a second stage with 128K-length for up to 10B additional tokens. Unless otherwise specified, we follow the remaining training hyperparameters of In-Place TTT. We use a long-context continued pre-training corpus, packing multiple samples into fixed-length sequences. To align sample boundaries with the 4,096-token TTT chunks, each sample is padded to the next chunk boundary before packing, and the fast-weight state is reset at every new sample, preventing updates from propagating across independent examples. Additional details on training configuration and data are provided in Appendix B.

3.3 Evaluation

Benchmark We evaluate TTT along three complementary dimensions. First, for controlled long-context retrieval, we use RULER (Hsieh et al., 2024), probing information retention and retrieval over long context. Second, for broader downstream evaluation, we use the question answering (QA) and in-context learning (ICL) subsets of HELMET (Yen et al., 2025). Finally, we evaluate MMLU (Hendrycks et al., 2021), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), and ARC-Challenge (Clark et al., 2018) to assess interference with pretrained capabilities. Details are in Appendix B.4.

Metric We report two complementary quantities throughout our evaluation. **TTT-Benefit** measures the improvement of a TTT-trained model over a separately trained `standard-CPT` baseline, whereas **TTT-Effect** isolates the contribution of the test-time fast-weight update by comparing the same model with the update enabled(TTT-On) and disabled(TTT-Off).

Table 2: **TTT-aware training yields inconsistent gains, while inference-time fast-weight updates have limited direct effects.** Long-context performance on RULER and the QA and ICL subsets of HELMET. **Perf.** (Performance) denotes accuracy; for TTT-trained models, it is measured with inference-time fast-weight updates enabled. **Ben.** (TTT-Benefit) is improvement over the separately trained `standard-CPT` baseline, while **Eff.** (TTT-Effect) measures the performance difference between enabling (TTT-ON) and disabling (TTT-OFF) the inference-time update within the same TTT-trained model. Average rows report the mean across the six TTT-trained variants.

Variant	16K			32K			64K			128K		
	Perf.	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**
RULER												
Qwen3-4B-Base	88.5	–	–	82.2	–	–	64.8	–	–	51.2	–	–
`standard-CPT`	89.4	–	–	84.3	–	–	78.5	–	–	69.3	–	–
`Frobenius`	89.6	0.2	0.1	83.5	-0.8	-0.3	79.1	0.6	-0.2	68.9	-0.4	-1.2
`Frobenius+`β	89.7	0.3	0.1	84.9	0.6	0.3	79.2	0.7	-0.4	66.9	-2.4	-3.2
`Delta`	89.1	-0.3	-0.1	84.1	-0.2	-0.4	78.1	-0.4	-0.6	67.4	-1.9	-1.2
`Delta+`β	89.6	0.1	-0.1	83.7	-0.6	0.2	78.8	0.3	0.6	68.3	-1.0	-0.5
`GDN`	89.8	0.4	0.1	84.6	0.3	0.2	78.6	0.1	0.5	68.3	-1.0	-0.4
`GDN-Residual`	89.9	0.5	0.0	84.1	-0.2	0.1	78.8	0.3	0.7	68.7	-0.6	-0.1
Average	89.6	0.2	0.0	84.2	-0.2	0.0	78.8	0.3	0.1	68.1	-1.2	-1.1
HELMET (QA + ICL)												
Qwen3-4B-Base	57.1	–	–	60.2	–	–	61.2	–	–	44.9	–	–
`standard-CPT`	56.3	–	–	60.9	–	–	61.6	–	–	55.3	–	–
`Frobenius`	57.2	0.9	0.4	60.9	0.1	0.7	63.9	2.4	0.8	55.5	0.3	-0.2
`Frobenius+`β	56.5	0.2	0.2	60.8	0.0	0.4	64.4	2.8	1.3	55.2	0.0	-0.5
`Delta`	56.4	0.0	0.1	60.2	-0.6	-2.1	62.8	1.3	-0.2	56.1	0.8	-0.6
`Delta+`β	55.1	-1.2	-0.2	60.3	-0.6	0.5	61.6	0.0	-0.3	54.9	-0.4	0.6
`GDN`	54.8	-1.5	0.2	60.3	-0.6	0.3	60.5	-1.1	-0.9	55.8	0.5	0.6
`GDN-Residual`	55.4	-0.9	-0.4	59.7	-1.1	-0.8	61.9	0.3	-0.1	55.2	0.0	-1.3
Average	55.9	-0.4	0.1	60.4	-0.5	-0.2	62.5	0.9	0.1	55.4	0.2	-0.2

4 UNDERSTANDING TEST-TIME TRAINING IN CONTINUED PRETRAINING

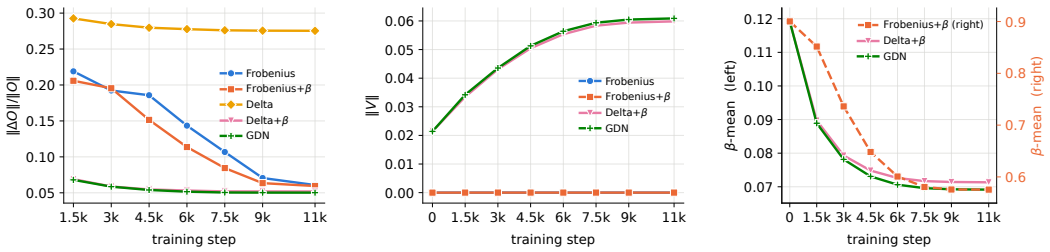
In this section, we study how pretrained LLMs use and regulate fast-weight updates. We first test whether TTT-aware training induces reliance on inference-time updates, then examine how outer-loop training shapes their writing behavior and what constrains their magnitude.

4.1 TTT-AWARE TRAINING DOES NOT GUARANTEE RELIANCE ON FAST-WEIGHT UPDATES

We first ask whether TTT-trained models actually rely on fast-weight updates at inference time. We distinguish **TTT-Benefit** (TTT-ON $-$ `standard-CPT`) from **TTT-Effect** (TTT-ON $-$ TTT-OFF), separating gains from TTT-aware training from the direct contribution of the inference-time update.

Long-context CPT strengthens the backbone. The results on RULER in Table 2 show that continued pre-training on our long-context corpus substantially improves the long-context capability of the backbone model, with `standard-CPT` improving over Qwen3-4B-Base($64.8 \rightarrow 78.5$ at 64K and $51.2 \rightarrow 69.3$ at 128K.) A similar trend appears on QA and ICL subsets of HELMET, where the largest improvement is observed at 128K (44.9→55.3).

TTT-aware training yields inconsistent additional gains. On RULER, TTT-Benefit varies across learning rules and context lengths: for example, `Frobenius+`β improves over `standard-CPT` by $+0.6$ and $+0.7$ points at 32K and 64K, but falls below it by -2.4 points at 128K. HELMET shows a similarly mixed pattern, with gains up to $+2.8$ points at 64K but no consistent improvement across variants.



(a) Relative MLP output change. (b) Value norm $\|V\|_2$. (c) Mean token-wise write gate β.

Figure 1: **Outer-loop training progressively reduces fast-weight updates.** (a) Relative MLP output change decreases across all evaluated learning rules. (b) Value norms exhibit distinct dynamics across learning rules. (c) Mean token-wise write gates decrease. Frobenius+β uses the right axis.

TTT-aware training does not imply reliance on fast-weight updates. On RULER, TTT-Effect is generally small: most values remain within one point across learning rules and context lengths, despite the models being explicitly trained with active fast-weight updates. More strikingly, TTT-aware training can improve performance even when enabling the inference-time update has little or negative effect: at 64K, for example, TTT-Benefits of `Frobenius` and `Frobenius+`β are $+0.6$ and $+0.7$ points, but their TTT-Effects are -0.2 and -0.4, respectively. The same behavior appears on HELMET: for `Delta`, TTT-Benefit is $+1.3$ at 64K and $+0.8$ at 128K, while TTT-Effect is -0.2 and -0.6. We observe the same qualitative behavior with LLaMA-3.1-8B in Table 4 in Appendix C.1, suggesting that this limited utilization on inference-time fast weight updates is not specific to a single backbone. **Thus, improvements from TTT-aware training should not be attributed directly to inference-time fast-weight updates;** we later show that much of its gain can instead arise from changes to the underlying model by outer-loop optimization(Section 5).

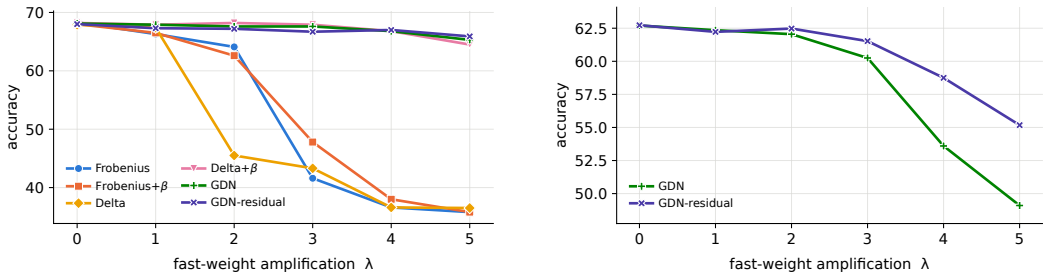
4.2 Outer-Loop Training Regulates Update Magnitude

The small TTT-Effect in Section 4.1 suggests that outer-loop training may limit the functional influence of fast-weight updates. We examine this by tracking the relative change in the output of the TTT-adapted MLP-down projection induced by the fast-weight update, [1] , together with $\|V\|$ and the learned β and α gates over training.[2] Details for the experiment are provided in Appendix B.6.

Different learning rules attenuate updates through distinct mechanisms. As shown in Figure 1a, the relative change in the TTT-adapted MLP output decreases over training, suggesting that outer-loop optimization progressively limits the influence of fast-weight updates. This behavior is consistent with the tension introduced earlier: because the fast weights are pretrained parameters rather than an empty memory state, large writes can interfere with their existing function. The mechanism differs across learning rules. For the Frobenius rule, $\Delta_t \propto V_t^\top K_t$, so keeping the learned target representation V_t small directly suppresses the write. In Figure 1b we observe that $\|V_t\|$ remains close to zero throughout training. For prediction-error rules, $\Delta_t \propto (V_t - O_t)^\top K_t$, where $O_t = K_t W_t^\top$. Because V_t is initialized near zero while the pretrained output O_t is nonzero, a small V_t does not imply a small update; initially, the residual is approximately $-O_t$. The observed growth of $\|V_t\|$ during training is therefore consistent with V_t moving toward the scale of O_t, which would reduce the prediction residual. When a write gate is available, the model additionally decreases β_t, providing a direct multiplicative mechanism for attenuating the remaining residual update. The ungated `Delta` variant lacks this additional control and retains a larger functional perturbation throughout training. Consistent with this, it exhibits negative TTT-Effect in 7 of 8 benchmark–length settings in Table 2. Overall, these dynamics reflect the tension introduced earlier: **when pretrained parameters serve as fast weights, outer-loop training must balance writing context-specific information against preserving their existing function**, leading to increasingly attenuated updates through rule-dependent mechanisms.

[1] $\|O_t - O_t^0\|/\|O_t^0\|$, where $O_t = K_t W_t^\top$ and $O_t^0 = K_t W_0^\top$. We average across TTT layers and chunks

[2] To examine how fast-weight behavior evolves from the pretrained initialization during training, we analyze the Stage-1 training trajectory. The same qualitative pattern of limited TTT-Effect is already present after Stage-1 (Appendix C.3).



(a) Accuracy on RULER at 64K

(b) Accuracy on MMLU, ARC-challenge, HellaSwag, and PIQA

Figure 2: **Retrieval robustness to stronger fast-weight updates in prediction-error-based rules does not ensure preservation of pretrained capabilities.** We amplify inference-time updates by a factor of λ. (a) `GDN` and `GDN-Residual` maintain retrieval performance well beyond the learned update scale. (b) The same amplification degrades pretrained capabilities after updates accumulate over an unrelated 128K-token context, with `GDN-Residual` degrading less severely than `GDN`.

4.3 Stronger Fast-Weight Updates Expose a Trade-off Between Adaptation and Preservation

The previous section shows that outer-loop training progressively reduces the magnitude of fast-weight updates. We next ask why such small update magnitudes are learned: do stronger writes interfere with writing context-specific information or with the pretrained capabilities of the model? For each chunk t, we rescale the update as $\Delta_t^{(\lambda)} = \lambda \Delta_t$, where $\lambda = 1$ corresponds to the originally learned update and $\lambda = 0$ disables test-time adaptation.

Prediction-error updates remain robust to amplification. Figure 2a shows how RULER performance changes when the learned fast-weight update is amplified at inference time. At 64K, `Frobenius` degrades rapidly as λ increases, while adding a token-wise write gate (`Frobenius+`β) delays this degradation. Because β re-weights token-level contributions before aggregated into the update, it can be interpreted that write gate is learning a more selective update direction that is less sensitive to global amplification, which is later discussed in Section 5.2.

In contrast, prediction-error-based variants (`Delta+`β, `GDN`, and `GDN-Residual`) remain stable over a substantially wider range of λ. This difference is consistent with their corrective form: the update depends on the current prediction residual $V_t - O_t$, so the correction changes as the fast weights approach the target, whereas the Frobenius rule continues to add $V_t^\top K_t$ independently of the current prediction. Importantly, λ rescales each rule relative to its own learned update, so these curves do not compare absolute update magnitudes across rules. Nor does greater robustness to amplification imply better performance at $\lambda = 1$. Instead, the result shows that, for prediction-error rules, the update magnitude learned during outer-loop training lies well below the point at which long-context retrieval itself becomes unstable. This motivates the next question: whether stronger updates instead interfere with capabilities already present in the pretrained model.

Stronger updates degrade pretrained capabilities. Unlike a dedicated recurrent memory initialized from an empty state, the fast weights in In-Place TTT begin from pretrained model parameters and therefore serve two roles simultaneously: they implement pretrained computation while also acting as writable state at test time. MLP layers have also been associated with storing parametric knowledge acquired during pretraining (Geva et al., 2021; Dai et al., 2022). Larger fast-weight updates may therefore interfere with existing model capabilities even when they remain useful for storing contextual information.

To isolate this effect, we use a 128K-token NarrativeQA passage (Kočiský et al., 2018) solely as an update-inducing context; the content contains no information needed to answer them. After processing this context and accumulating fast-weight updates, we evaluate on knowledge-heavy or commonsense-reasoning benchmarks—MMLU, ARC-Challenge, HellaSwag, and PIQA—using the resulting fast-weight state under the same amplification factor λ. As shown in Figure 2b, `GDN` progressively loses accuracy as λ increases, despite remaining robust on RULER. `GDN-Residual`

Table 3: **On RULER, query-aware prompting increases TTT-Benefit, while TTT-Effect remains small.** Notations are the same as Table 2. TTT-Average reports the mean across the six TTT-trained variants. Detailed per-variant results in Table 5 in Appendix C.2

Task	Variant	16K			32K			64K			128K		
		Perf.	Ben.	Eff.	Perf.	Ben.	Eff.	Perf.	Ben.	Eff.	Perf.	Ben.	Eff.
RULER(CQ)	Qwen3-4B-Base	88.5	–	–	82.2	–	–	64.8	–	–	51.2	–	–
	`standard-CPT`	89.4	–	–	84.3	–	–	78.5	–	–	69.3	–	–
	TTT-Average	89.6	0.2	0.0	84.2	-0.2	0.0	78.8	0.3	0.1	68.1	-1.2	-1.1
RULER(QCQ)	Qwen3-4B-Base	83.6	–	–	80.9	–	–	71.7	–	–	52.5	–	–
	`standard-CPT`	86.4	–	–	83.2	–	–	77.4	–	–	72.2	–	–
	TTT-Average	87.3	0.9	0.0	83.9	0.7	0.1	77.3	-0.1	-0.2	72.6	0.4	-0.1

degrades more slowly, suggesting that applying decay only to the residual fast-weight state better preserves pretrained computation.

Together, these results reveal a trade-off that retrieval evaluation alone does not expose: for prediction-error-based learning rules, fast-weight updates can be amplified beyond their learned scale without harming retrieval, yet stronger rewriting increasingly disrupts pretrained capabilities.

5 Disentangling Outer-Loop and Inner-Loop Contributions

Having characterized how outer-loop training shapes fast-weight updates in Section 4, we now turn to a setting where TTT-Benefit is more pronounced and ask whether these gains arise from the inference-time update itself or from changes induced by TTT-aware outer-loop training.

5.1 Query-Aware Writing Improves TTT Benefit

Under the standard RULER format, the model processes the context before seeing the retrieval query(CQ). Fast-weight updates are therefore written without knowing which information will later be queried, potentially limiting their utility. Motivated by prior work showing that presenting the query before the document can facilitate memory selection (Arora et al., 2024), we introduce a query-aware format (QCQ), where the query is presented before the context and repeated at the end for answer generation. This allows fast-weight updates to be conditioned on the query while processing the context. We use the same model trained with TTT as in Section 4.1, modifying only the prompt format. Detailed setup is provided in Appendix B.5.

Table 3 compares CQ and QCQ, averaged across six TTT-trained variants. With QCQ, mean TTT-Benefit increases from $+0.2$ to $+0.9$ at 16K, from -0.2 to $+0.7$ at 32K, and most markedly from -1.2 to $+0.4$ at 128K, but decreases from $+0.3$ to -0.1 at 64K. Meanwhile, mean TTT-Effect remains small under QCQ, ranging from -0.2 to $+0.1$. Detailed per-variant results are provided in Appendix C.2. These gains, despite the limited direct effect of inference-time updates, motivate us to examine where the additional TTT-Benefit originates.

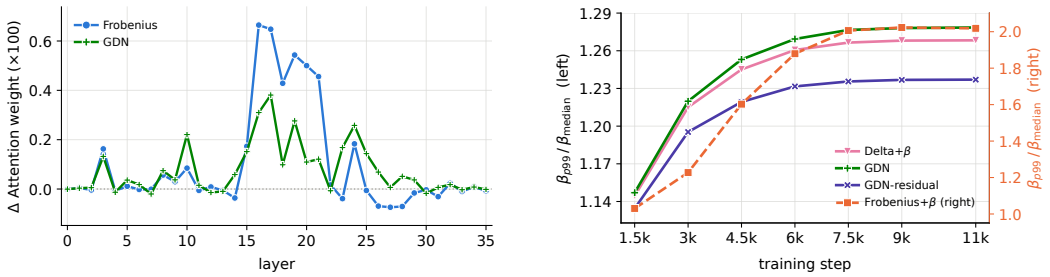
5.2 Disentangling the Sources of TTT-Benefit

The results above reveal an important distinction: query-aware prompting increases TTT-Benefit, but the corresponding TTT-Effect remains comparatively small. To identify where this gain comes from, we decompose TTT-Benefit as:

$$\underbrace{S_{\text{ON}} - S_{\text{CPT}}}_{\text{TTT-Benefit}} = \underbrace{S_{\text{OFF}} - S_{\text{CPT}}}_{\text{outer-loop effect}} + \underbrace{S_{\text{ON}} - S_{\text{OFF}}}_{\text{inner-loop effect(TTT-Effect)}},$$

where S denotes the score. This separates changes learned during TTT-aware training from the direct contribution of the fast-weight update. We analyze these two components in turn.

Outer-loop effect: TTT-aware training strengthens query-relevant attention. We first characterize the outer-loop effect, $S_{\text{OFF}} - S_{\text{CPT}}$, by asking whether TTT-aware training changes how the Transformer retrieves relevant information even when fast-weight updates are disabled.



(a) Query-to-needle attention.

(b) Write-gate selectivity.

Figure 3: **TTT-aware training strengthens query-relevant attention and increases write-gate selectivity.** (a) Relative change in query-to-needle attention for TTT-OFF compared with `standard-CPT` under QCQ, showing stronger attention to evidence even with fast-weight updates disabled. (b) Write-gate selectivity increases during training. Frobenius+β uses the right axis.

For each RULER needle-in-a-haystack example, we measure the Attention weight from the final query tokens to the relevant needle and compare the TTT-trained model with updates disabled (TTT-OFF) against `standard-CPT`. Experiment setup is presented in Appendix B.7. Figure 3a shows the difference in query-to-needle attention weights between TTT-OFF and `standard-CPT` (TTT-OFF $-$ `standard-CPT`). Across layers, TTT-OFF assigns greater Attention weight to the relevant needle, with an average relative increase of approximately 3.9% and up to 8.0% in intermediate layers. As shown in Section 5.1, TTT-aware training improves performance under QCQ while the direct TTT-Effect remains small. The stronger query-to-needle attention in TTT-OFF therefore provides evidence that the outer-loop component of this gain is accompanied by changes in the underlying Transformer's retrieval behavior.

Inner-loop effect: updates remain selective and query-aware despite their small direct effect. We next examine the inner-loop effect, $S_{\mathrm{ON}} - S_{\mathrm{OFF}}$. Although its direct contribution to performance remains small, the learned fast-weight updates are not merely noise. Because β directly controls the contribution of each token to the fast-weight update, it provides a token-level view of which information the inner-loop writer prioritizes. Although the average write gate β decreases over training (Figure 1c), its distribution becomes increasingly selective. We measure **selectivity** within each 4K-token chunk as $\mathrm{P}_{99}(\beta)/\,\mathrm{median}(\beta)$, where $\mathrm{P}_k(\beta)$ denotes the k-th percentile of the token-wise β values(i.e., $\mathrm{median}(\beta) = \mathrm{P}_{50}(\beta)$), and average the resulting ratio across chunks. As shown in Figure 3b, this selectivity ratio steadily increases, indicating that outer-loop training suppresses overall writing while concentrating the remaining write weight on a smaller subset of tokens.

We further ask whether this selectivity is aligned with query-relevant information. For the needle-in-a-haystack subsets of RULER, we measure **target enrichment** as $\mathrm{mean}(\beta_{\mathrm{needle}})/\,\mathrm{median}(\beta)$ within each chunk. Under QCQ, target enrichment increases from 1.169 to 1.192 compared with CQ. Although modest, this increase suggests that providing the query before the context slightly redirects the learned writer toward relevant evidence.

Summary. Together, these analyses suggest that the increased TTT-Benefit under QCQ arises primarily from changes learned during TTT-aware outer-loop training, while the inner-loop update remains selective and query-aware despite its small direct effect on performance. Thus, a small TTT-Effect does not necessarily imply indiscriminate fast-weight writing.

6 CONCLUSION

We studied how test-time training behaves when pretrained parameters are repurposed as fast weights. Across learning rules, training progressively reduces the influence of fast-weight updates, while amplifying them can preserve retrieval performance yet degrade pretrained capabilities. Much of the gain from TTT-aware training can instead come from changes to the underlying model, with inference-time updates contributing little directly. These findings highlight the challenge of turning pretrained parameters into writable memory while preserving their existing capabilities.

AI USE STATEMENT

We used generative AI tools to assist with language editing, rewriting, and the presentation of the manuscript. AI-generated suggestions were reviewed and revised by the authors, and all technical statements, experimental results, analyses, and citations were independently verified by the authors. We take full responsibility for the final content of this work, including text or other artifacts produced with the aid of generative AI.

REFERENCES

Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: closing the recall gap for recurrent language models, 2024. URL https://arxiv.org/abs/2407.05483.

Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past, 2016a. URL https://arxiv.org/abs/1610.06258.

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016b. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/9f44e956e3a2b7b5598c625fcc802c36-Paper.pdf.

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

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *arXiv preprint arXiv:2512.24695*, 2025b.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *Advances in Neural Information Processing Systems*, 38:113506–113543, 2026.

Aviv Bick, Kevin Li, Eric P. Xing, J Zico Kolter, and Albert Gu. Transformers to SSMs: Distilling quadratic knowledge to subquadratic models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=FJlrSZBMCD.

Aviv Bick, Eric P. Xing, and Albert Gu. Retrieval-aware distillation for transformer-SSM hybrids. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=8gsYox51jQ.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In Tsung-Hsien Wen, Asli Celikyilmaz, Zhou Yu, Alexandros Papangelis, Mihail Eric, Anuj Kumar, Iñigo Casanueva, and Rushin Shah (eds.), *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5/.

Aditya Chattopadhyay, Elvis Nunez, Prannay Kaul, Benjamin Bowman, Evan Becker, Luca Zancato, David Thomas, Wei Xia, and Stefano Soatto. Priming: Hybrid state space models from pre-trained transformers, 2026. URL https://arxiv.org/abs/2605.08301.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581. URL https://aclanthology.org/2022.acl-long.581.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ztn8FCR1td.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024. URL https://arxiv.org/abs/2402.19427.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, ZIJIA CHEN, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Celine Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=A1ztozypga.

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. URL https://openreview.net/forum?id=dTWfCLSoyl.

Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=COZDy0WYGg.

F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pp. 850–855 vol.2, 1999. doi: 10.1049/cp:19991218.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid

El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy,

Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Dongchen Han, Yining Li, Tianyu Li, Zixuan Cao, Ziming Wang, Jun Song, Yu Cheng, Bo Zheng, and Gao Huang. Vit^3: Unlocking test-time training in vision. *arXiv preprint arXiv:2512.01643*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daúme III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023/.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1311–1316, 2019.

Barak Lenz, Opher Lieber, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, Daniel Gissin, Daniel Jannai, Dor Muhlgay, Dor Zimberg, Edden M. Gerber, Elad Dolev, Eran Krakovsky, Erez Safahi, Erez Schwartz, Gal Cohen, Gal Shachaf, Haim Rozenblum, Hofit Bata, Ido Blass, Inbal Magar, Itay Dalmedigos, Jhonathan Osin, Julie Fadlon, Maria Rozman, Matan Danos, Michael Gokhman, Mor Zusman, Naama Gidron, Nir Ratner, Noam Gat, Noam Rozen, Oded Fried, Ohad Leshno, Omer Antverg, Omri Abend, Or Dagan, Orit Cohavi, Raz Alon, Ro'i Belson, Roi Cohen, Rom Gilad, Roman Glozman, Shahar Lev, Shai Shalev-Shwartz, Shaked Haim Meirom, Tal Delbari, Tal Ness, Tomer Asida, Tom Ben Gal, Tom Braude, Uriya Pumerantz, Josh Cohen, Yonatan Belinkov, Yuval

Globerson, Yuval Peleg Levy, and Yoav Shoham. Jamba: Hybrid transformer-mamba language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JFPaD7lpBD.

Xin Li and Dan Roth. Learning question classifiers. In *Coling 2002: The 19th international conference on computational linguistics*, 2002.

Zeman Li, Ali Behrouz, Yuan Deng, Peilin Zhong, Praneeth Kacham, Mahdi Karami, Meisam Razaviyayn, and Vahab Mirrokni. Tnt: Improving chunkwise training for test-time memorization. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 116868–116886, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/bdb30687f1c2255c29b11b0b45204ebe-Paper-Conference.pdf.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and qiang liu. Longhorn: State space models are amortized online learners. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8jOqCcLzeO.

Junchen Liu, Sven Elflein, Or Litany, Zan Gojcic, and Ruilong Li. Test-time training with KV binding is secretly linear attention. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=MpBeXMurIb.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. Benchmarking natural language understanding services for building conversational agents. In *Increasing naturalness and flexibility in spoken dialogue interaction: 10th international workshop on spoken dialogue systems*, pp. 165–183. Springer, 2021.

Xuan Ouyang, Zefan Cai, and Junjie Hu. Test-time training with next-token prediction, 2026. URL https://arxiv.org/abs/2606.21803.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Liangzu Peng, Aditya Chattopadhyay, Luca Zancato, Elvis Nunez, Wei Xia, and Stefano Soatto. Gated kalmanet: A fading memory layer through test-time ridge regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20212–20222, June 2026.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers, 2021. URL https://arxiv.org/abs/2102.11174.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

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 Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

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

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

Zixuan Wang, Xingyu Dang, Rui-Jie Zhu, Zixin Wen, Hengyu Fu, Wenhao Chai, and Jason D. Lee. Learning what to remember: Test-time training via context distillation, 2026. URL https://arxiv.org/abs/2608.01672.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 115491–115522. Curran Associates, Inc., 2024. doi: 10.52202/079017-3668. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d13a3eae72366e61dfdc7eea82eeb685-Paper-Conference.pdf.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=r8H7xhYPwz.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context models effectively and thoroughly. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=293V3bJbmE.

Luca Zancato, Arjun Seshadri, Yonatan Dukler, Aditya Golatkar, Yantao Shen, Benjamin Bowman, Matthew Trager, Alessandro Achille, and Stefano Soatto. B'MOJO: Hybrid state space realizations of foundation models with eidetic and fading memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=RnQdRY1h5v.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

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. *arXiv preprint arXiv:2505.23884*, 2025.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.814. URL https://aclanthology.org/2024.acl-long.814/.

A Related Work

A.1 Test-Time Training

Test-time training (TTT) treats inference over a sequence as an online learning process, updating part of the model on the observed context using a self-supervised objective. Originally introduced for adaptation to distribution shift (Sun et al., 2020), TTT has recently been extended to sequence modeling, where the hidden state itself can be represented as a learnable model (Sun et al., 2025). This view is closely related to fast weights and associative memory, where rapidly changing parameters provide temporary storage for recent inputs (Ba et al., 2016a; Schlag et al., 2021).

Recent work has explored richer test-time memory mechanisms (Behrouz et al., 2026; 2025a), scalable update procedures (Zhang et al., 2025; Li et al., 2026), and end-to-end objectives based on next-token prediction (Tandon et al., 2025). More recently, TTT has been adapted to pretrained LLMs by repurposing existing parameters as fast weights (Feng et al., 2026), and subsequent works reconsider the objective with next-token prediction (Ouyang et al., 2026) or with distillation from a long-context teacher (Wang et al., 2026). Rather than proposing a single new TTT mechanism, we study how such fast weights are learned and utilized when introduced into pretrained LLMs.

A.2 Recurrent Sequence Models as Online Learners

Recent state-space and hybrid architectures have explored efficient mechanisms for retaining and retrieving contextual information (Gu & Dao, 2024; Dao & Gu, 2024; Fu et al., 2023; Lenz et al., 2025; Dong et al., 2025; De et al., 2024; Zancato et al., 2024). Linear attention can be formulated as a recurrent model in which contextual information is compressed into a fixed-size state through key–value associations (Katharopoulos et al., 2020; Schlag et al., 2021). This recurrent state update can in turn be viewed as an online learning process, with different learning objectives giving rise to different update rules (Liu et al., 2025). Recent recurrent models have developed increasingly expressive variants of these updates. Mamba-2 (Dao & Gu, 2024) introduces input-dependent decay, DeltaNet (Schlag et al., 2021; Yang et al., 2024) uses prediction-error-based updates, Gated DeltaNet (Yang et al., 2025b) combines the two mechanisms, and GKA (Peng et al., 2026) considers the full history through Kalman-filter-based online ridge regression (Peng et al., 2026). Controlled comparisons in Priming (Chattopadhyay et al., 2026) show that more expressive update rules lead to better downstream long-context performance.

This perspective is closely connected to TTT: recent work shows that a broad class of TTT methods based on key–value binding can be expressed as learned linear attention (Liu et al., 2026). We build on this connection by transferring recurrent-state learning rules to the fast weights of pretrained LLMs. Unlike the dedicated recurrent states in these models, however, In-Place TTT (Feng et al., 2026) writes directly into pretrained MLP parameters, creating an additional need to preserve their existing function.

B Experiment Setup

B.1 Model

We train Qwen3-4B-Base (Yang et al., 2025a) and LLaMA-3.1-8B (Grattafiori et al., 2024). Following In-place TTT, we repurpose the MLP down projection in every sixth Transformer layer (layers $0, 6, \ldots, 30$) as fast weights, with an inner-loop chunk size of 4,096 tokens and a TTT learning rate of 3. Across all variants, the reconstruction target is constructed from the MLP input using a local convolution followed by a learned linear projection, following In-Place TTT.

Initialization All variants share the same backbone weights but only the modules that In-Place TTT and our update rules add – the depthwise target convolution W_{conv}, the target projection W_{proj}, and (where present) the forget-gate and write-gate projections W_α, W_β –are freshly initialized.

The initialization depends on the update rule. For the additive objectives (`Frobenius` and `Frobenius+`β), we follow In-Place TTT and zero-initialize W_{conv}; for prediction-error objectives, we initialize it with small Gaussian noise. Gate projections are zero-initialized, making the gates input-independent at initialization, with $\beta_0 = 0.90$ for `Frobenius+`β and 0.119 for `Delta+`β/`GDN`/`GDN-Residual`, and $\alpha_0 = 0.9975$ for `GDN` and 0.982 for `GDN-Residual`. The gates become input-dependent through training.

B.2 Training

Following In-Place TTT (Feng et al., 2026), we continue pre-training on NVIDIA H100 80 GB GPUs using FSDP2, bf16 mixed precision, and gradient checkpointing. All runs use AdamW (peak learning rate 5×10^{-6}, cosine decay to 10% of peak, 2% linear warmup, weight decay 0.1, gradient-norm clip 1.0) with a global batch size of 64 sequences.

Qwen3-4B-Base is trained in two stages on the long-context corpus. **Stage 1** uses 32K contexts for 11K steps (23B tokens). **Stage 2** continues from each Stage-1 checkpoint at 128K-length for 2K steps (17B tokens), using a fresh optimizer and YaRN RoPE scaling (factor 4) (Peng et al., 2024). Due to the higher computational cost, we train LLaMA-3.1-8B only with Stage 1.

B.3 Training Throughput

We additionally measure observed training throughput using the median throughput between steps 1K and 3.5K under the same Stage-1 training configuration. `Frobenius` and `Frobenius+`β achieve nearly identical throughput (72.0K tokens/s), suggesting little additional overhead from the write gate in our implementation. The prediction-error variants achieve 52.8K (`Delta`), 42.1K (`Delta+`β), 48.3K (`GDN`), and 49.0K (`GDN-Residual`) tokens/s, corresponding to a 27–42% reduction relative to `Frobenius`. These measurements reflect our implementation and are not intended as optimized systems benchmarks.

B.4 Evaluation

We evaluate long-context performance using RULER (Hsieh et al., 2024) and the long-document QA and many-shot ICL categories of HELMET (Yen et al., 2025). RULER provides controlled synthetic tasks for measuring retrieval and reasoning over long contexts, while HELMET evaluates more realistic downstream applications. We additionally evaluate standard multiple-choice common-sense reasoning benchmarks to measure whether accumulated fast-weight updates interfere with pretrained capabilities. For models with a native 32K context window, we apply YaRN scaling (factor 2) at 64K.

RULER RULER is a synthetic long-context benchmark comprising 13 tasks spanning needle-in-a-haystack retrieval, multi-hop tracing, aggregation, and question answering. We evaluate at context lengths $\{16\text{k}, 32\text{k}, 64\text{k}, 128\text{k}\}$ with 100 samples per task and length. Generation is capped at 64 tokens, and we report exact-match accuracy averaged across tasks.

HELMET-QA. We use the two ∞BENCH (Zhang et al., 2024) tasks from HELMET's long-document QA category. Both require answering questions about book-length contexts: ∞Bench QA uses short free-form answers and is scored with ROUGE-L F1, while ∞Bench MC requires selecting the correct answer from multiple choices and is scored by accuracy. Generation is capped at 10 tokens, following the HELMET setup.

HELMET-ICL. We evaluate many-shot in-context classification on five datasets: TREC-Coarse, TREC-Fine (Li & Roth, 2002), BANKING77 (Casanueva et al., 2020), CLINC150 (Larson et al., 2019), and NLU (Liu et al., 2021). These tasks span question and intent classification with label spaces ranging from 6 to 151 classes. Labels are represented by arbitrary integer IDs, requiring the model to infer the label mapping from demonstrations rather than relying on semantic label names. Generation is capped at 20 tokens, and we report classification accuracy using substring exact match on the predicted label.

Pretrained-capability evaluation. To measure interference with capabilities that do not require information from the long context, we additionally evaluate MMLU (Hendrycks et al., 2021), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), and ARC-Challenge (Clark et al., 2018). For the preservation analysis, the model first processes an irrelevant 128K long context from ∞BENCH (Zhang et al., 2024) and is then evaluated on these multiple-choice tasks with the updated fast-weight state. We report average accuracy across the four benchmarks.

B.5 Query-aware Prompting

For the query-bearing RULER categories (QA, NIAH, variable tracking) we use a *question–context–question* (QcQ) layout: the per-sample question is front-loaded *before* the context in addition to the standard trailing question, while the trailing half is verbatim the stock RULER template so answer prefixes and scoring are unchanged. Tasks without a per-sample query (frequent-and common-word extraction) keep the upstream template. The QA template is:

```
Answer the question based on the given documents. Only give
me the answer and do not output any other words.
Question: {query}
The following are given documents.
{context}
Answer the question based on the given documents. Only give
me the answer and do not output any other words.
Question: {query} Answer:
```

NIAH and variable-tracking use the analogous QcQ form (the retrieval / tracking instruction and its query are repeated before and after {context}).

B.6 Fast-Weight Dynamics

We evaluate each checkpoint on 260 prompts spanning all 13 RULER tasks at 32K context length. For each prompt, we perform a single prefill pass and average the measured statistics over all complete 4,096-token TTT chunks and TTT layers.

B.7 Attention Analysis

For each RULER NIAH prompt, we identify the needle tokens as those covering the gold-answer span. At each attention layer, we measure attention from the final $K = 16$ query positions to all preceding needle positions and average over the K queries. We define *needle attention* as the total attention weight assigned to the needle tokens and average it over 100 prompts for each of the eight NIAH tasks(800 prompts in total).

C Results

C.1 Results on LLaMA-3.1-8B

Table 4 reports the corresponding results on LLaMA-3.1-8B-Base. Because it is trained only at 32K context length, we evaluate it up to 64K. On RULER, TTT-Benefit remains small and inconsistent across learning rules and context lengths, ranging from -0.4 to $+0.5$ points. TTT-Effect is also small, with all values between -0.4 and $+0.2$ points.

HELMET shows the same qualitative separation between TTT-Benefit and TTT-Effect. All evaluated TTT variants underperform the `standard-CPT` baseline, while the direct effect of enabling fast-weight updates remains modest, ranging from $+0.4$ to -1.5 points. Overall, the LLaMA results reproduce the main observation from Qwen3-4B: training with fast-weight updates does not imply strong reliance on them at inference time.

C.2 Detailed Results on Query-Aware Prompting

Table 5 shows that QCQ substantially increases TTT-Benefit, particularly at 128K. For example, TTT-Benefit changes from $-0.4 \rightarrow +1.2$ for `Frobenius`, $-1.0 \rightarrow +0.5$ for `Delta+`β, $-1.0 \rightarrow +0.1$ for `GDN`, and $-0.6 \rightarrow +0.7$ for `GDN-Residual`. Despite these larger gains over `standard-CPT`, TTT-Effect remains small, ranging only from -0.4 to $+0.3$. Thus, providing the query before writing increases the benefit of TTT-aware training without increasing the direct contribution of the inference-time fast-weight update.

C.3 Stage-1 Results on Qwen3-4B

Table 6 reports RULER results of training Qwen3-4B-Base only with Stage 1 (32K-length). TTT-Benefit remains mixed across learning rules and context lengths. TTT-Effect is generally small, particularly at 16K and 64K, with `Delta` at 32K ($+1.4$ points) as the largest positive effect. Thus, the limited reliance on inference-time fast-weight updates observed in the main results is already present after Stage 1 and is not specific to the 128K continuation stage.

Table 4: **Long-context performance of LLaMA3.1-8B-Base across inner-loop learning rules. Perf.** (Performance) denotes accuracy; for TTT-trained models, it is measured with inference-time fast-weight updates enabled. **Ben.** (TTT-Benefit) is improvement over the separately trained `standard-CPT` baseline, while **Eff.** (TTT-Effect) measures the performance difference between enabling (TTT-ON) and disabling (TTT-OFF) the inference-time update within the same TTT-trained model.

Variant	16K			32K			64K		
	Perf.	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**
RULER									
LLaMA3.1-8B-Base	92.8	–	–	89.9	–	–	81.2	–	–
`standard-CPT`	93.5	–	–	90.3	–	–	79.0	–	–
`Frobenius`	93.3	-0.2	-0.1	90.7	0.4	0.1	78.8	-0.2	-0.4
`Delta+`β	93.2	-0.3	-0.3	90.0	-0.3	0.0	79.5	0.5	0.0
`GDN`	93.7	0.2	-0.2	89.9	-0.4	-0.2	79.2	0.2	0.2
HELMET (QA + ICL)									
LLaMA3.1-8B-Base	52.3	–	–	54.2	–	–	56.4	–	–
`standard-CPT`	49.9	–	–	51.9	–	–	54.2	–	–
`Frobenius`	49.8	-0.1	0.4	51.5	-0.4	-0.1	52.9	-1.3	0.3
`Delta+`β	49.3	-0.6	0.3	50.1	-1.8	-0.8	52.2	-2.0	-1.5
`GDN`	48.8	-1.1	-0.1	50.6	-1.2	-0.8	52.4	-1.8	-1.4

Table 5: **Query-aware prompting increases TTT-Benefit, while TTT-Effect remains small. Perf.** (Performance) denotes accuracy; for TTT-trained models, it is measured with inference-time fast-weight updates enabled. **Ben.** (TTT-Benefit) is improvement over the separately trained `standard-CPT` baseline, while **Eff.** (TTT-Effect) measures the performance difference between enabling (TTT-ON) and disabling (TTT-OFF) the inference-time update within the same TTT-trained model. Average rows report the mean across the six TTT-trained variants.

Variant	16K			32K			64K			128K		
	Perf.	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**
Qwen3-4B-Base	83.6	–	–	80.9	–	–	71.7	–	–	52.5	–	–
`standard-CPT`	86.4	–	–	83.2	–	–	77.4	–	–	72.2	–	–
`Frobenius`	87.5	1.1	0.1	83.4	0.2	0.0	77.5	0.1	-0.2	73.4	1.2	-0.4
`Frobenius+`β	87.2	0.8	0.1	84.4	1.2	0.4	77.7	0.3	0.1	71.8	-0.4	-0.4
`Delta`	86.9	0.5	-0.1	83.4	0.2	0.1	77.2	-0.2	-0.3	72.6	0.4	-0.5
`Delta+`β	87.3	0.9	0.0	83.7	0.5	0.2	77.0	-0.4	-0.3	72.7	0.5	0.3
`GDN`	87.2	0.8	0.0	84.2	1.0	0.0	77.3	-0.1	-0.1	72.3	0.1	0.2
`GDN-Residual`	87.7	1.3	0.1	84.1	0.9	-0.2	77.2	-0.2	-0.3	72.9	0.7	0.1
Average	87.3	0.9	0.0	83.9	0.7	0.1	77.3	-0.1	-0.2	72.6	0.4	-0.1

C.4 PER-CATEGORY HELMET RESULTS

Table 7 reports HELMET-QA and HELMET-ICL separately, rather than averaging the two categories as in Table 2. Both categories show the same qualitative pattern as the aggregate results: TTT-Benefit varies substantially across learning rules and context lengths, while TTT-Effect is generally smaller and inconsistent in sign. The strongest gains occur around 64K for several Frobenius and MSE variants, but do not consistently persist at 128K.

Table 6: **RULER performance of Qwen3-4B-Base after Stage 1 training. Perf.** (Performance) denotes accuracy; for TTT-trained models, it is measured with inference-time fast-weight updates enabled. **Ben.** (TTT-Benefit) is improvement over the separately trained `standard-CPT` baseline, while **Eff.** (TTT-Effect) measures the performance difference between enabling (TTT-ON) and disabling (TTT-OFF) the inference-time update within the same TTT-trained model.

Variant	16K			32K			64K		
	Perf.	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**
Qwen3-4B-Base	88.5	–	–	82.2	–	–	64.8	–	–
`standard-CPT`	90.0	–	–	81.5	–	–	68.1	–	–
`Frobenius`	89.8	-0.2	-0.1	82.0	0.5	0.0	67.4	-0.7	-0.3
`Frobenius+`β	89.7	-0.3	0.1	81.1	-0.4	0.0	67.1	-1.0	-0.4
`Delta`	89.8	-0.2	0.1	82.3	0.8	1.4	66.7	-1.4	-0.3
`Delta+`β	89.6	-0.4	0.1	81.7	0.2	0.2	67.4	-0.7	-0.6
`GDN`	89.7	-0.3	0.0	82.1	0.6	0.6	67.1	-1.0	-0.6
`GDN-Residual`	89.6	-0.4	0.1	81.4	-0.1	0.4	67.2	-0.9	-0.3

Table 7: **HELMET performance for each subtask for Qwen3-4B-Base. Perf.** (Performance) denotes accuracy; for TTT-trained models, it is measured with inference-time fast-weight updates enabled. **Ben.** (TTT-Benefit) is improvement over the separately trained `standard-CPT` baseline, while **Eff.** (TTT-Effect) measures the performance difference between enabling (TTT-ON) and disabling (TTT-OFF) the inference-time update within the same TTT-trained model.

Variant	16K			32K			64K			128K		
	Perf.	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**	**Perf.**	**Ben.**	**Eff.**
HELMET QA												
`standard-CPT`	34.4	–	–	40.4	–	–	41.3	–	–	43.4	–	–
`Frobenius`	35.2	0.8	0.5	39.9	-0.5	0.9	43.4	2.1	0.0	44.4	1.0	-1.1
`Frobenius+`β	34.4	0.0	0.5	39.3	-1.1	0.5	44.9	3.6	1.4	44.4	1.0	-1.3
`Delta`	33.9	-0.5	0.5	39.0	-1.4	-3.6	41.5	0.2	-1.5	45.7	2.3	-0.6
`Delta+`β	32.2	-2.2	-0.5	38.8	-1.6	0.8	40.5	-0.8	-0.9	42.7	-0.7	0.3
`GDN`	34.0	-0.4	0.0	40.6	0.2	0.8	39.6	-1.7	-1.7	44.8	1.4	0.9
`GDN-Residual`	32.7	-1.7	-0.6	37.5	-2.9	-1.5	40.9	-0.4	-0.1	43.6	0.2	-2.6
HELMET ICL												
`standard-CPT`	78.2	–	–	81.3	–	–	81.8	–	–	67.1	–	–
`Frobenius`	79.2	1.0	0.3	81.9	0.6	0.4	84.4	2.6	1.6	66.6	-0.5	0.7
`Frobenius+`β	78.6	0.4	-0.1	82.3	1.0	0.4	83.8	2.0	1.2	66.0	-1.1	0.2
`Delta`	78.8	0.6	-0.2	81.4	0.1	-0.5	84.1	2.3	1.2	66.4	-0.7	-0.6
`Delta+`β	78.0	-0.2	0.1	81.7	0.4	0.1	82.6	0.8	0.3	67.0	-0.1	0.8
`GDN`	75.6	-2.6	0.4	79.9	-1.4	-0.2	81.4	-0.4	0.0	66.8	-0.3	0.4
`GDN-Residual`	78.0	-0.2	-0.2	81.9	0.6	0.0	82.8	1.0	0.0	66.8	-0.3	0.0