DELTATTT: LAYERWISE OPTIMIZATION FOR NONLINEAR RECURRENT MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequential test-time training adapts a memory network through successive updates, each computing an inner-loop gradient based on the network's previous state. Intuitively, this state dependence should allow each update to account for what the memory has already learned and better incorporate new information. However, we find that this expected advantage does not consistently materialize in nonlinear memories: a fixed-base parallel TTT baseline outperforms its serial counterpart. Our exploratory experiments point to a key underlying difficulty: nonlinear memories can be harder to optimize than linear ones within a single pass over the sequence. To alleviate this optimization difficulty, we introduce DeltaTTT, which replaces joint inner-loop optimization of a two-layer memory network with layerwise learning. Each layer is assigned a local prediction target and updated through a state-dependent delta rule. This formulation retains a nonlinear readout while enabling chunkwise parallel computation. Experiments on DeltaNet and LaCT backbones show improvements in language modeling and retrieval over their recurrent baselines.

1 INTRODUCTION

In sequence modeling, Test-Time Training (TTT) compresses past key–value pairs into the parameters of a small inner network through online updates (Sun et al., 2024). By default, the updates are sequential, each computing an inner-loop gradient based on the network's previous state. Intuitively, this state dependence should improve memory learning: each update can account for what the network has already learned and better incorporate new information.

This intuition is supported by results on linear memories. DeltaNet replaces the state-independent additive writes of linear attention with residual corrections based on the current memory, achieving stronger associative recall and supporting the value of state-dependent updates (Yang et al., 2024). However, this expected advantage does not consistently materialize for nonlinear memories. In a LaCT-derived language model (Zhang et al., 2025), we compare serial updates, which evaluate each chunk at the current fast weights, with a fixed-base parallel baseline, which evaluates updates at a shared initial state and accumulates them causally. We find that the parallel baseline achieves better downstream results despite computing its updates without considering earlier writes. This result motivates us to revisit how nonlinear memory is optimized in serial TTT.

To examine memory optimization more directly, we fit standalone linear and two-layer MLP memories to fixed K/V sequences extracted from a pretrained full-attention model, using key-value reconstruction as the objective. A linear memory can achieve lower reconstruction error within a single epoch, while the MLP's advantage emerges with additional epochs. These observations suggest that greater expressivity alone is insufficient: the write rule must also realize that capacity within the limited inner-update budget, usually one epoch for TTT.

We therefore ask: if jointly optimizing a nonlinear memory can be less effective than optimizing a linear one within a single pass, can we optimize it more effectively layer by layer? This motivates DeltaTTT, which retains a nonlinear multilayer memory while equipping each linear layer with a local reconstruction target and its own delta-rule update. Given its inputs and targets, each layer admits efficient chunkwise parallel evaluation, while the nonlinear composition and dependencies between layers are preserved. This design aims to combine the expressive capacity of nonlinear memories with effective single-pass optimization and the computational efficiency of the delta rule.

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.

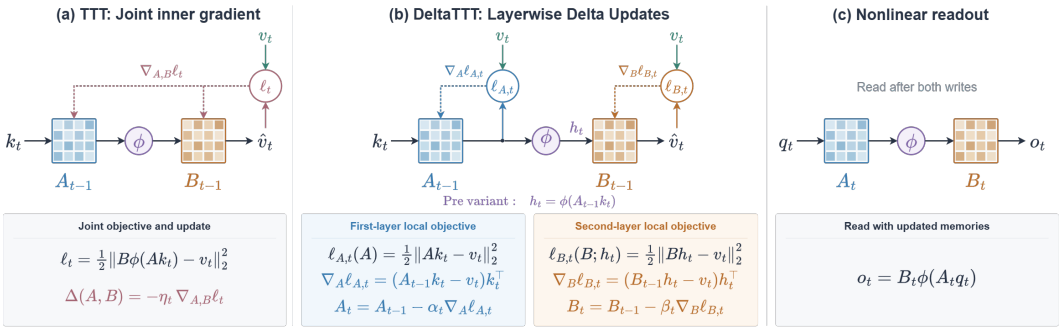

Figure 1: **Joint and layerwise optimization of nonlinear recurrent memory.** (a) TTT jointly optimizes both layers through a shared reconstruction objective. (b) DeltaTTT assigns a local reconstruction objective to each layer and performs layerwise delta-rule updates. (c) Both methods use the nonlinear readout after the writes.

Our contributions are threefold. 1) We show that serial nonlinear TTT can underperform fixed-base parallel TTT, and use fixed-K/V reconstruction experiments to highlight the difficulty of optimizing nonlinear memories within a single pass. 2) To alleviate this optimization difficulty, we propose DeltaTTT, which replaces joint inner-loop optimization of a two-layer memory network with layerwise learning, assigning each layer a local prediction target and a state-dependent delta-rule update. This formulation retains a nonlinear readout and enables chunkwise parallel computation while preserving exact tokenwise updates. 3) We evaluate DeltaTTT with 0.5B-parameter language models trained on 92.5B tokens, demonstrating improvements in language modeling and retrieval over recurrent baselines on both DeltaNet and LaCT backbones.

2 Background

Test-Time Training. Test-Time Training (TTT) represents sequence context through the parameters W of an inner network f_W, typically an MLP(Sun et al., 2024; Han et al., 2026). Given query, key, and value features (Q, K, V), TTT compresses key–value associations into W through inner-loop optimization:

$$W \leftarrow W - \eta \nabla_W \mathcal{L}\big(f_W(K), V\big), \tag{1}$$

where $\mathcal{L}$ is the inner learning objective and η is the learning rate. Queries retrieve information from the resulting memory through

$$O = f_W(Q). \tag{2}$$

As the sequence is processed, these updates allow the inner network to accumulate context in a fixed-size parameter state. With a fixed-size inner network and a single pass over the sequence, the total read and update cost scales linearly with sequence length.

Serial and fixed-base parallel TTT. Our serial baseline follows the chunkwise TTT formulation of LaCT (Zhang et al., 2025). Let $\mathcal{L}_c(W)$ denote the inner loss on chunk c evaluated at memory parameters W. We maintain an unnormalized accumulator S_c and a normalized memory W_c, initialized with $S_0 = W_0$. For clarity, we omit momentum. Serial TTT evaluates each gradient at the memory updated by the preceding chunks, whereas fixed-base parallel TTT evaluates every gradient at the shared initialization:

$$\text{Serial:} \quad S_c = S_{c-1} - \eta \nabla \mathcal{L}_c(W_{c-1}), \qquad W_c = \mathcal{N}(S_c), \tag{3}$$

$$\text{Fixed-base:} \quad S_c = S_{c-1} - \eta \nabla \mathcal{L}_c(W_0), \qquad W_c = \mathcal{N}(S_c), \tag{4}$$

where η is the inner learning rate and $\mathcal{N}$ means normalization. Serial gradients depend on preceding updates, while fixed-base gradients can be computed independently across chunks and accumulated through a causal prefix sum. Both baselines read before writing the current chunk, using $o_t = f_{W_{c-1}}(q_t)$ for tokens t in chunk c.

Table 1: Serial and parallel TTT with SwishGLU inner models, 0.5B parameters, and 92.5B training tokens. Single denotes mean single-needle retrieval accuracy across S1/S2/S3; 32k denotes the mean accuracy across S1/S2/S3 at 32k context length.

Update rule	Loss ↓	Single ↑	32k ↑
Serial	1.99	28.25	11.13
Parallel	1.98	37.87	30.87

The linear case: DeltaNet and Linear Attention. The same framework recovers DeltaNet by using a linear memory $f_W(k) = Wk$, the squared loss $\ell(y, v) = \frac{1}{2}\|y - v\|_2^2$, and one token per update (chunk size=1). Indexing states by token, serial gradient descent gives

$$\begin{aligned} W_t &= W_{t-1} - \eta_t \nabla \mathcal{L}_t(W_{t-1}) \\ &= W_{t-1} + \eta_t (v_t - W_{t-1} k_t) k_t^\top . \end{aligned} \tag{5}$$

This is precisely the delta-rule update used in DeltaNet. For supplied keys, values, and learning rates, this recurrence is affine in the preceding state, enabling efficient chunkwise parallel evaluation (Yang et al., 2024). DeltaNet thus combines tokenwise update granularity (chunk size=1) with computational parallelism, preserving the exact sequence of state-dependent updates. Applying fixed-base evaluation to the same objective instead yields

$$W_t = W_0 + \sum_{i=1}^{t} \eta_i (v_i - W_0 k_i) k_i^\top . \tag{6}$$

With $W_0 = 0$ and $\eta_i = 1$, this reduces to $W_t = \sum_{i=1}^{t} v_i k_i^\top$, which is exactly linear attention. Thus, DeltaNet and additive linear attention correspond to serial and fixed-base parallel TTT respectively within linear memory.

3 Motivation: Effective Optimization in a Single Pass

Serial TTT is intuitively promising because this state dependence should allow each update to account for what the memory has already learned and better incorporate new information. This expectation is supported by DeltaNet, which uses the residual $v_t - W_{t-1}k_t$ to perform state-dependent correction (Equation 5) and achieves stronger associative recall than additive linear attention (Yang et al., 2024). Surprisingly, we observe the opposite in nonlinear TTT: serial updates achieve a single-needle accuracy of 28.25, compared with 37.87 for fixed-base parallel updates (Table 1). We consider two possible explanations for this discrepancy.

Nonlinear memories can struggle to fit well within a single pass. Greater expressivity does not necessarily translate into better fitting under a limited optimization budget. To examine this, we conduct a toy experiment in which a linear map and a two-layer MLP fit the same fixed K/V sequence extracted from a full-attention model. We compare their reconstruction errors across epochs. As shown in Figure 2 (a), the linear memory can achieve lower reconstruction error after just one epoch, despite the MLP's greater capacity. The MLP attains lower error at small chunk sizes by epoch 16 (Figure 2), suggesting that its capacity advantage requires more optimization to emerge under this protocol. However, existing methods such as LaCT (Zhang et al., 2025), TTT (Sun et al., 2024), and DeltaNet (Yang et al., 2024) typically use a single pass over the context for computational efficiency, as additional epochs of inner learning would incur substantial overhead. These results motivate more effective optimization of nonlinear memories within a single-pass budget. Appendix B provides the experimental details.

Smaller chunks can improve fitting but incur high latency in serial nonlinear TTT. Figure 2 highlights a second constraint: at a fixed replay budget, smaller chunks allow more update steps and can yield lower reconstruction error, although the trend is not strictly monotonic; for the tuned MLP, the benefit is more apparent after multiple passes. A related tradeoff was observed by Sun et al. (2024), who found that smaller TTT mini-batches reduce language-model perplexity at the cost of higher latency. In serial nonlinear TTT, each chunk's gradient depends on the memory updated by preceding chunks, so smaller chunks require more sequential update steps while offering less

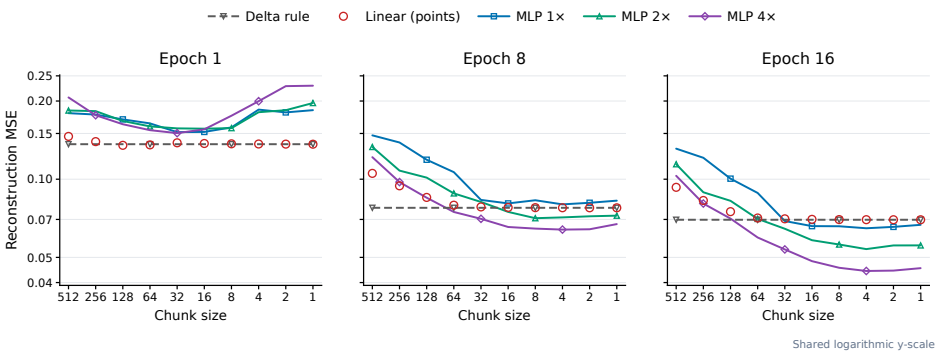

Figure 2: Fitting fixed Full-Attention K/V pairs at different replay budgets.

parallelism within each step. This computational tradeoff motivates LaCT's use of large chunks (Zhang et al., 2025). Linear recurrent memories with delta updates offer a way around this tradeoff. Given the keys and targets, each delta update is affine in the preceding memory state, enabling efficient chunkwise parallel evaluation while preserving the exact sequence of tokenwise corrections (Yang et al., 2024). The logical update granularity can therefore remain $C = 1$ even when the hardware kernel processes many tokens together.

4 DeltaTTT: Layerwise Memory Optimization

Given the difficulty of optimizing a nonlinear multilayer memory as a whole within a single epoch, can we optimize it more effectively layer by layer? This motivates DeltaTTT, which retains a nonlinear multilayer memory while equipping each linear layer with a local target and its own delta update. Given its inputs and targets, each layer admits efficient chunkwise parallel evaluation, while the nonlinear composition and dependencies between layers are preserved. This design aims to combine the expressive capacity of nonlinear memories with effective single-pass optimization and the computational efficiency of the delta rule.

4.1 Layerwise memory optimization

We now introduce DeltaTTT, a layerwise approach to optimizing nonlinear recurrent memory. For one memory head, let $k_t, q_t \in \mathbb{R}^{d_k}$ and $v_t \in \mathbb{R}^{d_v}$ denote the current key, query, and value. The memory W consists of two dynamic matrices $A \in \mathbb{R}^{d_v \times d_k}$ and $B \in \mathbb{R}^{d_v \times d_v}$, connected by $\phi = \mathrm{SiLU}$. Classical TTT jointly optimizes this two-layer memory to reconstruct values from keys:

$$f_{A,B}(x) = B\phi(Ax), \qquad \ell_t(A,B) = \frac{1}{2}\|B\phi(Ak_t) - v_t\|_2^2. \tag{7}$$

Our goal is to make this reconstruction task easier to optimize within a single pass. To this end, we decompose it into two local learning problems, one for each layer. The first layer learns to predict v_t directly from k_t. After updating A, its prediction is passed through the nonlinearity to form an intermediate representation $h_t = \phi(A_t k_t)$. The second layer then learns to reconstruct the same value v_t from h_t. In this way, the first layer makes an initial prediction, and the second provides another learned reconstruction stage that can compensate for imperfections in the first. Their local objectives are

$$\ell_{A,t}(A) = \frac{1}{2}\|Ak_t - v_t\|_2^2, \qquad \ell_{B,t}(B) = \frac{1}{2}\|Bh_t - v_t\|_2^2. \tag{8}$$

Each objective is linear regression in its own matrix, so a gradient step yields a delta-rule update. With learned write rates α_t and β_t, the complete update and readout are

$$A_t = A_{t-1} + \alpha_t(v_t - A_{t-1}k_t)k_t^\top, \tag{9}$$
$$h_t = \phi(A_t k_t), \tag{10}$$
$$B_t = B_{t-1} + \beta_t(v_t - B_{t-1}h_t)h_t^\top, \tag{11}$$
$$o_t = B_t\phi(A_t q_t). \tag{12}$$

Thus, the query still reads from a nonlinear memory. Meanwhile, each linear layer is updated using the delta rule, allowing DeltaTTT to reuse DeltaNet's chunkwise scan algorithm to compute tokenwise updates in parallel within each chunk (Yang et al., 2024). Through layerwise optimization, DeltaTTT combines tokenwise updates with efficient parallel computation for nonlinear memory, while potentially benefiting from the easier optimization of individual linear layers compared with the nonlinear network.

We call formulation above *DeltaTTT-post*, since h_t uses the updated first-layer state. To avoid waiting for the current A update, *DeltaTTT-pre* instead uses $h_t = \phi(A_{t-1}k_t)$, leaving the remaining equations unchanged. Once this representation is available, the two current-token writes can proceed concurrently. Our ablations show comparable performance between Pre and Post, making *DeltaTTT-pre* attractive for its greater potential concurrency. We therefore adopt *DeltaTTT-pre*, as illustrated in Figure 1(b), as our default method.

To clarify how this update and readout scheme relates to existing methods, Table 2 compares the states used to evaluate updates, the granularity of memory evolution, and the states used for query readout. Mini-batch TTT and E^2-TTT retain tokenwise state recurrences, but evaluate gradients at fixed batch- or chunk-start states. LaCT and our fixed-base parallel TTT baseline instead evolve memory at chunk boundaries and use chunk-start states for readout. DeltaNet supports state-dependent tokenwise updates and current-token readout with a linear memory. DeltaTTT combines these properties with a nonlinear memory, using layerwise delta-rule updates to enable chunkwise parallel computation. Pseudocode for the chunkwise computation is provided in Appendix A.

Table 2: Comparison of memory update schedules. Gradient/write evaluated at denotes the fast-weight state used to compute each update; state recurrence denotes the granularity of state evolution; output computed from denotes the state used for query readout. TTT rows assume nonlinear inner models. Parallel TTT adapts fixed-initialization batch-gradient TTT to our chunkwise, read-before-write baseline (Eq. 4).

Method	Gradient/write evaluated at	State recurrence	Output computed from	Nonlinear	Efficient training
Token-wise TTT (Sun et al., 2024)	Previous token	Token-wise	Current token	✓	×
Mini-batch TTT (Sun et al., 2024)	Batch start	Token-wise	Current token	✓	×
Chunk-TTT (Zhang et al., 2025)	Chunk start	Chunk-wise	Chunk start	✓	✓
E^2-TTT (Zhong et al., 2026)	Chunk start	Token-wise	Chunk start	✓	✓
Parallel TTT	Initial state	Chunk-wise	Chunk start	✓	✓
DeltaNet (Yang et al., 2024)	Previous token	Token-wise	Current token	×	✓
DeltaTTT (ours)	Previous token	Token-wise	Current token	✓	✓

5 Experiments

5.1 Experimental setup

We train on 92.5B tokens from the pretraining split of Long-Data-Collections. Our DeltaTTT models contain approximately 0.54B parameters, with 24 layers and a hidden dimension of 1024. Training uses 32K-token sequences and AdamW with a peak learning rate of 3×10^{-4} and cosine decay. We implement DeltaTTT on DeltaNet (Yang et al., 2024) and LaCT (Zhang et al., 2025) backbones by replacing their memory modules with our DeltaTTT module. Baselines include DeltaNet and a variant with a learnable initial memory W_0 ; Gated DeltaNet (Yang et al., 2025); serial LaCT and our fixed-base parallel LaCT variant; and Delta MLP (Irie et al., 2021), which uses layer-specific keys and values generated from the same input token to update a multilayer fast-weight memory via the delta rule. For a fair comparison, all recurrent models use sliding-window attention (SWA) with a 512-token window. Following LaCT's practice, we set both the update chunk size and SWA window size to 512 tokens for all serial and parallel LaCT baselines. A full-attention Transformer (Vaswani et al., 2017) serves as an additional reference. We evaluate single-needle retrieval using RULER (Hsieh et al., 2024) over 4K–32K contexts. Detailed experimental settings are provided in Appendix C.

Table 3: Results grouped by backbone at the 92.5B-token budget. All recurrent configurations use SWA with window 512. Single denotes the mean over S1/S2/S3.

Model	Params (M)	Train loss ↓	Single ↑	S1 ↑					S2 ↑					S3 ↑				
				4K	8K	16K	32K	Avg.	4K	8K	16K	32K	Avg.	4K	8K	16K	32K	Avg.
Full attention																		
Full attention	536.95	1.918769	72.42	100.0	100.0	99.2	95.2	98.60	95.4	91.2	95.6	88.2	92.60	50.0	27.8	22.2	4.2	26.05
DeltaNet backbone																		
DeltaNet	537.64	1.955473	44.38	99.8	98.4	97.2	91.8	96.80	83.0	32.6	10.8	2.4	32.20	7.6	4.2	2.2	2.6	4.15
Gated DeltaNet	562.80	1.967150	47.60	100.0	100.0	100.0	98.8	99.70	76.0	43.6	19.4	5.0	36.00	17.6	7.2	2.2	1.4	7.10
DeltaTTT	541.17	**1.950079**	**48.78**	100.0	100.0	100.0	99.8	**99.95**	93.4	44.0	14.2	4.6	**39.05**	21.4	4.6	1.6	1.8	**7.35**
LaCT backbone																		
DeltaNet	538.13	1.970605	25.00	85.2	64.2	34.0	4.0	46.85	50.0	29.2	10.2	3.4	23.20	11.2	4.2	2.6	1.8	4.95
DeltaNet (learned init)	539.70	1.969120	40.52	100.0	98.8	72.0	21.4	73.05	84.6	54.6	22.2	9.6	42.75	15.0	5.2	1.6	1.2	5.75
Delta MLP	592.10	1.968994	4.88	16.6	5.4	3.2	1.2	6.60	13.8	6.2	1.6	1.8	5.85	4.0	2.2	1.2	1.4	2.20
LaCT Serial SwishGLU	546.98	1.989887	28.25	56.4	52.2	41.4	29.4	44.85	79.4	21.8	9.0	2.2	28.10	32.6	9.0	3.8	1.8	**11.80**
LaCT Parallel SwishGLU	546.98	1.984051	37.87	89.0	94.6	91.0	88.8	90.85	41.8	12.0	7.6	1.8	15.80	16.4	7.2	2.2	2.0	6.95
DeltaTTT	541.67	**1.952344**	**50.47**	99.8	99.6	97.6	69.8	**91.70**	86.6	73.0	34.8	9.6	**51.00**	22.6	7.8	2.6	1.8	8.70

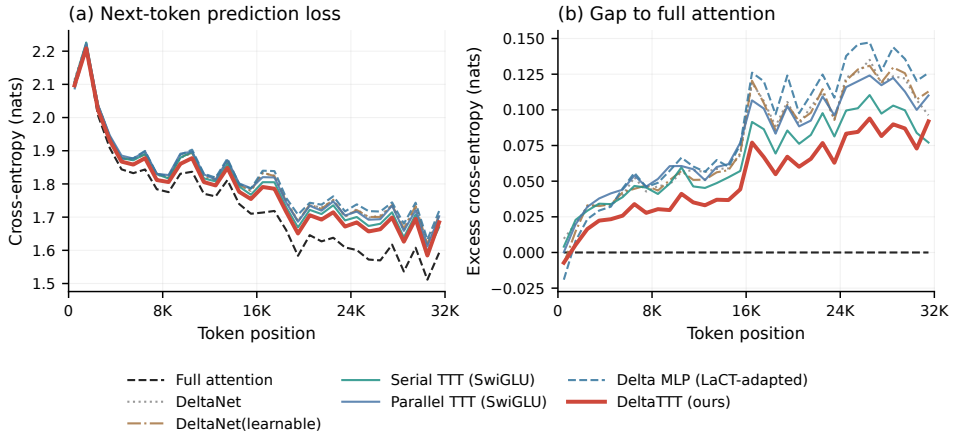

Figure 3: Average Position-wise loss on 43 held-out documents (19 books, 24 papers), each at least 32K tokens long.

5.2 Main Results

Table 3 shows that DeltaTTT improves both training loss and single-needle retrieval across the two backbones. On the LaCT backbone, fixed-base parallel updates outperform serial updates, but DeltaTTT surpasses both while retaining state-dependent correction, including a 12.60 point gain in single-needle retrieval over parallel LaCT. This suggests that a more effective optimization strategy can help realize the potential of nonlinear, state-dependent memory. Improvements over DeltaNet and Gated DeltaNet on the DeltaNet backbone further support the applicability of layerwise optimization across different backbone designs.

5.3 Context Use and Position-Dependent Loss

Position-dependent prediction. We select 19 books and 24 arXiv papers from the Long-Data-Collections, which are excluded from training. Each is long enough to provide 32K prediction targets from a single document prefix. We compute teacher-forced next-token loss for each document, then average losses across documents in 1K-token bins. Figure 3 shows both absolute loss and excess loss relative to full attention. DeltaTTT has the lowest average loss among the evaluated LaCT-backbone recurrent models and reduces the gap to full attention, with a larger advantage over parallel LaCT at later positions. Full attention retains the lowest overall loss.

Table 4: Training efficiency relative to full attention ($1.00\times$).

Model	Throughput/GPU ↑	Peak memory/GPU ↓
Full attention	$1.00\times$	$1.00\times$
DeltaNet	$1.42\times$	$0.53\times$
LaCT Serial (SwiGLU)	$0.83\times$	$0.68\times$
LaCT Parallel (SwiGLU)	$1.07\times$	$0.86\times$
DeltaTTT(ours)	$1.37\times$	$0.69\times$

Table 5: Ablations at 0.5B model and 92.5B training tokens, Removals in (b) are cumulative; Single is the mean single-needle retrieval accuracy across S1/S2/S3.

Model	Configuration	Train loss ↓	Single ↑
(a) Write address			
DeltaTTT-Pre (selected)	Address from A_{t-1}	1.962162	54.32
DeltaTTT-Post	Address from A_t	1.960538	52.22
(b) Progressive component removal			
DeltaTTT (selected)	Full model	1.952344	50.47
DeltaTTT	− B online update	1.967580	36.02
DeltaNet	− Learnable B_0	1.969120	40.52
DeltaNet	− Learnable A_0	1.970605	25.00

5.4 Training efficiency

We measure throughput and peak reserved memory for optimized 0.5B models at a sequence length of 32K at training stage. As shown in Table 4, DeltaTTT is 37% faster than full attention and uses 31% less peak memory. It further outperforms serial and parallel LaCT in throughput by 65% and 28%, respectively, with comparable or lower peak memory usage.

5.5 Ablations

Pre versus Post updates. Table 5(a) compares the Pre and Post variants on the DeltaNet backbone. Pre and Post achieve similar training losses and comparable single-needle retrieval accuracy. Pre constructs the second-layer write address from the pre-update state A_{t-1}, removing its dependence on the current first-layer update and enabling concurrent updates of the two memory modules during inference.

Memory components. Starting from the selected DeltaTTT+SWA model in Table 3, we progressively remove memory components in Table 5(b). We first disable online updates to B, retaining its learned initial state B_0 and the nonlinear readout. We then remove the remaining static second layer and the intermediate nonlinearity, yielding DeltaNet+SWA with a learnable initial memory A_0. Finally, fixing $A_0 = 0$ gives the zero-initialized DeltaNet+SWA baseline, while preserving its online delta-rule updates. The full model achieves the lowest training loss and highest retrieval accuracy. Disabling second-layer updates substantially reduces retrieval accuracy, and retaining a static second layer performs worse than the single-layer model. Removing the learnable initialization further degrades single-layer retrieval. These results highlight the benefits of online adaptation in the second layer and a learned initial memory.

5.6 Mechanism analysis

What do the two layers learn? To examine what the two layers learn, we visualize the initial memory matrices A_0 and B_0 after outer training (Figure 4). Since our main configuration includes a fixed $0.5I$ offset in B_0, we also inspect a variant in which both A_0 and B_0 are initialized to zero. The learned B_0 in this variant also exhibits a pronounced diagonal structure, suggesting that outer training learns a prior for refinement. This is consistent with the intended division of labor: A produces a value-aligned intermediate representation, and B further adjusts it to recover the value.

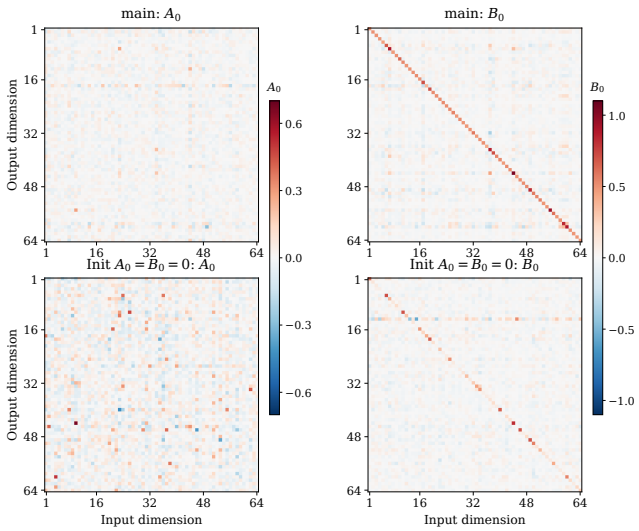

Figure 4: Learned A_0 and B_0 at layer 13, head 1. Top: main configuration; bottom: the variant with both matrices initialized to zero.

Table 6: Initial and final full-sequence reconstruction errors on the same 32K validation sequence. Each checkpoint uses its self-generated K/V features. Reduction denotes the relative decrease in MSE.

Model	Train tokens	Initial MSE	Final MSE	Reduction
DeltaNet	92.5B	0.00408555	0.00118968	70.88%
DeltaNet(learned init)	92.5B	0.00466836	0.00109973	76.44%
DeltaTTT	92.5B	0.00468315	0.00084878	81.88%

How much do online writes reduce reconstruction loss? Table 6 compares the initial and final full-sequence reconstruction errors of DeltaTTT and two DeltaNet baselines on the same 32K validation sequence. Each checkpoint is evaluated using its self-generated K/V pairs. DeltaTTT reduces MSE by 81.88%, compared with 70.88% for zero-initialized DeltaNet and 76.44% for its learned-initialization variant. These within-model reductions show that our layerwise online updates substantially improve memory reconstruction.

6 Related Work

TTT and nonlinear fast weights. TTT was introduced for adapting models to distribution shifts through self-supervised updates at test time (Sun et al., 2020). For sequence modeling, TTT layers represent recurrent memory as an inner learner updated through local self-supervised objectives (Sun et al., 2024). LaCT scales nonlinear fast weights using large update chunks (Zhang et al., 2025), while Titans incorporates momentum and forgetting into online neural memory (Behrouz et al., 2025c). TNT addresses the chunk-size tradeoff through hierarchical memories and subsequent fine-tuning at smaller chunk sizes (Li et al., 2026b). E^2-TTT preserves token-dependent learning rates, momentum, and decay in a parallel chunk update under shared chunk-start gradient evaluation (Zhong et al., 2026). Modular TTT systematically studies inner-learner design and finds that deeper fast-weight networks can underperform shallow alternatives in its evaluated one-step setting (Tang et al., 2026). Complementing this observation, Liu et al. (2026) show that better inner KV fitting need not improve downstream performance and analyze connections between KV-binding TTT and learned linear attention. Other work changes the adaptation objective: TTT-E2E uses next-token prediction and meta-learns an initialization for adaptation (Tandon et al., 2025), while TTCD supervises fast weights using a long-window teacher and a short-window student (Wang et al., 2026b). TTCD's off-policy implementation constructs updates from base-weight activations and accumulates them causally, providing a related instance of fixed-base updating. In vision, ViT^3 studies

inner architectures and training rules (Han et al., 2026), while T^5 converts pretrained vision Transformers into TTT models through architectural and representational alignment (Li et al., 2026a). Our work focuses on layerwise optimization of nonlinear recurrent memory within a single pass, with reconstruction error serving as a diagnostic alongside downstream evaluation.

Delta-rule memories and online optimization. Linear attention admits a recurrent formulation that accumulates key–value associations (Katharopoulos et al., 2020). Schlag et al. (2021) connect linear attention to fast weight programmers and introduce delta-rule writes that correct existing associations. Yang et al. (2024) develop efficient chunkwise evaluation of this recurrence, and Gated DeltaNet combines delta-rule corrections with adaptive forgetting (Yang et al., 2025). Beyond linear memories, Irie et al. (2021) explore recurrent and multilayer fast networks, including Delta MLPs whose matrices receive delta-rule updates from a slow controller. Test-time regression and MIRAS provide broader frameworks for understanding associative memories through their function classes, objectives, and learning rules (Wang et al., 2026a; Behrouz et al., 2025b). ATLAS optimizes memory over a recent context window (Behrouz et al., 2025a), while MesaNet uses a conjugate-gradient solver for in-context regression (von Oswald et al., 2026). DeltaTTT assigns explicit value-reconstruction targets to each linear layer and constructs the second-layer write address from the first layer's prediction. Its local objectives replace joint inner-loop optimization, while the nonlinear intermediate representation couples the two memory stages and each stage reuses chunkwise delta-rule evaluation.

Layerwise and local learning. Difficulties in training deep networks motivated early methods that learned representations one layer at a time. Deep belief networks and greedy layerwise pretraining used successive shallow learning problems to initialize deep models before global fine-tuning (Hinton et al., 2006; Bengio et al., 2006). Stacked denoising autoencoders similarly learned deep representations through local reconstruction objectives (Vincent et al., 2010). Later work explored layerwise targets (Lee et al., 2015), synthetic gradients that relax update dependencies (Jaderberg et al., 2017), and greedy supervised learning at ImageNet scale (Belilovsky et al., 2019). Decoupled Greedy Learning further uses local objectives to relax update locking between network modules (Belilovsky et al., 2020). Our motivation shares the idea of making a deep composition easier to optimize through simpler local learning problems. In DeltaTTT, these problems govern the online updates of context-dependent memory states, while the outer language model remains trained end to end.

7 Conclusion

Our findings suggest that greater memory expressivity alone does not guarantee effective learning within a single pass. We introduced DeltaTTT, which decomposes nonlinear memory optimization into local linear reconstruction objectives with layerwise delta-rule updates, preserving nonlinear composition and enabling chunkwise parallel evaluation. Experiments on two backbones demonstrate improvements in language modeling and associative retrieval.

8 Limitations

Our current DeltaTTT remains slower and uses more memory than DeltaNet. Since the kernels for our two-layer memory are not yet fully optimized, we believe there is room to reduce this overhead through further implementation improvements. In addition, our experiments only consider two-layer MLP memories with a hidden expansion ratio of $1\times$. Extending the approach to wider intermediate representations and deeper nonlinear memories remains an important direction for assessing whether layerwise optimization can effectively exploit greater memory capacity. Finally, our evaluation is limited to relatively small models and a bounded training-token budget. Further experiments with larger models and more training data are needed to establish the scaling potential of DeltaTTT.

AI USE STATEMENT

Generative AI assisted with organizing the manuscript and drafting text and equations from existing experiment records in this revision. No new training or evaluation was performed for this manuscript edit.

REPRODUCIBILITY STATEMENT

Section 4 specifies the memory updates and readout, and Section 5 describes the experimental setup and evaluation. Appendix A provides the chunkwise algorithms for DeltaNet and DeltaTTT-pre. Appendix B details the fixed-KV reconstruction experiments, including data selection, memory architectures, learning-rate tuning, and evaluation. Appendix C documents the shared training configuration, backbone-specific implementations, and main-table DeltaTTT settings. Appendix D reports ablations of the learned initial memory states.

REFERENCES

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. URL https://arxiv.org/abs/2505.23735.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's All Connected: A Journey Through Test-Time Memorization, Attentional Bias, Retention, and Online Optimization. *arXiv preprint arXiv:2504.13173*, 2025b. URL https://arxiv.org/abs/2504.13173.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2025c. URL https://arxiv.org/abs/2501.00663.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to ImageNet. In *Proceedings of the International Conference on Machine Learning*, pp. 583–593, 2019. URL https://proceedings.mlr.press/v97/belilovsky19a.html.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled Greedy Learning of CNNs. In *Proceedings of the International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/1901.08164.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman (eds.), *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper_files/paper/2006/file/5da713a690c067105aeb2fae32403405-Paper.pdf.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2512.01643.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006. URL https://www.cs.toronto.edu/~hinton/absps/fastnc.pdf.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024. URL https://arxiv.org/abs/2404.06654.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2106.06295.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of ICML*, 2017. URL https://proceedings.mlr.press/v70/jaderberg17a.html.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/2006.16236.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases*, 2015. URL https://arxiv.org/abs/1412.7525.

Yining Li, Dongchen Han, Zeyu Liu, Hanyi Wang, Yulin Wang, and Gao Huang. Linearizing vision transformer with test-time training. In *Proceedings of the International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2605.02772.

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 *International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2511.07343.

Junchen Liu, Sven Elflein, Or Litany, Zan Gojcic, and Ruilong Li. Test-Time Training with KV Binding Is Secretly Linear Attention. In *Proceedings of the International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.21204.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the International Conference on Machine Learning*, pp. 9355–9366, 2021. URL https://proceedings.mlr.press/v139/schlag21a.html.

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 *Proceedings of the International Conference on Machine Learning*, pp. 9229–9248, 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, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024. URL https://arxiv.org/abs/2407.04620.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025. URL https://arxiv.org/abs/2512.23675.

Bohao Tang, Zhen Qin, Yuqi Pan, Zheng Li, Pengfei Liu, and Ya Zhang. Modular TTT: Rethinking Test-Time Training as Composable Modules. *arXiv preprint arXiv:2608.07110*, 2026. URL https://arxiv.org/abs/2608.07110.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010. URL https://www.jmlr.org/papers/v11/vincent10a.html.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Sarthak Mittal, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie, Charlotte Frenkel, Razvan Pascanu, Blaise Agüera y Arcas, and João Sacramento. MesaNet: Sequence Modeling by Locally Optimal Test-Time Training. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.05233.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: A unifying framework for designing sequence models with associative memory. *Journal of Machine Learning Research*, 27 (169):1–41, 2026a. URL https://jmlr.org/papers/v27/25-0903.html.

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. *arXiv preprint arXiv:2608.01672*, 2026b. URL https://arxiv.org/abs/2608.01672.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024. URL https://arxiv.org/abs/2406.06484.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

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. URL https://arxiv.org/abs/2505.23884.

Zeyun Zhong, Joya Chen, Manuel Martin, Frederik Diederichs, Juergen Gall, and Juergen Beyerer. Rethinking expressivity and efficiency in test-time training. *arXiv preprint arXiv:2608.21308*, 2026. URL https://arxiv.org/abs/2608.21308.

A CHUNKWISE SCAN OF DELTATTT

Algorithms 1 and 2 describe one memory head. The sequence is divided into N chunks of up to C tokens, with Q_c, K_c, V_c stacking token vectors as rows. W denotes DeltaNet's memory, while A and B denote DeltaTTT's first- and second-layer memories, respectively. $\boldsymbol{\eta}_c$, $\boldsymbol{\alpha}_c$, and $\boldsymbol{\beta}_c$ contain the tokenwise write rates for W, A, and B, respectively. For an incoming memory M, the chunkwise writes are

$$\mathrm{Delta}(K, V, \boldsymbol{r}; M) = \big[I + D\,\mathrm{tril}(KK^\top, -1)\big]^{-1} D(V - KM^\top), \qquad D = \mathrm{diag}(\boldsymbol{r}). \tag{13}$$

Here I is the identity of the chunk length, and the inverse denotes a triangular solve. $\mathrm{tril}(X)$ includes the diagonal, whereas $\mathrm{tril}(X, -1)$ excludes it. $\phi = \mathrm{SiLU}$ is applied elementwise. In DeltaTTT-pre, H_c and R_c are the first-layer pre-update key and post-update query representations, respectively. U_c stores write-rate-scaled residuals, and $[O_1; \ldots; O_N]$ concatenates the output chunks.

Algorithm 1 Chunkwise DeltaNet: one scan

Require: Chunks $\{Q_c, K_c, V_c, \boldsymbol{\eta}_c\}_{c=1}^N$; initial memory W_0
$W \leftarrow W_0$
for $c = 1, \ldots, N$ **do**
  $U_c \leftarrow \mathrm{Delta}(K_c, V_c, \boldsymbol{\eta}_c; W)$
  $O_c \leftarrow Q_c W^\top + \mathrm{tril}(Q_c K_c^\top) U_c$
  $W \leftarrow W + U_c^\top K_c$
end for
return $[O_1; \ldots; O_N], W$

Algorithm 2 Chunkwise DeltaTTT-pre: two successive scans

Require: Chunks $\{Q_c, K_c, V_c, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c\}_{c=1}^N$; initial memories A_0, B_0
$A \leftarrow A_0$; $B \leftarrow B_0$
for $c = 1, \ldots, N$ **do** ▷ First scan: one update, two reads
  $U_c \leftarrow \mathrm{Delta}(K_c, V_c, \boldsymbol{\alpha}_c; A)$
  $H_c \leftarrow \phi\big(K_c A^\top + \mathrm{tril}(K_c K_c^\top, -1) U_c\big)$
  $R_c \leftarrow \phi\big(Q_c A^\top + \mathrm{tril}(Q_c K_c^\top) U_c\big)$
  $A \leftarrow A + U_c^\top K_c$
end for
for $c = 1, \ldots, N$ **do** ▷ Second scan: one update, one read
  $U_c \leftarrow \mathrm{Delta}(H_c, V_c, \boldsymbol{\beta}_c; B)$
  $O_c \leftarrow R_c B^\top + \mathrm{tril}(R_c H_c^\top) U_c$
  $B \leftarrow B + U_c^\top H_c$
end for
return $[O_1; \ldots; O_N], A, B$

B FIXED-KV RECONSTRUCTION EXPERIMENTS

Figure 2 compares linear and nonlinear memories fitted to fixed K/V caches from a frozen 0.5B full-attention model trained on approximately 10B tokens. We use eight 32K sequences and all 24 layers, concatenating heads to a feature dimension of 1,024. Four sequences are used to select learning rates, and the remaining four are used to report results.

We compare a zero-initialized linear memory with a two-layer SiLU MLP of hidden width 1x/2x/4x and Xavier-uniform initialization. Both memories are optimized using ordered SGD replay, with chunk sizes ranging from 1 to 512 in powers of two. Each chunk $\mathcal{C}$ of size C produces one update under the key-normalized squared loss:

$$\mathcal{L}_{\mathcal{C}}(\theta) = \frac{1}{2C} \sum_{i \in \mathcal{C}} \frac{\|f_\theta(k_i) - v_i\|_2^2}{\|k_i\|_2^2}. \tag{14}$$

The MLP updates both matrices simultaneously, and initializations are shared across chunk sizes.

Learning rates are tuned separately for each model and chunk size using reconstruction MSE after 16 epochs; the MLP uses a separate rate for each layer. The selected rates remain fixed for evaluation. We report MSE reconstruction loss over each complete fitted sequence, averaged across the four reporting sequences and all layers. The tokenwise Delta reference corresponds to the linear memory with $C = 1$. Equal numbers of passes do not imply equal computational cost: smaller chunks require more updates.

C Main-Table Experimental Details

Shared training setup. Table 7 summarizes the shared training configuration. All recurrent models use SWA with a 512-token window.

Table 7: Shared training configuration

Setting	Value
Training data	Long-Data-Collections, 92.5B tokens
Model dimensions	24 layers; hidden width 1,024; FFN width 4,096
Vocabulary and embeddings	131,072 tokens; tied input/output embeddings
Sequence length	32,768 tokens
Optimizer	AdamW, $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$
Weight decay / gradient clipping	0.1 / 1.0
Learning rate	Peak 3×10^{-4}; cosine decay to 3×10^{-5}
Warmup	Approximately 1.05B tokens

Backbone implementations. We implement DeltaTTT within both DeltaNet and LaCT backbones, replacing their recurrent memory modules while retaining backbone-specific feature processing and output integration. The main differences concern positional encoding and memory output modulation: the DeltaNet backbone uses RoPE only in the SWA branch and directly adds the RMS-normalized memory output to the SWA output, whereas the LaCT backbone also applies RoPE to memory Q/K and uses an input-dependent scale to modulate the normalized memory output. LaCT additionally retains learned affine transforms on the memory inputs.

DeltaTTT configuration. We align the main architectural settings with our DeltaNet baseline: both implementations use 16 memory heads of width 64, causal depthwise Conv4 with SiLU, L2-normalized memory Q/K. The memory and SWA branches share Q/K/V projections, and their outputs are summed before the output projection. Both main-table DeltaTTT models use Pre updates for memory layers. The learned initial states use $A_0 \sim \mathcal{N}(0, 0.02^2)$ and $B_0 = LR + 0.5I$, where L, R are learned rank-32 factors.

D Initial-State Ablation

Table 8 shows that different learnable initializations yield similar training loss and single-needle retrieval accuracy.

Table 8: Initial-state ablations for DeltaTTT on the DeltaNet backbone. Single denotes average single-needle accuracy (%).

Model	Initial states	Train loss ↓	Single ↑
DeltaTTT	$A_0 = B_0 = 0$ (learnable)	1.95	47.70
DeltaTTT	$(A_0)_{ij}, (B_0)_{ij} \sim \mathcal{N}(0, 0.02^2)$	1.95	48.00
DeltaTTT	$(A_0)_{ij} \sim \mathcal{N}(0, 0.02^2),\ B_0 = LR + 0.5I$	1.95	48.78