Test-Time Training Done Right

Large Chunk Test-Time Training (LaCT)

Paper by Tianyuan Zhang et al.
MIT and Adobe Research

arXiv:2505.23884v1 (29 May 2025)
Paper discussion prepared 21 September 2026

The contribution is a capacity–efficiency tradeoff

- Write context into fast weights in large chunks: 2K tokens to about 1M, depending on the task.
- Use the resulting GPU efficiency to scale nonlinear memory and richer update rules.
- Keep local window attention for fine-grained structure and causality.
- Evidence: novel view synthesis, language modeling, and autoregressive video diffusion.

Paper, Secs. 1–2, pp. 1–3. Three separate task-specific models.

Fast weights are a temporary neural memory

$$\text{Write: } W \leftarrow W - \eta \nabla_W \mathcal{L}(f_W(k), v) \qquad \text{Read: } o = f_W(q).$$

- Slow weights learn across training samples; fast weights adapt within a context.
- The same memory mechanism runs during training and inference.
- A fixed state compresses history, so storage does not grow with every token.
- Compression limits what can be retrieved exactly.

Paper, Sec. 2.1, Eqs. (1)–(2), p. 2.

Small updates can leave matrix units underused

$$r_{\text{intensity}} = \frac{2h^2 b}{2h^2 + 4hb} = \frac{b}{1 + 2b/h} \leq \min(b, h/2).$$

- For BF16 matmul: FLOPs divided by bytes moved.
- Both chunk size b and matrix width h constrain arithmetic intensity.
- Large chunks offer sequence-dimension parallelism within each update.
- Reported peak utilization is a layer-level result, not universal end-to-end utilization.

Paper, Sec. 2.2, Eq. (3), Fig. 1, p. 3.

Two paths handle two scales of dependency

- Window attention: local token structure and fine-grained causality.
- Large-chunk TTT: compressed history in a nonlinear weight state.
- A conventional feed-forward network supplies channel mixing.
- Language/video share QKV between TTT and window branches; outputs are combined.

Paper, Fig. 2, p. 4; Algorithm 2, p. 20.

The memory is a SwiGLU network

$$f_W(x) = W_2[\mathrm{SiLU}(W_1 x) \circ (W_3 x)]$$

$$\mathcal{L}_i = -f_W(k_i)^\top v_i, \qquad g = \nabla_W \sum_{i=1}^{b} \eta_i \mathcal{L}_i.$$

- $x, k_i, v_i \in \mathbb{R}^h$; $W_1, W_3 \in \mathbb{R}^{rh \times h}$ and $W_2 \in \mathbb{R}^{h \times rh}$.
- h: head width; r: hidden-width multiplier. The objective is negative dot product, not MSE.
- Token-dependent positive η_i controls write importance; normalization stabilizes the state.

Paper, Sec. 3.1, Eqs. (4)–(7), p. 4; Algorithm 1, p. 19.

Muon changes how the write uses matrix directions

$$g = U\Sigma V^\top, \quad \mathrm{Muon}(g) \approx UV^\top, \qquad W \leftarrow \mathrm{L2Norm}(W - \mathrm{Muon}(g)).$$

- Five Newton–Schulz iterations approximate gradient orthogonalization.
- With momentum: $M \leftarrow \bar{\beta} M + g$, then transform M.
- Large chunks amortize matrix update overhead.
- Muon helps key NVS/language results; video experiments do not use it.

Paper, Sec. 3.2, p. 5; Appendix A, pp. 18–20; C.3, p. 27.

Read/write order defines the dependency mask

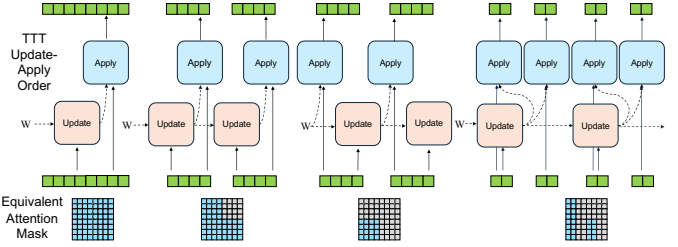


Figure 3: Different 'Update' and 'Apply' orders and their equivalent attention mask. A blue mask in i-th row and j-th column means the i-th token's output depends on the j-th token.

- Update then apply: current chunk can communicate internally.
- Apply then update: read only previous chunks; no future leakage.

Original Fig. 3, p. 5. Masks describe dependencies, not numerical attention equivalence.

Language needs both delayed writes and a causal window

Teaching example: chunk length 4

Tokens 5–8 read the state written by tokens 1–4. A causal local window handles recent tokens. Write tokens 5–8 only after reading the chunk.

- Window length at least chunk length covers the missing recent dependencies.
- Actual chunks/windows: 2048 or 4096 tokens.
- Low-rank learnable initialization reduces slow parameters, not the full online state.

Paper, Sec. 4.2, p. 7; Fig. 10 and Appendix B.2, p. 22; C.2, pp. 25–26.

For image sets, prefill builds a scene-specific memory

- Input: posed RGB images; query: camera rays of target views.
- Within-image attention preserves local spatial structure.
- Each layer writes all input views, then reads for input and target tokens.
- After prefill, render new views with fixed fast weights; targets do not write back.

Paper, Sec. 4.1, pp. 6–7; Fig. 9 and Appendix B.1, pp. 20–21.

Video writes clean context, not denoising candidates

$$S = [X_1^{\text{noise}}, X_1, X_2^{\text{noise}}, X_2, \ldots], \qquad X_i^{\text{noise}} = (1 - t_i)X_i + t_i\epsilon.$$

- One chunk = 3 latent frames = 4680 visual tokens.
- Only clean chunks update persistent memory; noisy chunks read it.
- A local window spans two chunks with a causal inter-chunk mask.
- 1.3B, 5-second training repeats noisy chunks twice; 14B does not.

Paper, Sec. 4.3, p. 7; Fig. 11, p. 23; Appendix C.3, pp. 27–28.

Large chunks also support distributed writes

$$g = \sum_j \nabla_W \sum_{i \in \text{shard } j} \eta_i \mathcal{L}_i.$$

- All devices compute local gradients at the same old state.
- All-reduce the gradients before nonlinear Muon and normalization.
- NVS: split tokens inside a chunk; reported throughput overhead 1–3%.
- Video: shard TTT heads, switching between sequence and head layouts.

Paper, Sec. 3.4, p. 6; Appendices E–F, pp. 29–32.

The three evaluations test different capabilities

Task	Setup	Main metric
View synthesis	Objaverse/GSO; DL3DV; 312M model	PSNR, SSIM, LPIPS
Language	760M/40B and 3B/60B training tokens; 32K context	Per-position loss; retrieval
Video	Wan 2.1, 1.3B and 14B; proprietary training data	Held-out denoising loss

- Linear-cost baselines receive matching local attention.
- Maximum lengths and training protocols must not be mixed across tasks.

Paper, Sec. 5 and Table 1, pp. 7–11; Appendix C, pp. 24–28.

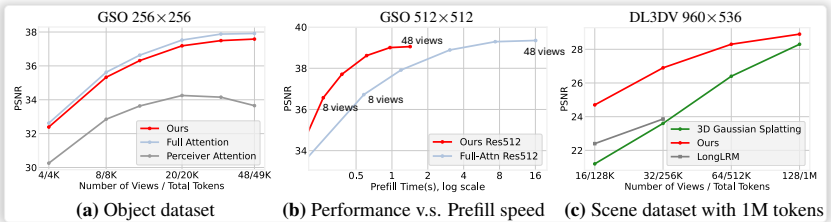
NVS substantially reduces prefill time

Model	Parameters	Prefill (s)↓	Render FPS↑
Full attention	284M	16.1	2.3
Perceiver attention	287M	16.8	34.4
LaCT	312M	1.4	38.7

- A100; 48 input images at 512×512; 196K input tokens.
- Calculated: $11.5\times$ faster prefill and $16.8\times$ rendering throughput versus full attention.
- These are separate phases, not a single end-to-end speedup.

Paper, Table 2, p. 8. Ratios calculated from reported values.

View quality stays close to full attention

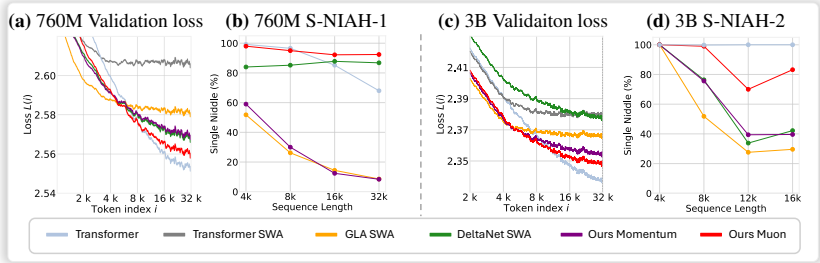


(a) Object dataset **(b)** Performance v.s. Prefill speed **(c)** Scene dataset with 1M tokens

- GSO, 512^2, 48 views: 39.1 dB LaCT vs. 39.3 dB full attention.
- DL3DV: 24.7 to 28.9 dB as input views rise from 16 to 128.
- The million-token result uses posed images, not text retrieval.

Original Fig. 4, p. 8; Tables 5–6, p. 25.

Language curves show useful long-range memory



- Lower later-position loss than GLA/DeltaNet in the reported settings.
- Selected retrieval curves favor LaCT Muon.
- The complete appendix tables reveal important failure cases.

Original Fig. 5, p. 9; full retrieval results in Tables 8–9, p. 27.

Complex retrieval remains far below full attention

3B model, 32K context	Single-needle avg.	Multi-needle avg.
Transformer	91.2	43.0
GLA + SWA	13.3	11.7
DeltaNet + SWA	28.4	14.8
LaCT Momentum	40.6	14.2
LaCT Muon	48.0	21.2

- Accuracy in percent; higher is better.
- Single-needle gain vs. DeltaNet: +19.6 percentage points.
- Gap to Transformer: 43.2 points. Muon is not best in every table cell.

Paper, Tables 8–9, p. 27. Averages use the three reported tasks in each group.

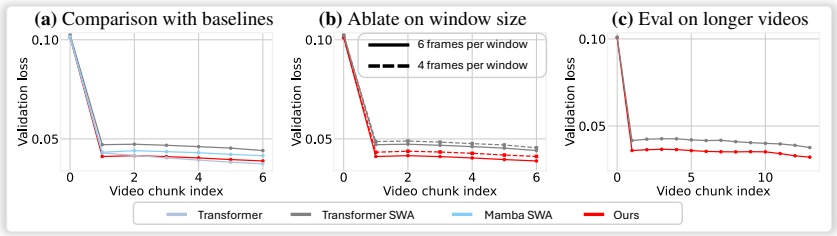
Capacity improves without becoming the fastest model

Model	Tokens/s/GPU	State elements/block
Transformer	4.1K	—
Transformer SWA	6.4K	—
GLA SWA	5.0K	$384d$
DeltaNet SWA	5.1K	$128d$
LaCT Momentum	4.9K	$2304d$
LaCT Muon	4.3K	$2304d$

- 3B, 32K, $d = 3072$; 8 A100-40GB GPUs, FSDP and checkpointing.
- LaCT has $6\times$ GLA / $18\times$ DeltaNet fast-weight state.

Paper, Table 3, p. 9; Appendix C.2 and Table 7, p. 26.

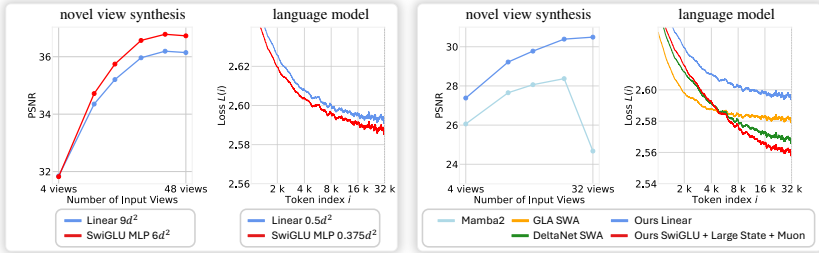
Video loss supports memory utility within trained lengths



- Comparable to full block-causal attention; better than SWA/Mamba+SWA.
- Longer-video evaluation follows longer-video fine-tuning.
- 14B: 56,160 visual tokens; about 107K interleaved training tokens.

Original Fig. 6, p. 10; Appendix C.3, pp. 27–28. No zero-shot length extrapolation claim.

The ablations support the combined design



(a) Linear v.s. NonLinear Fast weight

(b) Large-Chunk v.s. Per-token Recurrence

- Nonlinear memory beats larger linear memory in these comparisons.
- Equal-state linear large chunks help NVS, but underperform GLA in language.
- The winning language variant also adds larger state and Muon.

Original Fig. 8, p. 11; state/optimizer analysis in Fig. 7, p. 10.

Interpret state size using actual matrix dimensions

$$S = 3n_h h^2 r = \frac{3rd^2}{n_h}, \qquad \mathrm{FLOPs}_{\mathrm{write+read}} \approx 6nS.$$

- n_h: head count; $d = n_h h$; n: tokens; r: hidden-width multiplier. Three matrices and nine main matmuls.
- NVS default: $6d^2$ per block; about 84M fast weights in 312M parameters.
- The 40% state/parameter ratio is a larger-state configuration.
- Momentum, attention cache, activations and Muon cost are additional.

Derivation from Eq. (6), p. 4 and Algorithm 1, p. 19; Appendix A, p. 18 has notation errors.

The evidence has clear boundaries

- NVS requires camera poses; unposed reconstruction is untested.
- Language reasoning and much larger language models are untested.
- Fixed-size memory still loses retrieval detail; no unlimited-capacity guarantee.
- Video uses proprietary data and denoising loss rather than a complete perceptual evaluation.
- Q/K rotation invariance is absent in the proposed fast-weight branch.

Paper, Sec. 7, p. 13; reader analysis based on Tables 8–9, p. 27.

Take away the design principle, with its conditions

Main idea

Amortize memory writes over large chunks, spend the resulting efficiency on expressive state, and preserve local structure with attention.

- Strong evidence for an NVS quality–speed tradeoff.
- Useful language/video results, with task-dependent limitations.
- Open question: how does the tradeoff change at matched total compute and peak memory?

Paper, Sec. 8, p. 13. Final question is a discussion prompt, not a paper result.

Source and reading scope

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, Hao Tan.

Test-Time Training Done Right. arXiv:2505.23884v1, 2025.
`https://arxiv.org/abs/2505.23884v1`

Read: full paper and Appendices A–F; same-version author TeX and PDF.

Figures are cropped from the original PDF. Tables select reported values. Interpretations and derived ratios are identified. No experimental reproduction or project-video assessment was performed.

Prepared as a research-group paper discussion, approximately 25–30 minutes.