The Surprising Effectiveness of Shared Memory in Looped Transformers

LPT: learn to write once, then read across recursions

Paper by Giovanni Monea, Keshav Ramji, Yousef El-Kurdi,
Luis A. Lastras, Yoav Artzi, Nathan Godey,
and Ramón Fernandez Astudillo

Cornell University and IBM Research

Independent paper-reading presentation

arXiv:2610.02383v1 | 16 October 2026

Training to share memory improves quality

The central result

Later recursions reuse the first recursion's KV cache and keep only a short local window. Sharing is learned during pretraining.

1B model	PPL ↓	Accuracy ↑	Memory / STD ↓
STD, one pass	11.35	62.5%	1.00
LT, five recursions	11.39	63.4%	5.00
LPT, five recursions	10.32	68.2%	1.06
H-LPT, five recursions	**10.23**	68.1%	**0.24**

Paper, Table 1, PDF p. 4. FineWeb-Edu PPL; mean accuracy over five zero-shot tasks.

Weight sharing does not eliminate cache growth

$$h^{(t)} = \mathcal{M}^{L}(h_{\text{in}}^{(t)}), \qquad h_{\text{in}}^{(t+1)} = \mathcal{N}_{\text{rec}}(h^{(t)})$$

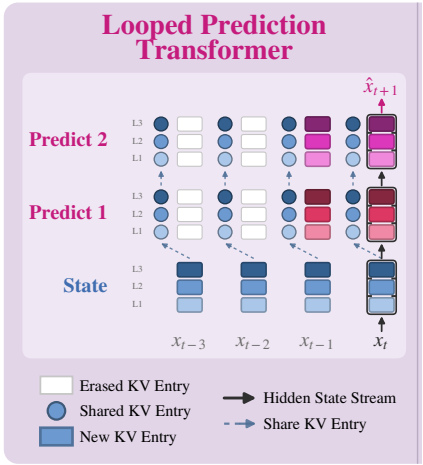
- One stack of L layers is reused R times: effective depth LR.
- Each use receives a different hidden state and writes different keys and values.
- Standard LT stores a full cache at every recursion:

$$\mathcal{A}_{\text{LT}}^{(t)}(i) = C_{\leq i}^{(t)}, \qquad M_{\text{KV}}^{\text{LT}} = LRT.$$

Compute and context memory both grow with recurrent depth.

Paper, Sec. 2, PDF pp. 2–3, Eqs. (1)–(2); Eq. (4), p. 4. Common KV byte factors omitted.

LPT assigns memory and prediction to different passes



- **State recursion:** build a full causal KV cache at each layer.
- **Prediction recursions:** read that same layer's shared cache and a local window.
- Keep the sequential hidden-state recurrence and shared weights.

Sharing is across recursions at a fixed layer.

Paper, Fig. 1 (left), PDF p. 1; Sec. 3, p. 3. Original panel, including its legend.

Shared and local entries compete in one softmax

$$\mathcal{A}^{(t)}_{\mathrm{LPT}}(i) = \begin{cases} \mathcal{C}^{(1)}_{\le i}, & t = 1, \\ \mathcal{C}^{(1)}_{\le i} \cup \mathcal{C}^{(t)}_{[i-\omega,i]}, & t = 2, \dots, R. \end{cases}$$

$$o = \mathrm{softmax}\left(\frac{q[K_s; K_l]^\top}{\sqrt{d_h}} + \mathrm{mask} \right) [V_s; V_l]$$

- s: shared entries; l: local entries; d_h: head dimension.
- ω counts *preceding* positions; the current local entry also participates.
- $\omega = 0$: no local history. $\omega = T$: full local history plus sharing.

Paper, Sec. 3, PDF p. 3, Eq. (3). Second equation is a teaching expansion of the joint softmax.

Five recursions need only 6.25% more cache than STD

$$M_{\mathrm{KV}}^{\mathrm{LT}} = LRT, \qquad M_{\mathrm{KV}}^{\mathrm{LPT}} = L[T + (R-1)\omega]$$

$T = 4096,\ R = 5,\ \omega = 64$	LT	LPT
Cached positions per layer	20,480	4,352
Cache relative to STD	5.00×	1.0625×

Storage and reading are different costs

LPT retains 21.25% of LT's cache (4.706× smaller), but every recursion still reads a full shared prefix.

Local windows add 1.25% attended entries; total decoding FLOPs rise by less than 1% relative to LT.

Paper, Sec. 3, PDF p. 4, Eq. (4); Table 1. Steady-state cache convention; current-entry details omitted.

H-LPT combines cache sharing with a hybrid backbone

LT2 baseline

Gated DeltaNet layers, with roughly one full-attention layer per five layers.

- GDN state is independent of context length.
- Remaining attention caches still grow with R.

H-LPT modification

Apply the LPT sharing rule to each full-attention layer.

- GDN keeps a separate state per recursion.
- Full-attention caches share the first recursion's memory.

Context memory includes both KV caches and GDN recurrent states.

Paper, Secs. 2–4, PDF pp. 3–5; Appendix A, p. 15.

Experiments hold parameters and training tokens fixed

Base configuration	S	M	L
Parameters	151.9M	454.2M	1.01B
Layers / hidden width	12 / 768	24 / 1024	36 / 1280
FineWeb-Edu tokens	20B	20B	50B

- Context: 4,096; recursions: 2, 3, 5; default window: 64.
- PPL: 10M held-out tokens. Zero-shot accuracy: ARC-Easy, HellaSwag, PIQA, SciQ, and LAMBADA (mean).
- LT uses learned weights over per-pass NLLs (**adaptive**); others supervise only the last pass (**final**).
- Same optimizer recipe; four seeds at S, one main run at M/L.

Paper, Sec. 4, PDF pp. 4–5; Appendices A–B, pp. 15–16. More recursions require more compute.

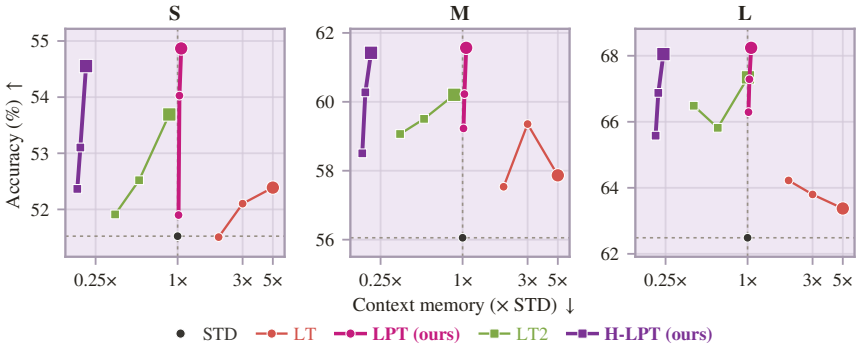
LPT keeps improving from three to five recursions

Size	Model	$R = 2$ PPL	$R = 3$ PPL	$R = 5$ PPL
S	LT	16.17	15.98	16.00
	LPT	15.73	15.28	**14.91**
M	LT	13.28	13.13	13.34
	LPT	12.91	12.55	**12.33**
L	LT	11.20	11.11	11.39
	LPT	10.75	10.45	**10.32**

At five recursions, LPT improves accuracy over LT by
2.5 / 3.7 / 4.8 percentage points at S / M / L.

Paper, Table 1, PDF p. 4. Main objectives: adaptive for LT, final for LPT. PPL has no nats unit.

H-LPT reaches high accuracy with much less context memory

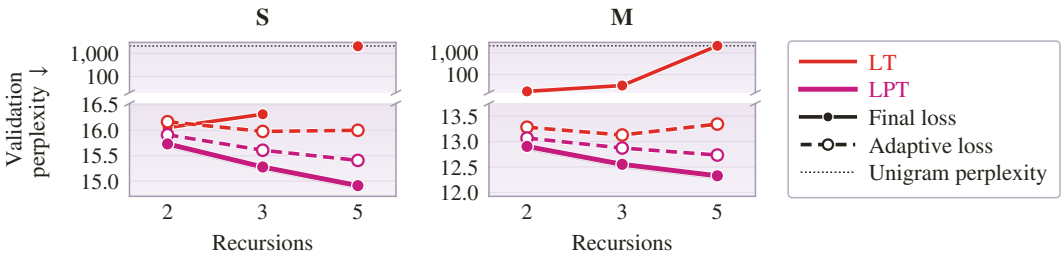


At $R = 5$, relative to same-size STD across S/M/L:

- PPL lower by 1.12–1.82; accuracy higher by 3.1–5.6 points.
- Context memory lower by approximately 76–79%.

Paper, Fig. 2, PDF p. 5; Table 1, p. 4. Curves: $R = 2, 3, 5$; largest marker: $R = 5$.

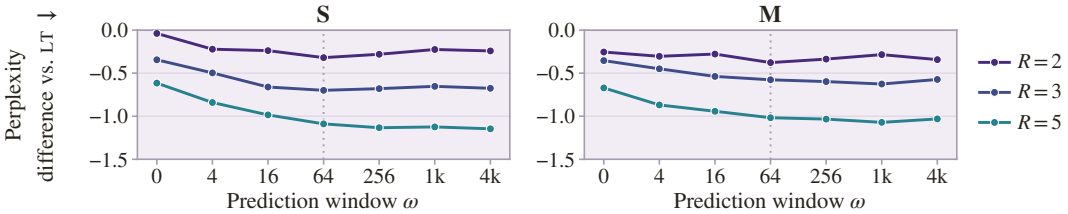
Sharing makes final-only supervision effective



- Under the default Ouro normalization, final-loss LT collapses at $R = 5$.
- LPT stays stable; final loss beats its adaptive loss by 0.16–0.50 PPL.
- Interpretation: intermediate passes can specialize in further computation.

Paper, Fig. 3, PDF p. 7; Sec. 6, p. 6. Filled: final; hollow: adaptive. Upper strip uses a log scale.

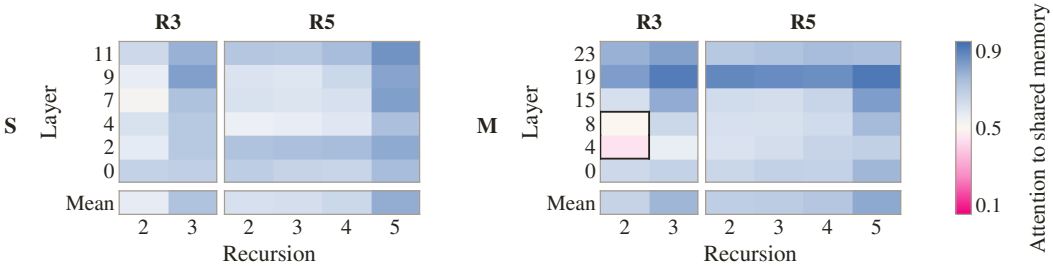
Access to shared memory matters more than window size



- LPT beats LT at every tested window, including $\omega = 0$.
- With full local history, gains remain: 0.24–1.15 PPL.
- $\omega = 64$ balances quality and cache size.

Paper, Fig. 4, PDF p. 7; Sec. 6, p. 6. Negative PPL differences favor LPT. LT uses its better objective.

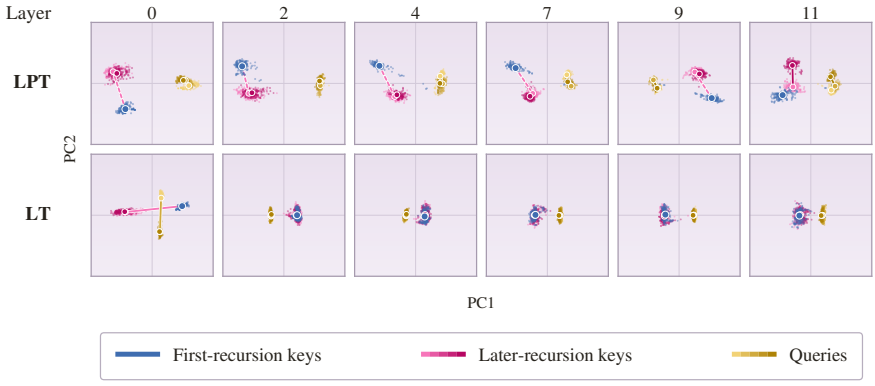
Later recursions place more attention on shared memory



- Shared attention mass: 57–71% in earlier prediction passes.
- Final recursion: 73–80%; mean over heads and query positions.
- Full local caches: overall preference persists, with some exceptions.

Paper, Fig. 5, PDF p. 8; Fig. 11, p. 19. Same 4,096-token sequence; mean rows use all layers.

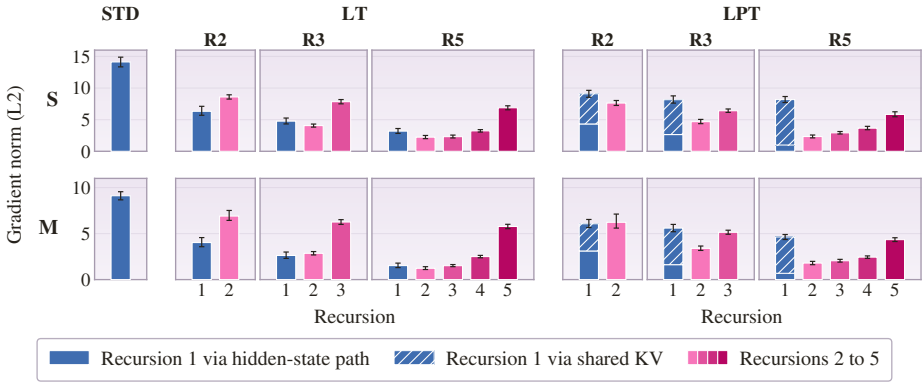
Shared and local keys learn different representations



- LPT separates first-pass keys from later-pass keys more clearly.
- Joint Q/K PCA before RoPE; each panel has its own basis.

Paper, Fig. 6, PDF p. 8; Appendix D, p. 17. S, $R = 3$, final loss for both models.

Shared KV gives the first pass a direct gradient route



- LT's first-pass gradient shrinks strongly with depth.
- LPT restores signal; shared KV dominates its first-pass contribution at $R = 5$.

Paper, Fig. 7, PDF p. 9. First LPT bar: path projections. Whiskers: 95% sequence-bootstrap intervals.

Detaching shared reads harms learning from scratch

Size	Model, final loss	$R = 3$ PPL	$R = 5$ PPL
S	LT	16.32	2,040
	LPT, detached shared KV	16.14	33.21
	LPT	**15.28**	**14.91**
M	LT	31.74	2,090
	LPT, detached shared KV	1,995	1,527
	LPT	**12.55**	**12.33**

- Forward reads and the hidden-state gradient path remain unchanged.
- Evidence supports a learning role for shared-cache gradients.

Paper, Table 6, PDF p. 20; Fig. 8, p. 9. Does not separate path length from gradient scale/direction.

Stronger normalization narrows the gap to LT

Size	Model	$R = 2$ PPL	$R = 3$ PPL	$R = 5$ PPL
S	Best tested LT norm	16.04	15.65	15.24
	LPT, Ouro norm	15.73	15.28	14.91
M	Best tested LT norm	12.96	12.75	12.76
	LPT, Ouro norm	12.91	12.55	12.33

- Alternatives allow final-loss LT to train at depth.
- LPT stays ahead, but the M / $R = 2$ displayed gap is only 0.05 PPL.
- Stability claims depend on normalization and optimizer settings.

Paper, Table 5, PDF p. 17. Best LT selected separately per setting; learning rate/warmup not retuned.

Memory savings improve capacity while recursions add latency

1B model	R	Max batch	Tokens/s	B=1 ms/step
STD	1	108	3,162	6.7
LT	5	21	303	37.9
LPT	5	101	578	42.3
LT2	5	103	1,776	21.7
H-LPT	5	390	2,553	22.6
H-LPT	3	444	4,333	13.6

LPT vs. LT, $R = 5$: 4.81× batch capacity, 1.91× peak throughput, but 1.12× batch-one latency.

Paper, Table 8, PDF p. 25. One H100 80GB; bf16, 4,096 context, shared research implementation.

Nine recursions expose a limit of pure-attention LPT

	LPT PPL ↓		H-LPT PPL ↓	
Size	$R = 5$	$R = 9$	$R = 5$	$R = 9$
S	14.91	14.80	15.01	14.74
M	12.33	12.39	12.31	12.14
L	10.32	10.87	10.23	10.11

- 1B LPT accuracy falls from 68.2% to 66.5%.
- H-LPT continues improving at all three sizes.
- Small cache growth does not guarantee useful extra computation.

Paper, Appendix E, PDF pp. 20–23; Table 7, p. 21.

The contribution is a learned read–write division of labor

1. **Architecture:** keep sequential depth, share first-pass memory, retain a short local window.
2. **Quality:** lower PPL than the main looped baselines; H-LPT gives the strongest memory trade-off.
3. **Mechanism:** distinct key geometry plus a useful direct gradient route to memory construction.

The cache is both a reusable forward representation and a route for learning what to store.

Synthesis of Secs. 3, 5–8, PDF pp. 3–10. Role specialization is an interpretation supported by probes.

The deployment choice depends on the binding resource

- **Memory-limited concurrency:** sharing admits larger batches; three-pass H-LPT is an attractive measured point.
- **Single-request latency:** repeated passes remain expensive.
- **Compute budget:** no FLOPs-matched larger STD was trained; five-pass LPT costs 4.3–4.8$\times$ STD decoding FLOPs.
- **Scope:** one corpus, up to 1B parameters, 4K context; long-context tasks remain untested.

Paper, Sec. 8, PDF p. 10; Appendix F, pp. 24–26. Resource-based interpretation by this presentation.

Paper reference and discussion

Giovanni Monea, Keshav Ramji, Yousef El-Kurdi, Luis A. Lastras, Yoav Artzi, Nathan Godey, and Ramón Fernandez Astudillo.

The Surprising Effectiveness of Shared Memory in Looped Transformers.
arXiv:2610.02383v1, 1 October 2026.

arxiv.org/abs/2610.02383v1

Discussion

If memory construction can be reused across depth, how much recurrent compute should be spent writing versus reading it?

All experimental figures in this deck are original paper figures. Selected tables are transcribed from the paper.

Backup: all baselines answer different questions

Baseline	Comparison role
STD	One pass, same-size parameter reference.
LT / LT2	Direct full-attention / hybrid sharing controls.
PLT	Shared first-pass memory with gated dual attention and parallel recurrence.
Loopie	Repeat a layer before advancing to the next layer.
DeepStd	Untied, deeper and narrower; matches effective depth and approximate parameters.
SPS	Separate memory and prediction residual streams; listed at the two-pass cost point.

PLT differs from LPT in multiple design choices; its gap does not isolate the softmax choice alone.

Paper, Sec. 4, PDF p. 4; Sec. 7, pp. 8–9; Appendix E, pp. 20–21.

Backup: seed evidence is strongest at S

S-size comparison	R	Mean PPL gap $\pm$ 95% CI	One-sided p
LPT $-$ LT	2	-0.40 ± 0.05	6.5×10^{-5}
	3	-0.71 ± 0.05	9.4×10^{-6}
	5	-1.05 ± 0.04	2.4×10^{-6}
H-LPT $-$ LT2	5	-0.13 ± 0.02	1.6×10^{-4}
LPT $-$ PLT	2	-0.07 ± 0.09	4.8×10^{-2}

- Four paired seeds; initialization and data order both change.
- Intervals use Student-t; tests use paired same-seed differences.
- M/L and downstream accuracy have less replication coverage.

Paper, Appendix B, PDF pp. 15–16, Tables 3–4. Negative gaps favor the first model.

Backup: gradient bars are projected contributions

- Give each recursion an equal-valued independent parameter copy for the backward probe.
- Detach shared KV reads to isolate g_{hidden}; then

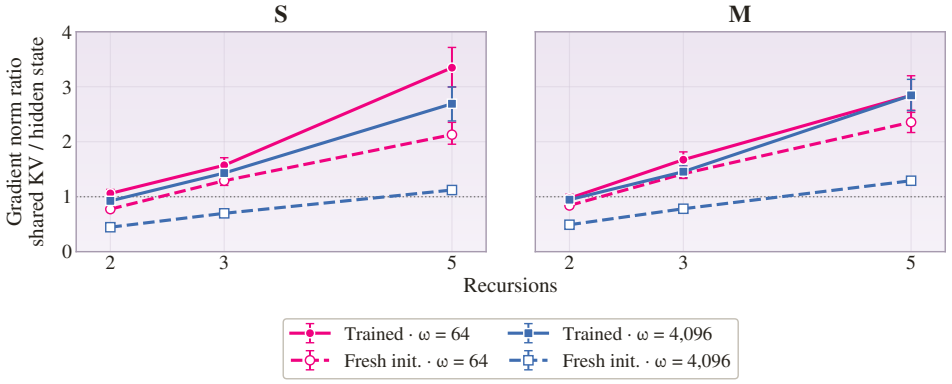
$$g_{\text{KV}} = g^{(1)} - g_{\text{hidden}}.$$

$$p_a = \frac{\langle g_a, g^{(1)} \rangle}{\|g^{(1)}\|_2}, \qquad p_{\text{KV}} + p_{\text{hidden}} = \|g^{(1)}\|_2.$$

- Raw path norms are not additive.
- 32 sequences $\times$ 32 sampled loss positions; final loss in every probe.
- Excludes embeddings, readout, recurrent norm, and halting gate.

Paper, Appendix D, PDF pp. 17–19; Eq. (5), p. 18. Error bars resample sequences, not training seeds.

Backup: a larger KV path already appears at initialization



- More recursions create more shared-cache reads even before training.
- Trained ratios are larger; detachment is the stronger learning intervention.

Paper, Fig. 13, PDF p. 20. Geometric mean norm ratios; fresh random weights, not saved initial checkpoints.

Backup: implementation details explain the cost accounting

- Training: concatenate shared/local KV and use a FlexAttention block mask.
- Decoding: combine two attention outputs using log-sum-exp weights to reproduce a joint softmax.
- GPU timing: preallocated caches, whole-step CUDA graphs, random weights; maximum feasible batch per model.
- Hybrid measured state is 3–17% above the analytic estimate: fp32 GDN state plus short-convolution buffers.

Paper, Sec. 3, PDF p. 4; Appendix A, p. 15; Appendix F, p. 24.

Backup: quality claims should retain their metric and setting

- H-LPT beats LT2 on main-table PPL, but two-pass M/L accuracy is lower by 0.6/0.9 points.
- The Introduction labels a 1.09 PPL reduction as "nats." PPL is dimensionless; NLL uses nats/token.
- Under better normalization, LT can learn with final loss and the quality gap narrows.
- Nine-pass 1B LPT degrades relative to five passes; H-LPT still improves in this experiment.

Paper, Table 1, PDF p. 4; Introduction, p. 2; Table 5, p. 17; Table 7, p. 21.

Backup: a concrete query separates time from recursion

Teaching example: one layer, token position $i = 5$, $R = 3$, $\omega = 2$.

Recursion	First-pass entries	Current-pass entries
$t = 1$	Positions $1, 2, 3, 4, 5$	No extra local set
$t = 2$	Positions $1, 2, 3, 4, 5$	Positions $3, 4, 5$ from pass 2
$t = 3$	Positions $1, 2, 3, 4, 5$	Positions $3, 4, 5$ from pass 3

- The current local entry is included; ω counts the two preceding positions.
- Two representations of the same token remain distinct entries.
- Position 6 is never visible. Prediction passes leave first-pass KV intact.

Teaching expansion of Eq. (3), PDF p. 3. All positions assumed to lie in the same document.

Backup: two attention kernels can implement one softmax

For one query, let $a_{s,j}$ and $a_{l,j}$ be its shared/local scores, including scaling, position information, and masking.

$$\ell_s = \log \sum_j e^{a_{s,j}}, \qquad o_s = \sum_j e^{a_{s,j} - \ell_s} v_{s,j}$$

Define ℓ_l, o_l analogously and $\ell = \log(e^{\ell_s} + e^{\ell_l})$. Then

$$\boxed{o = e^{\ell_s - \ell} o_s + e^{\ell_l - \ell} o_l.}$$

- The coefficients are the two sides' joint attention masses.
- They sum to one and depend on query–key scores.
- Separate cache storage still permits exact joint normalization.

Teaching derivation of the log-sum-exp merge described in Appendix F, PDF p. 24.