Learning to (Learn at Test Time)

RNNs with Expressive Hidden States

Paper authors: Yu Sun et al.

arXiv:2407.04620v4 | Reading: 21 September 2026

Learn a small model on the current sequence;
use its weights as recurrent memory.

Research-group presentation, approximately 25 minutes plus backup.
Source: v4, revised 31 August 2025. All paper experiments were completed in v1.

Fixed-size memory needs a better update rule

Self-attention

Keep a growing list of keys and values.

Query the history at every step.

- Rich access to past context.
- Total attention work: $O(T^2)$.

Recurrent layers

Compress history into a fixed-size state.

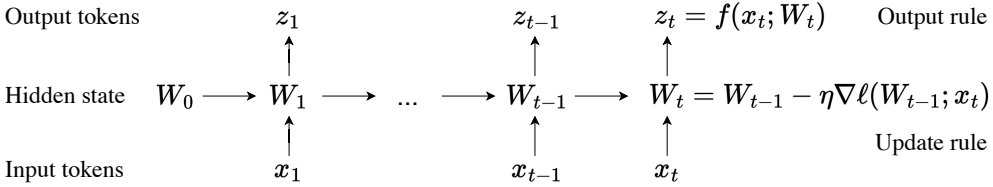
Update that state at every step.

- Total sequence work: $O(T)$.
- Compression can lose useful information.

TTT asks: can learning itself be the compression rule?

Paper, Secs. 1–2, PDF pp. 2–4. Complexity in context length, at fixed model width.

The hidden state becomes a trainable model



$$W_t = W_{t-1} - \eta \nabla_W \ell(W_{t-1}; x_t), \qquad z_t = f(x_t; W_t)$$

- Each sequence has its own evolving fast weights W_t.
- Training this inner model is part of the layer's forward pass.

Paper, Fig. 1, p. 1; Sec. 2.1, Eqs. (1)–(3), p. 4. Naive online formulation.

The outer loop learns what to write and what to query

$$k_t = \theta_K x_t, \qquad v_t = \theta_V x_t, \qquad q_t = \theta_Q x_t$$

Inner learning problem

$$\ell(W; x_t) = \|f(k_t; W) - v_t\|^2, \qquad z_t = f(q_t; W_t)$$

- k_t: training view; v_t: reconstruction target; q_t: test view.
- For one head: $x_t \in \mathbb{R}^D$, $k_t, v_t, q_t \in \mathbb{R}^d$.
- The projections are learned for next-token prediction.
- Current-token reconstruction requires no future-token label.

Paper, Sec. 2.3, Eqs. (4)–(5), PDF pp. 5–6. Simplified per-head dimensions.

Inner adaptation and outer training have different roles

	Inner loop	Outer loop
Data unit	Token representation	Sequence
Training data	Current prefix	Dataset of sequences
Objective	Reconstruction	Next-token prediction
Updated variables	Fast weights W	Shared parameters θ
At inference	Active	Frozen

- Outer training differentiates through inner gradient updates.
- Learn $\theta_K, \theta_V, \theta_Q$, initialization, learning rate, and the rest of the network.
- Reset each sequence to the shared initialization W_0.

Paper, Sec. 2.2, p. 5; Sec. 2.7, pp. 11–12; Table 2, p. 18. Table adapted.

The practical learners need stable inner optimization

$$f(x) = x + \mathrm{LN}(f_{\mathrm{res}}(x))$$

TTT-Linear

$f_{\mathrm{res}}(x) = Wx$

Square linear core; LN and residual added.

TTT-MLP

Two-layer MLP with GELU.

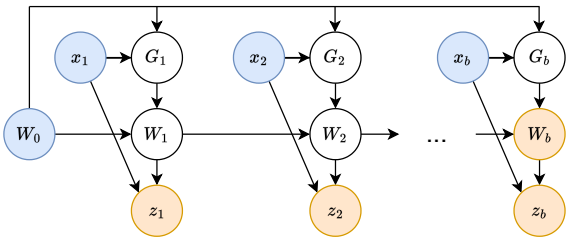
Hidden width is $4\times$ input width.

$$W_0 = \theta_{\mathrm{init}}, \qquad \eta(x) = \eta_{\mathrm{base}}\sigma(\theta_{\mathrm{lr}} \cdot x)$$

- Base learning rate: 1 for Linear, 0.1 for MLP.
- Default backbone adds temporal convolutions and gating.

Paper, Sec. 2.7, PDF pp. 11–12; Appendix C, pp. 29–30.

Mini-batches parallelize gradients without future leakage



For a block starting after token a:

$$G_t = \nabla_W \ell(W_a; x_t)$$

$$W_t = W_a - \eta \sum_{s=a+1}^{t} G_s$$

Gradients share W_a; outputs use causal prefix sums.

- $b = 1$: online GD; $b = T$: full-sequence batch GD.
- Main experiments choose $b = 16$; blocks remain sequential.

Paper, Sec. 2.4 and Fig. 6, p. 7. Constant η for exposition.
For 1-based tokens, use $a = b\lfloor (t-1)/b \rfloor$ to make boundary indexing explicit.

Dual form avoids materializing every intermediate weight

First block, bare linear model, identity views; $X = [x_1, \ldots, x_b]$.

$$E = W_0 X - X, \qquad W_b = W_0 - 2\eta E X^\top$$

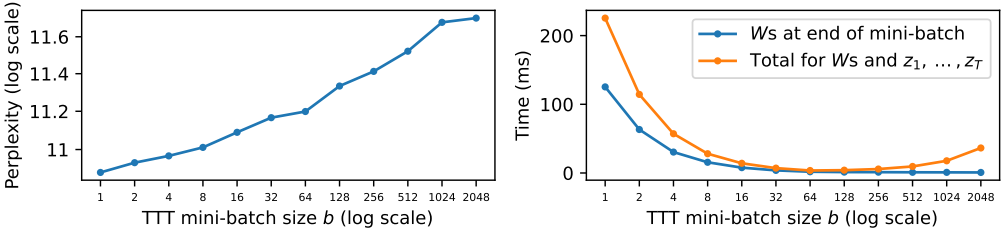
Compute all causal outputs with matrix products

$$Z = W_0 X - 2\eta E \,\mathrm{triu}(X^\top X)$$

- Columns are tokens: upper triangular means source $s \leq t$.
- Same outputs as the primal form of the same mini-batch rule.
- Per block: $O(bd^2 + b^2 d)$; more arithmetic, less costly I/O.
- Reported $> 5\times$ training speedup over primal in JAX/TPU.

Paper, Sec. 2.5, PDF pp. 8–9. Speedup is not relative to Transformer or Mamba.

Batch size trades learning quality against hardware efficiency



Smaller b improves perplexity here. Larger b first reduces latency, then increases output-computation cost.

Paper, Fig. 7, p. 8. Right: $T = 2048$, a single TTT-Linear 1.3B layer in pure PyTorch.
This is not the final fused-kernel whole-network benchmark. Main setting: $b = 16$.

Attention appears as special cases, with important conditions

Linear attention: a parametric learner

Bare linear f, batch GD, $W_0 = 0$, $\eta = 1/2$:

$$W_t = \sum_{s \le t} v_s k_s^\top, \qquad z_t = W_t q_t.$$

Softmax attention: a nonparametric learner

Nadaraya–Watson regression with weights $\exp(k_s^\top q_t)$:

$$z_t = \frac{\sum_{s \le t} \exp(k_s^\top q_t) v_s}{\sum_{s \le t} \exp(k_s^\top q_t)}.$$

The second interpretation still stores a growing history; it gives no linear-time replacement for exact softmax attention.

Paper, Sec. 2.6, Theorems 1–2, pp. 10–11; Appendix B, p. 28. Key/query order made explicit.

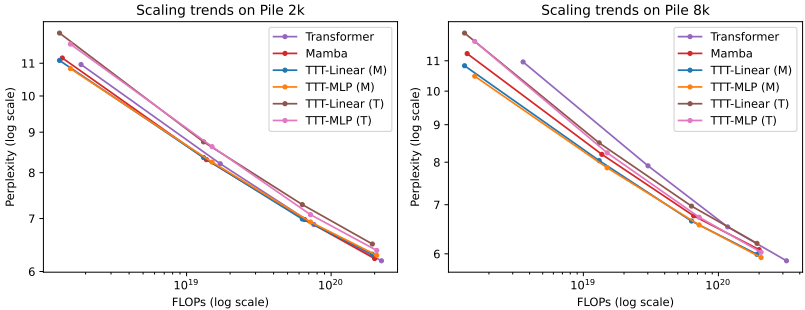
The evaluation separates size, compute, and elapsed time

Setting	Reported protocol
Pile	2k and 8k contexts
Books3	1k, 2k, 4k, 8k, 16k, 32k
TTT / Transformer	125M, 350M, 760M, 1.3B parameters
Mamba	130M, 370M, 790M, 1.4B parameters
Outer training batch	0.5M tokens, independent of context length
Tokenizer	Llama tokenizer for all methods

- Perplexity is lower-is-better; it is not accuracy.
- Similar parameter counts do not imply equal FLOPs or latency.
- Scaling plots connect observations, not fitted scaling laws.

Paper, Sec. 3, pp. 12–14; Appendix C, pp. 29–30. Main plots report no multi-seed error bars.

On Pile, the tradeoff improves as context gets longer



- 2k: Linear (M), Mamba, and Transformer are broadly comparable.
- 8k: TTT improves the compute–perplexity tradeoff vs. Mamba.

Paper, Fig. 10 and Sec. 3.1, p. 13. Axes: training FLOPs and perplexity, both log scale. (M)/(T) denote the backbone. A same-size MLP gain can be offset by extra FLOPs.

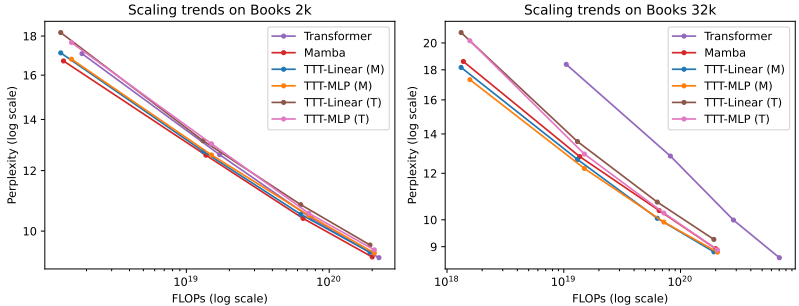
The largest ablation step changes the update regime

Cumulative configuration	PPL ↓	Change
Improved linear attention / equivalent TTT	15.23	—
+ Learnable W_0	15.27	+0.04
+ LN and residual in f	14.05	−1.22
+ Mini-batch TTT	12.35	−1.70
+ Learnable η	11.99	−0.36
+ Mamba backbone	11.09	−0.90

- Mini-batch change: $b = 2048 \rightarrow 16$.
- Learnable initialization supports stability of later variants.
- Sequential additions do not isolate independent causal effects.

Selected rows adapted from Table 1, PDF p. 11. All models: 125M, Pile recipe, 2k.
Original linear attention: 15.91; improved implementation: 15.23.

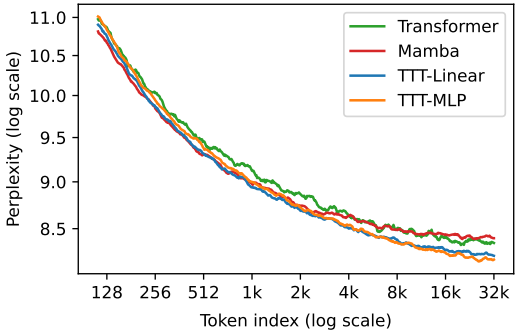
Books extends the evidence to 32k context



- At 32k, TTT variants improve the tradeoff relative to Mamba.
- The stronger long-context Transformer baseline needs finetuning.

Paper, Fig. 11, p. 14. Full results including TF finetune: Fig. 15, p. 32.
TF finetune starts at Books 2k and uses 20% additional tokens at the target length.

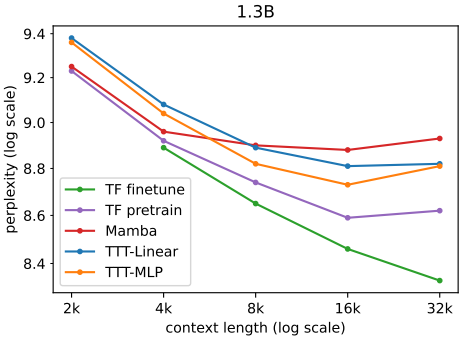
Within a long sequence, TTT keeps benefiting from history



- X-axis: position within a 32k sequence.
- Mamba plateaus after about 16k.
- TTT and Transformer continue to improve.
- Training FLOPs matched to Mamba 1.4B.

Paper, Fig. 2 right, p. 2; Appendix C, p. 30. Transformer is TF finetune, with 19 blocks.
Its inference FLOPs are 2.8× the baseline budget; matched training is not matched inference.

Longer training context is a different question

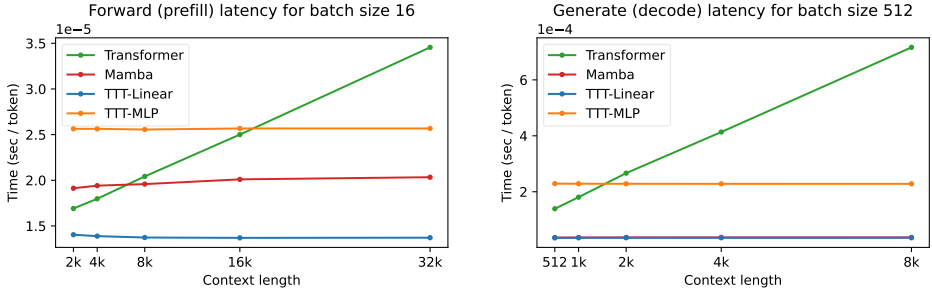


- Each point uses a model trained at a different context length.
- From-scratch performance can worsen beyond an optimum.
- Finetuning improves the Transformer comparison.

Better use of a prefix does not imply that longer-context training is always better at a fixed budget.

Paper, Fig. 16, p. 33: cropped 1.3B panel, original axes retained.
TF finetune has extra training tokens; this is not the matched-budget experiment of Fig. 2.

TTT-MLP's compute efficiency does not guarantee low latency



- TTT-Linear: fast prefill; decode close to Mamba in this setup.
- TTT-MLP: substantially slower despite a fixed-size state.

Paper, Fig. 12, p. 15. A100 80GB PCIe; 1.3B models (Mamba 1.4B).
Prefill batch 16; decode batch 512. Units: seconds/token. Transformer uses vLLM.

The implementation uses different paths for different phases

Phase	Computation strategy
Training / prefill	Dual form for parallel token processing
Autoregressive decode	Primal form for sequential updates
Training memory	Checkpoint hidden states through time

Separate TPU training result

At 2k on a v5e-256 TPU pod:
Transformer: 0.30 s/iteration; TTT-Linear: 0.27 s/iteration.
This is a 10% reduction in iteration time.

- It is not a GPU training comparison against Mamba.
- Only inference GPU kernels were written for this study.

Paper, Sec. 3.3, pp. 15–16; Appendix C, p. 30.

v4 places the contribution in the fast-weights lineage

- Linear matrix states and fast weights predate this paper.
- DeltaNet corresponds to an online linear learner without the added LN and residual.
- Prior fast-weight programmers also explored MLP fast weights.
- The paper contributes an explicit learning framework, learned tasks, practical parallelization, and language-model evidence.

What was not newly benchmarked in v4

Discussion of Mamba 2 and Gated DeltaNet is not an experimental comparison against them.

Paper, Sec. 1, p. 3; Secs. 4.1–4.2, pp. 16–17. All experiments were completed in v1.

The evidence supports a promising design, with clear limits

- Tested up to approximately 1.3B parameters and 32k context.
- Perplexity evidence does not establish exact retrieval or broad reasoning superiority.
- Fixed-size memory still compresses and can interfere with itself.
- Nonlinear learners retain substantial memory-I/O and latency costs.
- Main result plots lack multi-seed uncertainty estimates.

Open question for discussion

At the same end-to-end latency budget, when does a richer hidden model provide enough additional context utility?

Paper, Secs. 3–5, pp. 12–18; discussion question is this reading's analysis.

Take home: learn the memory update, then measure its cost

1. **Representation:** a hidden state can be a small model's weights.
2. **Learning:** the outer objective can shape the inner self-supervised task.
3. **Execution:** mini-batches and dual forms make the idea practical, but nonlinear state is still expensive.

Keep model size, training FLOPs, and measured latency separate.

Synthesis of the paper's method and reported experiments, arXiv:2407.04620v4.

Reference and reading scope

Learning to (Learn at Test Time): RNNs with Expressive Hidden States

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin.

arXiv:2407.04620v4, revised 31 August 2025.

- Read: author TeX, the matching 33-page PDF, and Appendices A–C.
- Figures reproduced from author source; selected tables adapted.
- This presentation reports the paper, not an independent replication.

Companion Chinese notes include detailed locators, indexing cautions, and evidence boundaries.

Backup: error feedback differs from pure association

For a bare linear learner:

$$\nabla_W \ell(W; k, v) = 2(Wk - v)k^\top.$$

Teaching example, not a paper experiment

Scalar $w_0 = 0$, $\eta = 1/2$; observations $(k, v) = (1, 2), (1, 3)$.

$$\text{Online GD: } 0 \to 2 \to 3.$$

$$\text{All gradients at } w_0: \quad 0 \to 2 \to 5.$$

The first rule corrects the existing prediction; the second accumulates associations. Mini-batch TTT lies between these regimes.

Derived teaching example from the loss and GD definitions in Secs. 2.3–2.4.

Backup: nonlinear dual form works layer by layer

Use $\eta = 1$ and half squared loss, as in Appendix A. Let $\hat{X}^k$ be training activations, $\bar{X}^k$ query activations, and $D^k = \nabla_{Z^k} I$.

$$W_b^k = W_0^k - D^k (\hat{X}^k)^\top$$

$$\bar{Z}^k = W_0^k \bar{X}^k - D^k \,\mathrm{triu}\Big((\hat{X}^k)^\top \bar{X}^k\Big)$$

$$\bar{X}^{k+1} = \sigma_k(\bar{Z}^k)$$

- A standard backward pass supplies D^k at the block start.
- A modified forward pass produces all causal query outputs.
- VJPs support normalization; their internal cost remains.

Paper, Appendix A, PDF pp. 26–27. Loss rescaling absorbs the factor 2.

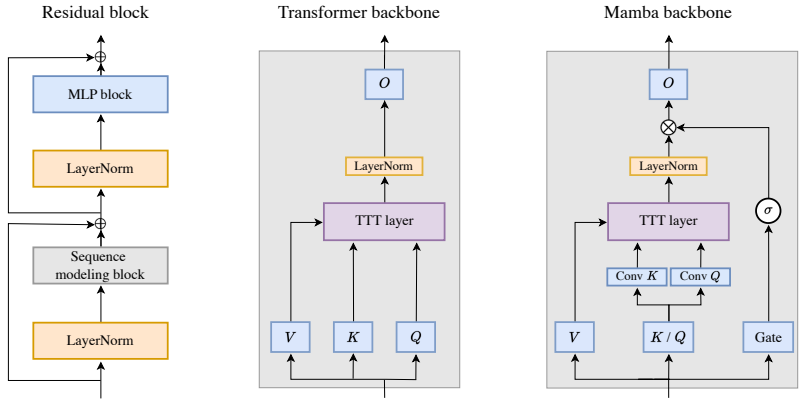
Backup: the reported outer training recipe

Parameters	Blocks	Width	Steps	Tokens
125M	12	768	4,800	2.5B
350M	24	1,024	13,500	7B
760M	24	1,536	29,000	15B
1.3B	24	2,048	50,000	26B

- AdamW: $\beta = (0.9, 0.95)$; weight decay 0.1; clipping 1.0.
- 10% warmup, cosine decay to 10^{-5}; no dropout.
- Peak learning rates: 0.003, 0.0015, 0.00125, 0.001.
- Applies to TTT and Transformer; Mamba has a different structure.

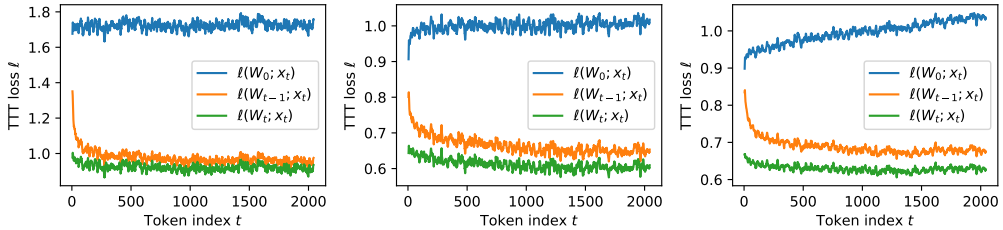
Selected columns from Table 3 and Appendix C, PDF pp. 29–30.

Backup: the backbone is part of the measured gain



Paper, Fig. 13, PDF p. 29. Mamba-style TTT uses temporal convolutions and a gate. The initial K/Q projection is shared in this backbone; separate convolutions follow.

Backup: inner loss supports adaptation, not every outer claim



- A gradient update lowers the current reconstruction loss.
- Past adaptation improves over the initial state on later tokens.
- Some middle layers have rising loss as input norms grow; adaptation still helps relative to initialization.

Paper, Fig. 4, p. 5; all 12 layers in Fig. 14, p. 31. 125M, Pile, $T = 2048$, online-GD diagnostic. Reconstruction improvement does not prove that each next-token loss decreases.

Backup: two notation details matter in implementation

Block boundaries

For token $t = 1, \ldots, T$, the preceding block ends at

$$a = b\lfloor (t-1)/b \rfloor.$$

The printed $t - \mathrm{mod}(t, b)$ is ambiguous at multiples of b under this indexing.

Kernel argument order

Pair a historical key with the current query: $\exp(k_s^\top q_t)$. The argument order in Eqs. (9)–(10) needs care if the first argument is interpreted as the query.

Reading notes on Sec. 2.4, p. 7 and Sec. 2.6, p. 10.
These are notation clarifications, not new experimental findings.