Memory-Efficient Looped Transformer

Decoupling Compute from Memory in Looped Language Models

Paper discussion based on Conchello Vendrell et al.

arXiv:2605.07721v2 (19 May 2026)
Reading: 20 September 2026

The result is a memory–quality trade-off

- At four loops, MELT uses $\frac{1}{4}$ of Ouro's KV bytes per token.
- MATH-500: 93.4 vs. 94.4; HumanEval: 81.7 vs. 76.8.
- One learned memory per token and layer replaces loop-specific caches.
- Constant memory refers to loop depth, not sequence length.

Paper, Tables 1–2, PDF pp. 6–7. Scores are percentages.

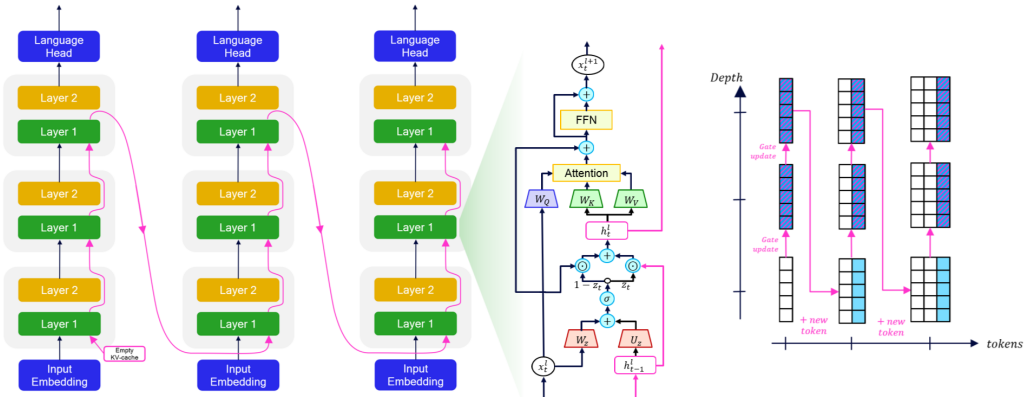
Weight sharing does not imply cache sharing

$$F^{(T)} = \mathrm{lmhead} \circ \underbrace{\mathcal{M} \circ \cdots \circ \mathcal{M}}_{T \text{ repeated stacks}} \circ \mathrm{emb}$$

- N distinct layers; L tokens; T loops; hidden width d.
- Ouro retains per-loop KV states: $O(NLT)$ cache scaling.
- MELT targets $O(NL)$ while preserving full token-axis attention.
- More loops still require more computation.

Paper, Sec. 3.1–3.2, PDF p. 3; fixed width and precision.

MELT overwrites the current token's memory



One new row per token; recurrent updates refine that row.

Paper, Fig. 2, PDF p. 4. Original author artwork; two layers and three loops shown.

A feature-wise gate chooses what to retain

$$z_t^{(l)} = \sigma(x_t^{(l)} W_z^{(l)} + h_{t-1}^{(l)} U_z^{(l)} + b_z^{(l)})$$
$$h_t^{(l)} = z_t^{(l)} \odot h_{t-1}^{(l)} + (1 - z_t^{(l)}) \odot x_t^{(l)}$$
$$k_t^{(l)} = h_t^{(l)} W_K^{(l)}, \qquad v_t^{(l)} = h_t^{(l)} W_V^{(l)}$$

- x: current hidden state; h: retained memory; $z \in (0,1)^d$.
- $z \approx 1$ retains; $z \approx 0$ overwrites. Gates vary by feature.
- Update latent memory, project to KV, then apply attention and FFN.

Paper, Sec. 3.2, Eq. (1), PDF p. 4.

Earlier loops are compressed, not stored verbatim

Teaching example: one dimension, fixed gate $z = 0.2$

$$h_3 = 0.008h_0 + 0.032x_1 + 0.16x_2 + 0.8x_3.$$

- Later tokens read the final memory of earlier tokens.
- Token positions remain separately addressable by attention.
- The real gate is learned and input-dependent; this is not a measured gate trajectory.

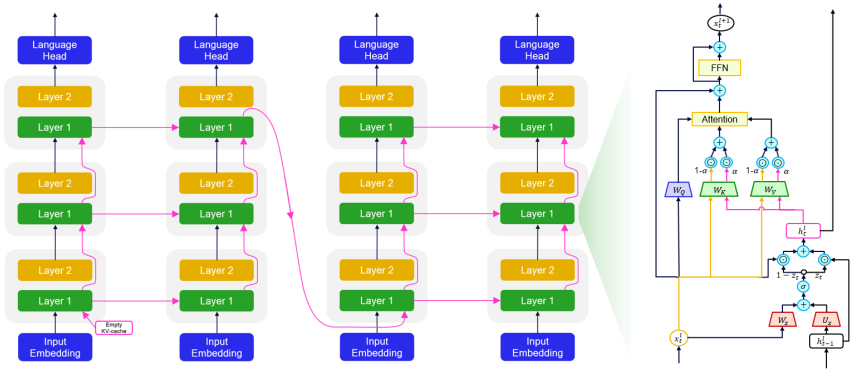
Teaching derivation from Eq. (1), PDF p. 4.

New cache semantics create a training dependency

- At inference, token n+1 uses token n's completed recurrent state.
- Fully parallel same-loop training uses a different history.
- Fully sequential token training is expensive.
- Chunk-wise training processes chunks sequentially and tokens within each chunk in parallel.

Paper, Sec. 3.3, PDF pp. 4–5.

Chunks trade fidelity for training throughput



Smaller chunks better approximate autoregressive inference.

Paper, Fig. 3, PDF p. 5. Example chunk = 2; actual training chunk = 500 tokens.

Phase 1 gradually changes the attention inputs

$$KV = \alpha KV_{\mathrm{MELT}} + (1-\alpha) KV_{\mathrm{base}}, \qquad \alpha : 0 \to 1$$

- Initialize from Ouro-1.4B-Thinking; randomly initialize new gates.
- Fine-tune the full model with chunk-wise training.
- Distill from the original LoopLM at all reasoning loops.
- 500 steps / 160M tokens; interpolation avoids an abrupt switch.

Paper, Sec. 3.3, PDF p. 5; Table 6, PDF p. 18.

Phase 2 anchors attention outputs to the teacher

$$\mathcal{L} = \mathcal{L}_{\mathrm{KD}} + \frac{\beta}{NT} \sum_{l=1}^{N} \sum_{t=1}^{T} \left\| o_{\mathrm{MELT}}^{(l,t)} - \mathrm{sg}\left(o_{\mathrm{LoopLM}}^{(l,t)} \right) \right\|_2^2$$

- o: post-attention representation; sg: stop-gradient.
- Align outputs, not attention probability matrices.
- Frozen teacher; pure MELT dynamics; $\beta = 0.1$.
- 300 more steps / 96M tokens to reduce representation drift.

Paper, Sec. 3.3, PDF pp. 5–6; Appendix C, Fig. 5, p. 18.

The adaptation uses 256M training tokens

Setting	Value
Layers / hidden width / loops	24 / 2048 / 4
Added gating parameters	About 0.2B
Chunk / sequence / batch tokens	500 / 10K / 320K
Data mix	50% AceReason + 50% OpenThoughts3
Model / gate learning rate	$8 \cdot 10^{-6}$ / $5 \cdot 10^{-4}$
Main training compute	1,040 H100 GPU-hours

The main run is smaller than the full project's reported compute budget.

Paper, Table 6, PDF p. 18. Full project estimate: about 20,000 GPU-hours.

Evaluation measures quality, not equal-compute gains

- Teacher: Ouro-1.4B-Thinking. Non-looped models: Qwen3-1.7B, Gemma4-E2B, Qwen3.5-2B, DeepSeek-R1-1.5B.
- LightEval 0.8.1; temperature 1.0; top-p 0.7; completion cap 32k.
- AIME/AMC: 16 completions per question, pass@1 and pass@10.
- Other tasks: one completion; HLE uses a 300-question subset.
- Training histories and inference FLOPs are not matched.

Paper, Sec. 4.1, PDF pp. 6–7; Table 7, PDF p. 19.

Math performance approaches the teacher

Task	Ouro	MELT	Qwen3-1.7B
AIME24	50.2	46.7	43.1
AIME25	36.7	33.3	33.1
AIME26	44.0	41.0	31.7
AMC23	81.2	80.2	79.2
MATH-500	94.4	93.4	90.6
OlympiadBench	67.5	64.7	63.5
Macro average	62.3	59.9	56.9

Paper, Table 1, PDF p. 6. Percent; AIME/AMC pass@1, others accuracy. Uncertainty omitted here.

Non-math gains are task-dependent

Task	Ouro	MELT	Qwen3-1.7B
GPQA	40.8	42.6	37.3
HLE	2.7	2.0	1.3
MMLU-Red	74.2	74.2	73.8
HumanEval	76.8	81.7	71.3

- HumanEval rises by 4.9 percentage points vs. Ouro.
- MELT does not win every task: Qwen3.5 scores 45.1 on GPQA.
- HLE accuracy remains very low; reported errors do not establish significance.

Paper, Table 1, PDF p. 6. Percent accuracy; uncertainty omitted here.

The robust memory result is a 4x KV ratio

Model	KV MB/token	Reported total GB at 32k
MELT	0.196608	9.49
Ouro	0.786432	27.97
Qwen3-1.7B	0.114688	7.07

- Ouro/MELT KV ratio = 4; reported total ratio $\approx$ 2.95.
- Total is estimated weights + KV, not measured runtime peak.
- The table's units and sums are inconsistent; see backup audit.
- No end-to-end latency or throughput result is reported.

Paper, Table 2 and Sec. 4.3, PDF p. 7. Values reproduced as reported.

Learned feature-wise gating beats simple aggregation

Phase-1 variant	AIME24	AIME25	MATH-500
Element-wise MELT	44.8	32.9	92.8
Mean	29.0	23.3	83.2
EMA-0.2	30.2	21.5	84.6
Last	33.7	24.0	84.0
Scalar gate	34.4	23.1	85.6

Dynamic feature selection helps, but gating also adds parameters.

Paper, Table 3, PDF p. 8. AIME pass@1; MATH-500 accuracy; percentages.

Training components matter as a combined recipe

Cumulative removal	AIME24 pass@1	MATH-500
Full recipe	46.7	93.4
Remove attention alignment	44.8	92.8
Also remove interpolation	35.4	86.6
Also remove all-loop KD	35.8	85.2
Also remove chunking	0.0	0.0

Cumulative ablations do not isolate independent marginal effects.

Paper, Table 4, PDF p. 9. Phase 2 also adds training steps; scores in percent.

Some gains reverse for multi-sample success

Metric	Phase 1	Full model
AIME25 pass@10	66.1	61.9
AMC23 pass@10	99.3	97.8

- Phase 2 improves the reported pass@1/accuracy columns.
- It does not improve every metric.
- The paper's claim of uniformly consistent improvement is too broad.

Paper, Table 4, PDF p. 9; interpretation based on reported values.

Untrained cache sharing fails in this long-reasoning setup

- Four variants: first/last-loop sharing, with/without original prompt cache.
- All score zero on the four tested math benchmarks.
- Qualitative generations start coherently, then deteriorate.
- Authors hypothesize accumulated cache mismatch and protocol differences.
- This motivates adaptation; it is not a universal impossibility result.

Paper, Appendix B, Table 5 and Fig. 4, PDF pp. 16–17.

The stability argument is conditional

$$J_t = \mathrm{diag}(z_t) + \mathrm{diag}(h_{t-1} - x_t)\frac{\partial z_t}{\partial h_{t-1}} + \mathrm{diag}(1 - z_t)\frac{\partial x_t}{\partial h_{t-1}}$$

- If $z_t \to 1$ and other terms stay controlled, $J_t \to I$.
- Near saturation: $(1-\epsilon)^T$ suggests requiring $\epsilon \ll 1/T$.
- A bounded gate alone does not bound the full Jacobian.
- Four-loop experiments do not validate arbitrary-depth stability.

Paper, Appendix E, Eqs. (2)–(3), PDF pp. 19–20; final two points are reader analysis.

What is established, and what remains open

- Established: learned loop-axis compression preserves useful quality with much less KV storage.
- Current scope: one pretrained family, 1.6B scale, four fixed loops.
- Open: adaptive depth, GQA, more parallel training, larger models.
- Needed evidence: depth/chunk sweeps, runtime peak memory, throughput, matched-compute comparisons.

Paper, Sec. 5, PDF p. 9; proposed additional evaluations are reader analysis.

Reference and discussion

Memory-Efficient Looped Transformer: Decoupling Compute from Memory in Looped Language Models

Victor Conchello Vendrell, Arnau Padrés Masdemont, Niccolò Grillo, Jordi Ros-Giralt, Arash Behboodi, Fabio Valerio Massoli.

arXiv:2605.07721v2, 19 May 2026. Qualcomm AI Research.

Discussion: What survives when training tokens, inference FLOPs, and actual peak memory are matched?

Reading uses the versioned PDF and author TeX; no experiments were independently reproduced.

Backup: memory arithmetic needs clarification

- Text specifies 32,768 tokens. In decimal GB: $32768 \times 0.196608/1000 = 6.44245$, not 6.29.
- 6.29 is close to the 32,000-token calculation.
- The MELT row also gives $3.272 + 6.29 = 9.562$, not 9.49.
- Consistent 32,768-token totals from listed weights and KV: MELT 9.71445, Ouro 28.63880, Qwen3 7.20010 GB.
- These are arithmetic checks, not new measurements.

Paper, Table 2, PDF p. 7; calculations by the reader.

Backup: use the tables when prose disagrees

- Fig. 1 left panel says MATH-500 below it, but the y-axis is AIME26 pass@1.
- Sec. 4.2 incorrectly identifies several winning baselines.
- Table 1: Gemma4 leads AMC23 pass@1; Qwen3.5 leads GPQA.
- MMLU-Red: MELT 74.2, Gemma4 and Qwen3.5 both 75.3.
- These issues qualify broad claims without reversing the KV ratio.

Paper, Fig. 1, p. 1; Table 1, p. 6; Sec. 4.2, p. 7.

Backup: reproducibility details and limits

- Appendix F reports that the authors could not fully reproduce the original Ouro paper's scores.
- Their inspected early-exit implementation selected logits without skipping all later loop computation.
- These are author observations, not an audit of current code.
- v2 promises a future code/model release; later availability was not checked.
- Exact KD details, initial memory handling, and runtime overhead need implementation-level verification.

Paper, Sec. 1, PDF p. 2; Appendices D and F, pp. 18–21; reader notes.