LOOPED LATENT ATTENTION: CROSS-LOOP KV COMPRESSION FOR LOOPED TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped, weight-tied Transformers reduce parameters by reusing a single block, but decoding still stores a separate key/value (KV) cache for every recurrence step. We show this loop-indexed cache is highly structured: for a fixed token, layer and head, the per-loop K/V vectors trace a short low-rank trajectory across loops, while the head and layer axes stay much flatter. We introduce Looped Latent Attention (LLA), a post-training cache codec that stores compact K and V latents and reconstructs loop-specific vectors only when attention reads them. The codec is initialized from the SVD of teacher activations and refined by logit and attention-output distillation. An LLA-2D variant folds heads into one latent for the extreme-compression regime. At matched cache budget, per-head LLA beats head-axis MLA, cross-layer sharing, KV quantization and zero-parameter final-loop reuse. The last result shows that an unadapted endpoint cannot replace the loop trajectory. The advantage holds on Ouro-2.6B-Thinking and transfers to Huginn-3.5B, where an SVD codec is near-lossless to $32\times$ on decoder-independent fidelity. The same full-RoPE codecs that produce the accuracy results also serve: with a fused latent-read kernel, LLA decodes end to end on one B200 at $1.2\times$ the teacher's fixed-batch throughput and 2.6–$5.9\times$ its peak throughput while caching 4–$21\times$ less, with decoder-independent quality unchanged. Serving through the streaming latent store exposes a train/serve mismatch, which we trace to drift of the stored K/V of generated tokens and remove by fitting the codec under the served semantics with an exact 64-token window: served GSM8K rises from 0.31 to 0.67 at $3.8\times$ on Ouro-1.4B (oracle re-forward 0.79) and from 0.41 to 0.84 on Ouro-2.6B, matching the oracle, with the same fused kernel. An absorbed variant, LLA-D, reads latents without reconstruction and reaches $21\times$ the teacher's peak throughput. On student-generated prefixes, on-policy refinement raises MATH-500 at $4\times$ from 0.43 to 0.64–0.66 across two runs and reduces no-answer generations.

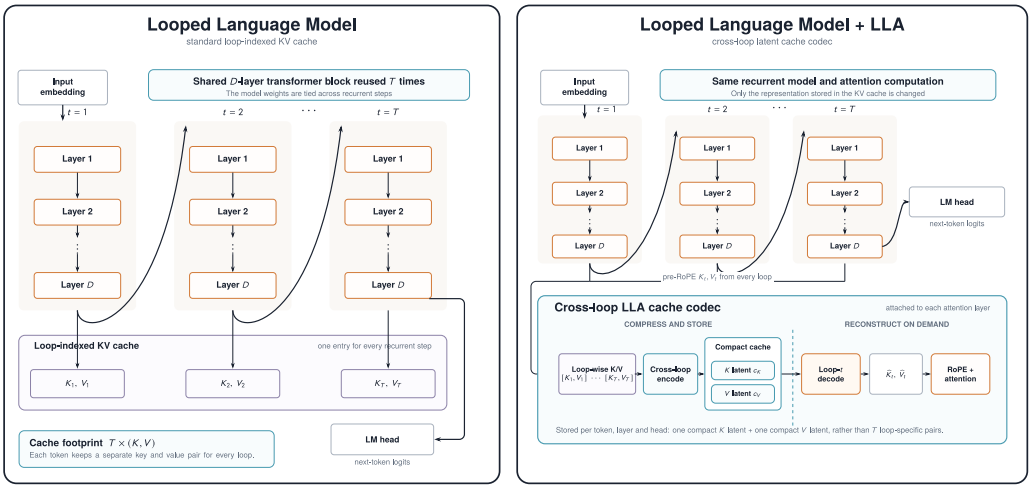
1 INTRODUCTION

At inference, the memory that constrains a Transformer is its KV cache, not its weights. Weights are loaded once and shared across the batch, whereas the cache is private to each request and grows with context and with every generated token. Past a certain length this per-request state, not the parameter count, decides how many requests fit on a device and whether batched serving stays economical.

Looped, weight-tied Transformers reduce the parameter side of this cost by applying one block repeatedly. Universal Transformers (Dehghani et al., 2019) recast depth as recurrence, and recent looped language models such as Ouro/LoopLM (Zhu et al., 2025) scale the idea to billions of parameters. The cache, however, is untied, since each recurrence step recomputes attention and writes its own keys and values, so the cache is indexed by recurrence step in addition to layer, head and token. A model with $T{=}4$ loops over $D{=}24$ layers therefore carries the KV cache of a 96-layer decoder while sharing the parameters of a 24-layer one.

This cache carries far fewer degrees of freedom than its nominal size. For a fixed token, layer and head, the per-loop K/V vectors trace a short, low-rank trajectory, and the redundancy is specific to the loop axis: at the same budget the head and layer spectra are much flatter. Looped Latent Attention (LLA) exploits this by replacing the T per-loop K/V vectors with a single latent and loop-specific reconstruction maps. A looped decoder admits a degenerate shortcut. If recurrence merely converged

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.



Looped Latent Attention (LLA). A standard looped language model writes a separate key/value cache entry at every recurrent step. LLA inserts a lightweight codec only on the cache path: it compresses the trajectory across loops into separate K and V latents, then reconstructs the requested loop's $\tilde{K}_t$ and $\tilde{V}_t$ immediately before RoPE and attention. The recurrent block, model weights, and attention computation remain unchanged.

Figure 1: **Cross-loop Looped Latent Attention.** *Left:* a weight-tied looped model reuses the same D-layer stack for T recurrent steps. In the standard implementation each step contributes a separate (K_t, V_t) to a loop-indexed KV cache, so the stored cache grows with the number of steps. *Right:* LLA replaces these loop-specific entries with a cross-loop codec. For each layer, token and head, the keys and values produced across the T steps are compressed into compact K and V latents, and at step t loop-specific up-projections reconstruct only the K_t, V_t attention requires. Reconstruction occurs before RoPE and attention, leaving the recurrent computation, weights and attention operation unchanged.

to a fixed point, one could store the final loop and reuse it. Our controlled swaps keep the teacher frozen and the attention pattern intact and change only what the cache stores. Under this direct, untrained endpoint test, final-loop reuse obtains the same $4\times$ reduction at no parameter cost yet collapses GSM8K generation to zero, while a small low-rank trajectory code holds performance at the same budget. This isolates raw endpoint substitution and does not rule out learned single-state distillation.

Contributions. (i) LLA, a cross-loop latent K/V codec for weight-tied looped Transformers that stores $(r_k{+}r_v)H$ scalars per token per layer instead of $2THd_{\text{head}}$, for compression $\rho{=}2Td_{\text{head}}/(r_k{+}r_v)$, and LLA-2D folds heads into one latent for extreme compression. (ii) We show recurrence is the most compressible cache axis in Ouro-1.4B, Ouro-2.6B-Thinking and Huginn-3.5B, that K and V follow different trajectories (motivating $r_v{>}r_k$), and that matched-budget comparisons against head-axis MLA, cross-layer sharing, KV quantization and final-loop reuse all favor the loop axis. (iii) We tie the results to deployment. An on-policy refinement objective uses student-generated prefixes to stabilize long rollouts. A fused latent-read kernel turns the codecs that produce the accuracy results into a serving method, and end to end on one B200 they match or exceed the teacher's fixed-batch throughput (210 vs 178 tok/s at $4\times$) and reach 2.6–5.9$\times$ its peak throughput. We report accuracy under both the oracle re-forward decoder and the served streaming decoder, trace the difference to drift of the stored K/V and close it with a served-semantics fine-tune. The absorbed LLA-D variant reaches $21\times$ the teacher's peak throughput, measured on the same tasks.

2 Related Work

Universal Transformers (Dehghani et al., 2019) apply a weight-tied block for several steps, and recent looped language models carry this to billion-parameter LMs (Zhu et al., 2025), and mechanistic studies report that per-loop computation settles onto low-dimensional trajectories (Blayney et al., 2026). Existing KV-cache compressors operate on single-pass Transformers and target other axes:

Multi-head Latent Attention (MLA) compresses the head/channel axis (DeepSeek-AI, 2024), cross-layer sharing targets layers (Brandon et al., 2024; Sun et al., 2024; Liu et al., 2024), quantization targets precision (Hooper et al., 2024), token eviction targets the sequence axis (Zhang et al., 2023; Xiao et al., 2024), and multi-query/grouped-query attention share heads (Shazeer, 2019; Ainslie et al., 2023). LLA instead compresses recurrence, an axis only weight-tied looped models have and one that is redundant because the same block and projections are reused every step. LLA-2D adds the head axis, and sequence eviction composes with it. The closest concurrent work, MELT (Vendrell et al., 2026), distills a single gated K/V state per layer, whereas LLA keeps the loop trajectory and compresses its rank. Our zero-parameter final-loop-reuse control tests direct endpoint copying, not learned state aggregation: it fails at the same cache budget, while a tunable trajectory latent can lie below or above the loop-count reduction through the choice of r_k, r_v.

3 Looped Latent Attention

LLA retrofits onto a frozen teacher and changes only the representation held in memory. Consider a weight-tied looped Transformer with D layers, H heads of width d_{head} and T recurrence steps. For one layer, head, token and axis (K or V) the teacher emits T row vectors, stacked as $x = [x_1; \ldots; x_T] \in \mathbb{R}^{1\times T d_{\text{head}}}$; the uncompressed cache costs $2THd_{\text{head}}$ scalars per token per layer.

Cross-loop codec. For each layer, head and axis, LLA stores a centered low-rank latent and reconstructs each loop with a loop-specific up-projection,

$$c = (x - \mu)W_{\text{down}}, \qquad \hat{x}_t = c\,W_{\text{up},t}^{\top} + \mu_t, \tag{1}$$

with $c \in \mathbb{R}^{1\times r}$, $W_{\text{down}} \in \mathbb{R}^{Td_{\text{head}}\times r}$, $W_{\text{up},t} \in \mathbb{R}^{d_{\text{head}}\times r}$, and μ_t the t-th d_{head}-wide slice of the calibration mean μ. K and V use separate latents and ranks r_k, r_v, with $r_v > r_k$ because the value cache stabilizes more slowly across loops. The compressed cache stores $(r_k+r_v)H$ scalars per token per layer, so

$$\rho = \frac{2THd_{\text{head}}}{(r_k + r_v)H} = \frac{2Td_{\text{head}}}{r_k + r_v}. \tag{2}$$

As in standard KV-cache accounting, ρ counts request-dependent state. Codec weights are shared across contexts and amortized across tokens and batches. Full reconstruction applies the shared rotary phase (Su et al., 2021) after reconstruction, whereas absorbed decoding requires the decoupled positional path analyzed in Sec. 7. **LLA-2D** also folds the head axis into one latent per layer, and its explicit maps, cache ratio and transient-memory accounting are in Appendix A.

Initialization and conversion. We collect teacher K/V activations on a calibration set, center by the per-loop mean, and initialize W_{down} from the top-r right singular vectors of the stacked-loop matrix (loop slices give the initial $W_{\text{up},t}$). The SVD codec is then fine-tuned with the teacher frozen under a forward-KL plus attention-output objective,

$$\mathcal{L} = \text{KL}(p_{\text{teacher}} \| p_{\text{swap}}) + \frac{\lambda_{\text{attn}}}{TD} \sum_{t,d} \text{MSE}(a_{t,d}, \hat{a}_{t,d}), \tag{3}$$

where $a_{t,d}$ and $\hat{a}_{t,d}$ are attention-module outputs and MSE averages all batch, token and channel entries. Only codec parameters are updated and λ_{attn}=0.5. The SVD prior dominates every other choice. At 4× it reaches held-out KL 0.105 after a short conversion, versus 0.771 for random initialization at the same budget (Appendix Table 8). **Matched-axis baselines** use the identical recipe with only the codec architecture changed: per-loop MLA (head axis), cross-layer K/V sharing (layer axis), low-bit KV quantization (precision), and a zero-parameter control that caches only the final loop and reuses it, which at T=4 has the same 4× footprint as LLA.

On-policy refinement. Eq. (3) is teacher-forced. For long rollouts the swapped model instead conditions on prefixes produced by its own codec, so we add an optional second stage in which the swapped model samples completions, the frozen teacher scores those sampled prefixes, and only codec parameters are updated (no reward or correctness filtering). Sec. 6 gives both objectives.

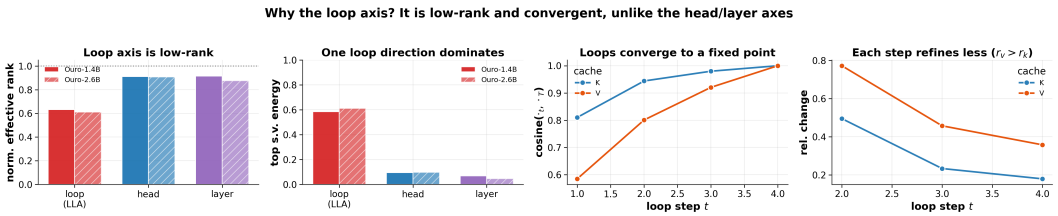


Figure 2: **Cross-loop K/V is low-rank, and its per-loop trajectory stabilizes.** *Left two panels:* the loop axis has much lower normalized effective rank than the head or layer axes, and a single direction carries a large fraction of the energy. *Right two panels:* K and V move toward the final loop while the step-to-step change shrinks, and V converges more slowly than K, motivating $r_v > r_k$.

4 Cross-loop structure in the teacher cache

Before training any codec we ask whether the frozen teacher already has the structure LLA assumes, a stricter test than downstream accuracy because it reads the cache tensor directly. **The loop axis is low-rank where the head and layer axes are not.** In Ouro-1.4B and Ouro-2.6B-Thinking the loop axis has normalized effective rank 0.61–0.63 with one direction carrying approximately 60% of the energy, while the head and layer axes are near full rank (Fig. 2). Low rank is read from the spectra, not inferred from convergence. **The trajectory explains the rank split and rules out endpoint reuse.** Across steps K and V move toward the final loop while step-to-step change shrinks. K stabilizes earlier and V carries more cross-loop variance, so we allocate $r_v > r_k$ (a matched-budget $96/160$ split beats $128/128$). Loop-to-loop variation is nonetheless 6% of total K and 17% of total V variance across tokens, and the PCA paths are short, consistent trajectories rather than a shared endpoint (Appendix Fig. 11), which is why the zero-parameter endpoint-copy control fails below. Huginn-3.5B, a different recurrent-depth architecture with up to 32 steps (Geiping et al., 2025), shows the same redundant loop axis and slower V convergence, supporting LLA as a property of weight-tied iteration rather than an Ouro artifact.

5 Experiments

At a fixed cache budget we test whether compressing recurrence beats compressing heads, layers or precision, then transfer to a larger Ouro model and to Huginn. The main teacher is Ouro-1.4B with $T{=}4$. Generative results use the faithful cached-reconstruction decoder and held-out KL is decoder-independent (plots retain the legacy "train-KL" label). All learned iso-cache baselines share data, optimization and selection (Appendix A).

Iso-cache axis comparison. Compressing the loop axis preserves GSM8K where the head, layer and precision baselines and zero-parameter final-loop reuse do not (Table 1, Fig. 3). At $4\times$, LLA reaches 0.800 GSM8K against 0.522 for head compression, 0.660 for layer sharing and 0.702 for precision quantization, and direct endpoint reuse collapses to 0.000 (K/V relative error 0.35/0.60), which rejects unadapted endpoint copying, not learned state aggregation. Precision is competitive at $4\times$ but falls at $5.33\times$. At $21.3\times$, LLA still reaches 0.740 GSM8K and 0.622 MC-average, and although LLA and head compression then have identical held-out KL (0.128) they score 0.740 versus 0.680 on GSM8K: teacher-forced reconstruction alone does not predict rollout quality, and the recurrent allocation matters even at matched budget. LLA-2D is competitive on teacher-forced metrics but trails per-head LLA on generation. Across the broad suite (Appendix Fig. 12) code matches or exceeds the teacher through $2\times$ and MMLU-Pro changes little, GSM8K and BBH degrade gradually up to $2\times$, and MATH-500 is the most sensitive task, which motivates Sec. 6. Appendix A distinguishes a separate rank sweep whose $4\times$ GSM8K of 0.590 uses different checkpoints, subset and decoder from Table 1.

Scale and family transfer. On Ouro-2.6B-Thinking, LLA remains near-lossless on full GSM8K from 4 to $10.67\times$ (0.834, 0.817, 0.821 against a teacher near 0.83, held-out KL 0.057–0.070), while head compression drops to 0.585 at $4\times$ and precision quantization to 0.351 at $5.33\times$. On the eight-task decoder-independent suite (Appendix Table 16) LLA averages 0.645, 0.646 and 0.640 against the

Table 1: **Iso-cache comparison across compression axes** (Ouro-1.4B). Learned codecs are budget-matched, and highlighted LLA rows compress recurrence and each ratio is independently optimized. Final-loop reuse is the $4\times$ zero-parameter endpoint-copy control. *kv_quant* is a simple inference-only precision control, not a state-of-the-art quantizer. Neither control reports held-out KL.

Method	ρ	held-out KL ↓	GSM8K ↑	MC-avg ↑	MMLU ↑
teacher	$1\times$	n/a	0.794	0.646	0.682
final-loop reuse	$4\times$	n/a	0.000	0.443	0.537
LLA (loop)	$4\times$	**0.059**	**0.800**	0.640	**0.672**
mla_head (head)	$4\times$	0.122	0.522	0.585	0.657
cla (layer)	$4\times$	0.267	0.660	0.597	0.666
kv_quant (prec.)	$4\times$	n/a	0.702	**0.645**	0.672
LLA (loop)	$5.33\times$	0.062	0.752	**0.640**	**0.674**
mla_head	$5.33\times$	0.123	0.180	0.576	0.640
LLA-2D	$5.33\times$	0.078	**0.760**	0.628	**0.674**
kv_quant (prec.)	$5.33\times$	n/a	0.122	0.621	0.607
LLA (loop)	$21.33\times$	0.128	**0.740**	0.622	0.668
mla_head	$21.33\times$	0.128	0.680	0.613	0.633
LLA-2D	$21.33\times$	**0.097**	0.723	**0.624**	**0.670**

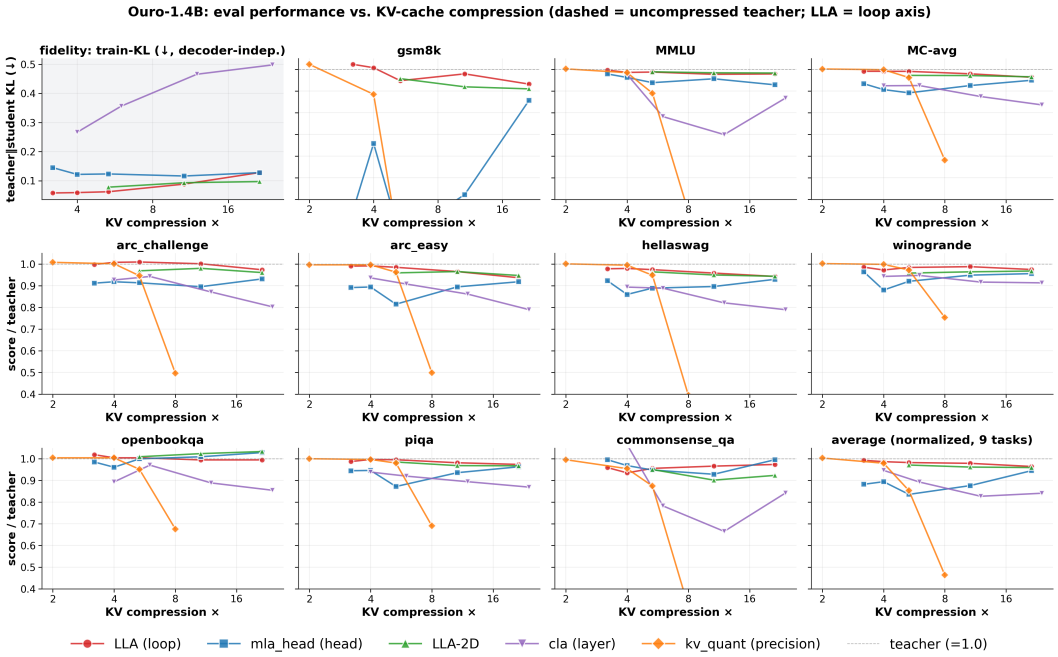


Figure 3: **Ouro-1.4B performance versus KV-cache compression.** Each panel plots a metric against compression for per-head LLA, LLA-2D (loop×head), and the head, layer and precision baselines. The dashed line is the uncompressed teacher. LLA tracks the teacher most closely across the suite, while precision quantization collapses on GSM8K past moderate compression.

teacher's 0.653, the only method within 1.3 points at every budget, while head compression trails by 3.9–5.4 points. Huginn-3.5B changes the recurrent architecture and uses 32 steps: a training-free SVD codec is near-lossless to $32\times$ (held-out KL ≤ 0.010, top-1 ≥ 0.95) and short distillation recovers useful fidelity at 64–$128\times$ (KL 0.145/0.177, Appendix Table 17), with MC-average unchanged through $32\times$ (0.523/0.525 vs 0.523). A linear subspace of teacher activations capturing most of the trajectory is what one expects if weight-tied iteration itself is the source of the low rank. **Ablations.**

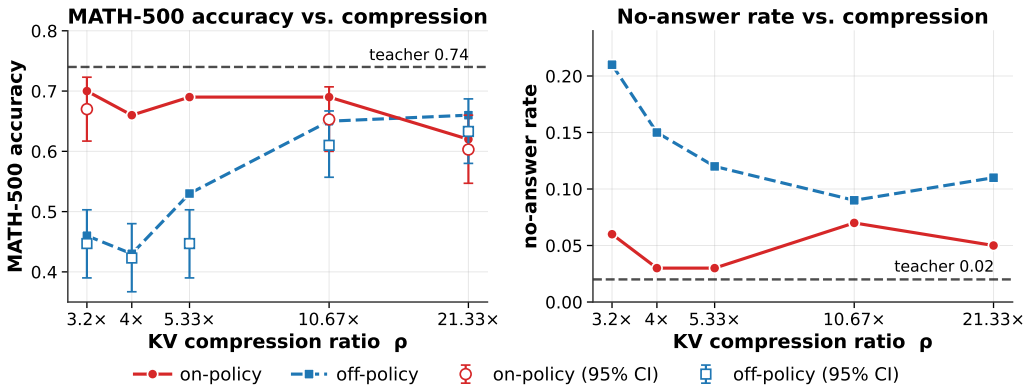


Figure 4: **On-policy distillation stabilizes the long-generation Pareto.** On MATH-500, self-rollout distillation raises the low-compression accuracy points and reduces no-answer generations throughout the Pareto. The horizontal axis is cache compression ρ. Hollow markers are independent re-evaluations with bootstrap 95% confidence intervals.

SVD initialization is the largest factor, cutting held-out KL by an order of magnitude at matched budget (0.105 vs 0.771 at $4\times$); the asymmetric rank split beats a shared rank and attention-output matching adds a small gain (Appendix Table 8). LLA mostly exploits teacher geometry rather than learning a new attention mechanism.

6 Long-generation stability

Teacher-forced conversion trains on teacher trajectories, but at inference the swapped model conditions on its own codec's tokens, so a small early reconstruction error moves later prefixes off the conversion distribution and compounds, most visibly as non-terminating MATH-500 solutions even when teacher-prefix KL is low. With p_T the frozen teacher and p_S^θ the swapped model,

$$\mathcal{L}_{\text{off}} = \mathbb{E}_{y\sim p_T}\Big[\tfrac{1}{|y|}\textstyle\sum_t \mathrm{KL}\big(p_T(\cdot|y_{<t})\|p_S^\theta(\cdot|y_{<t})\big)\Big], \quad \mathcal{L}_{\text{on}} = \mathbb{E}_{\tilde{y}\sim p_S^\theta}\Big[\tfrac{1}{|\tilde{y}|}\textstyle\sum_t \mathrm{KL}\big(p_T(\cdot|\tilde{y}_{<t})\|p_S^\theta(\cdot|\tilde{y}_{<t})\big)\Big]; \tag{4}$$

detached student sampling changes only the visited-state distribution and the frozen teacher remains the target. **On-policy refinement corrects rollout drift until rank becomes the bottleneck.** It raises MATH-500 by 0.16–0.24 at 3.2–5.33$\times$ and cuts no-answer generations throughout the Pareto (Fig. 4): at $4\times$ accuracy rises $0.43 \to 0.64$ in round 1 and to 0.66 in a validation-selected rerun while no-answer falls $0.15 \to 0.03$, and the gain grows with response length (Appendix Fig. 13). At $21.3\times$ it improves termination but not accuracy, separating exposure bias from irreversible information loss (Appendix Table 9). **Long-context retrieval** draws the same boundary: exact passkey retrieval (Appendix Fig. 14) stays perfect to 64k up to $5.33\times$, its reliable length falls to 16k at $10.67\times$ and 8k at $14.2\times$, and at $\geq 15.5\times$ retrieval is impaired at every length, while 2.6B remains perfect to 32k at $10.67\times$ where 1.4B reaches 0.80. Operationally, accuracy-sensitive serving belongs in the moderate-compression band, and the most aggressive ratios are capacity points rather than drop-in settings.

7 Serving the codecs

A latent cache is only useful if the model can be served from it, so we serve the *same* codecs that produce the accuracy results above (the *base* codecs), end to end, with a decoder whose only persistent per-request state is the latent store, and report the accuracy of that decoder. A fused Triton kernel streams (c_k, c_v) from memory, reconstructs $\hat{K}_t$, applies RoPE and takes the softmax in on-chip memory, and aggregates $\sum_j p_j c_v(j)$ in latent space, and one V up-projection follows per read. The absorbed LLA-D variant (one 512+64+512-wide state per token and layer shared across loops and heads, $15.1\times$) has a kernel that reads each block once for all heads.

Table 2: **One method, both axes: quality and end-to-end serving of the same checkpoints** (Ouro-1.4B, one B200, context 4 096). KL is held-out teacher-forced KL. GSM8K is given under the oracle re-forward decoder (first 500 problems) and the served streaming decoder (full test set). MC-avg is the six-task decoder-independent average and MMLU is zero-shot. tok/s is decode at batch 32, max batch and its aggregate throughput follow the protocol above, and cache is the measured per-token store. LLA-D serves through its latent-space kernel (oracle GSM8K on the first 200 problems). MATH-500, passkey and the remaining LLA-D variants are in Appendix Table 10. Binomial standard errors are ≤ 0.02 on 500 problems and ≤ 0.013 on the full set.

			GSM8K				tok/s	max	peak	KiB/
method	ρ	KL	oracle	served	MC-avg	MMLU	B=32	B	tok/s	tok
teacher	1.00×	0.000	0.794	0.794	0.646	0.682	178	48	188	768.0
LLA 4×	4.00×	0.059	0.788	0.308	0.640	0.672	210	192	481	192.0
LLA 4× on-policy	4.00×	0.078	0.778	0.369	0.636	0.676	210	192	481	192.0
LLA 14.2×	14.22×	0.104	0.744	0.005	0.628	0.662	223	768	933	54.0
LLA 21.3×	21.33×	0.128	0.740	0.000	0.622	0.668	207	1024	1102	36.0
LLA-D 512/64/512 oracle-trained	15.06×	0.191	0.665	0.000	0.582	0.649	214	768	3994	51.0

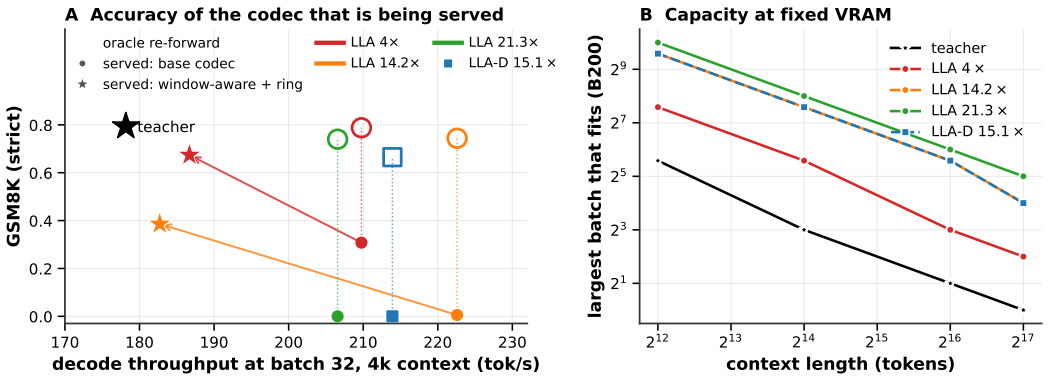


Figure 5: **The codec that is measured is the codec that is served** (Ouro-1.4B, one B200). *A:* GSM8K against decode throughput at batch 32 and 4k context. The black star is the teacher, hollow markers are oracle re-forward accuracy, small filled markers are the base codecs served through the streaming store, and stars are the window-aware codecs served with their exact ring (throughput measured with the ring). *B:* largest batch that fits at fixed VRAM against context. The hollow-to-filled distance is the served difference, and the arrow is what this section recovers.

Two decoders. Secs. 5–6 use an *oracle* re-forward decoder, in which at every step the teacher's own K/V of the sequence so far are encoded, so the latents a token reads are always those of the teacher's trajectory. A server cannot do this, because its state is the latent store and the K/V of each generated token are produced by the model *attending through latents* and then encoded. We call accuracy through this decoder *served*. Decoder-independent metrics are identical under both, but generative ones need not be, so Table 2 carries both, and Appendix Table 12 compares them on identical problems. The protocol uses one B200 in bf16 with a preallocated full KV cache and SDPA for the teacher, and only the attention read differs between arms. Decode is timed over 128 tokens after 8 warm-up steps at batch 32 and at the largest batch that fits, and served GSM8K is the full test set (Appendix F).

Full-RoPE LLA serves at the teacher's speed. With the fused kernel the 4× codec decodes at 153 ms/token against the teacher's 180 at batch 32 (210 vs 178 tok/s) and fits a 4× larger batch, for 2.6× the teacher's peak throughput. The 14.2× and 21.3× codecs reach 5.0× and 5.9× its peak. Beyond 4k the teacher falls to one sequence at 128k while the latent stores still fit batches of 4–32 (Appendix Table 18). Decoder-independent quality is unchanged when served (MC-avg 0.640/0.628/0.622, MMLU 0.672/0.662/0.668 at 4/14.2/21.3×, teacher 0.646/0.682).

Streaming decoding drifts, and we locate the drift. Through the streaming decoder the base 4× codec scores 0.308 on GSM8K against 0.788 under the oracle decoder on the same checkpoint,

Table 3: **Where served accuracy goes: teacher-forced KL along the teacher's own continuations.** For 16 GSM8K questions the teacher greedily writes up to 128 tokens, and along that fixed text we compare the next-token distribution of the oracle swap (latents from the teacher's K/V) and of the served decoder (latents of generated tokens from the model's own K/V) with the teacher's. Sampling plays no role, so the served excess is drift of the stored K/V. It is flat in position at $4\times$ and grows along the continuation from $14.2\times$ on. The oracle-trained LLA-D codec assumes a latent diagonal that a streaming decoder does not have. Served-semantics training removes that mismatch, and the exposure-consistent objective removes most of the remaining drift at a cost in teacher-forced quality.

	KL(T‖oracle)	KL(T‖served)	KL(T‖served) by position			KL(oracle‖served)	top-1
method	mean	mean	0–31	32–63	64–127	mean	T/served
LLA $4\times$	0.020	0.090	0.089	0.114	0.079	0.059	0.924
LLA $4\times$ (1-pass read)	0.020	0.099	0.097	0.117	0.093	0.073	0.927
LLA $4\times$ on-policy	0.019	0.087	0.082	0.113	0.078	0.058	0.931
LLA $14.2\times$	0.050	0.422	0.338	0.427	0.474	0.339	0.814
LLA $21.3\times$	0.065	1.205	0.665	1.142	1.552	1.095	0.662
LLA-D $15.1\times$ oracle-trained	0.089	8.328	8.283	8.299	8.389	8.161	0.039
LLA-D $15.1\times$ served-trained	0.210	1.726	0.795	1.664	2.271	1.465	0.534
LLA-D $15.1\times$ served-trained + self-latents	0.322	0.504	0.418	0.539	0.538	0.194	0.799

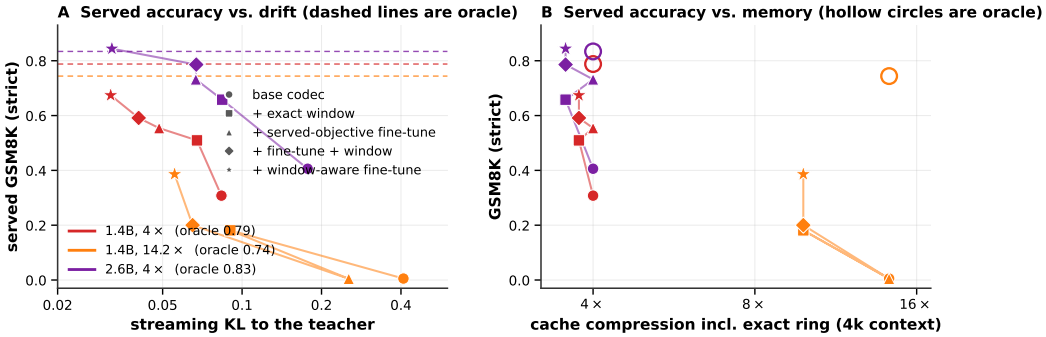


Figure 6: **Fitting the codec to the served semantics closes the gap.** One path per budget: base codec, exact ring, served-objective fine-tune, both, window-aware fine-tune (stars). *A:* served GSM8K against streaming KL, with dashed lines at oracle accuracy. *B:* the same against cache compression including the ring, with hollow circles at oracle accuracy. Numbers in Table 4.

and the on-policy $4\times$ codec 0.369. At $14.2\times$ and $21.3\times$ served GSM8K falls to 0.005 and 0.000 (oracle 0.744 at $14.2\times$), and served MATH-500 at $14.2\times$ to 0.01 against 0.64. Table 3 locates the loss with a teacher-forced KL along the teacher's own continuations, so sampling plays no role: the served decoder's KL to the teacher is 0.084 at $4\times$ (oracle 0.018) and flat in position, 0.41 at $14.2\times$ (oracle 0.044) rising along the continuation, and 1.21 at $21.3\times$ (oracle 0.065), and top-1 agreement drops from 93% to 66%. The stored K/V of generated tokens drift from the teacher's and the drift compounds through the store. The on-policy stage leaves served KL unchanged (0.087).

The absorbed kernel sets the throughput ceiling. LLA-D at $15.1\times$ matches the fixed-batch speed of LLA (214 tok/s) and, because its read never reconstructs and shares one store across heads, its throughput keeps scaling with batch, reaching 3 994 tok/s at batch 768, $21\times$ the teacher and $4.3\times$ LLA at the same budget. Oracle-trained it keeps MC-avg 0.582 and MMLU 0.649, but it was trained reading every position, including the current token, through a latent, while a streaming decoder has that token's exact K/V: KL 8.3 from the first generated token. Training under the served semantics, with or without an exposure-consistent objective, removes the mismatch, and closing the remaining drift under the absorbed read (served GSM8K ≤ 0.006, Appendix F) is the target for native latent-cache training. Batch-1 decode ($\approx$ 150 vs 63–68 ms/token) and the two-pass prefill (Appendix Table 13) are the remaining systems costs.

Fitting the codec to the served semantics. The difference is a training-protocol mismatch, so we address it from both sides, leaving the kernel and the store unchanged (Fig. 6, Table 4). *Served-*

Table 4: **Closing the served gap with the kernel and the latent store unchanged.** Streaming KL to the teacher along GSM8K continuations (Table 3 protocol, 64 questions), served GSM8K (strict; flexible in parentheses where it differs by more than two points), MC-avg, and the per-token store at 4k with its ratio to the teacher. The fine-tune runs 2000 steps (1400 at 2.6B) from the base codec on teacher completions, with the generated tokens' latents re-encoded from the model's own K/V over three hops and averaged with the oracle objective. Window W keeps the exact K/V of the last W tokens in a ring, and window-aware places the ring inside the loss (800 steps, 1850 at $14.2\times$, 600 more at 2.6B). All ablations: Appendix Table 11.

codec / configuration	streaming KL	served GSM8K	MC-avg	KiB/tok	ρ
Ouro-1.4B, LLA $4\times$; oracle GSM8K 0.788 (first 500)					
base codec	0.084	0.308	0.640	192	4.0
+ exact window $W{=}64$	0.067	0.509 (0.538)	0.640	204	3.8
+ served-objective fine-tune	0.048	0.554	0.638	192	4.0
+ fine-tune + window $W{=}64$	0.041	0.591 (0.632)	0.638	204	3.8
+ window-aware fine-tune, $W{=}64$	0.032	0.674	0.639	204	3.8
Ouro-1.4B, LLA $14.2\times$; oracle GSM8K 0.744 (first 500)					
base codec	0.409	0.005	0.628	54	14.2
+ exact window $W{=}128$	0.090	0.181 (0.206)	0.628	78	9.8
+ served-objective fine-tune	0.254	0.005	0.623	54	14.2
+ fine-tune + window $W{=}128$	0.065	0.200 (0.314)	0.623	78	9.8
+ window-aware fine-tune, $W{=}128$	0.056	0.386	0.625	78	9.8
Ouro-2.6B-Thinking, LLA $4\times$ *(2-shot)*; oracle GSM8K 0.834 (full set)					
base codec	0.177	0.406	0.662	384	4.0
+ exact window $W{=}128$	0.084	0.658	0.662	432	3.6
+ served-objective fine-tune	0.067	0.732	0.666	384	4.0
+ fine-tune + window $W{=}128$	0.067	0.786	0.666	432	3.6
+ window-aware fine-tune, $W{=}128$	0.032	0.844	0.667	432	3.6

objective fine-tuning re-encodes the generated tokens' latents from the K/V the swapped model produces when it reads the previous hop's latents, for three hops of the exposure recursion, averaged with the oracle objective: at $4\times$ streaming KL falls from 0.084 to 0.048 and served GSM8K rises to 0.554 with MC-avg, MMLU and oracle GSM8K within 1.4 points ($0.788 \rightarrow 0.774$ on the same problems). *An exact recent window* keeps the exact K/V of the most recent W tokens in a ring next to the latent store, capping the recursion depth for $+12/+23$ KiB per token at 4k ($W{=}64/128$, $+3/+6$ at 16k). The same fused kernel reads it, moving the $4\times$ decode at 4k and batch 32 from 153 to $171/174$ ms/token ($187/184$ tok/s, teacher 178) with the max batch unchanged, and alone it raises served GSM8K to 0.509 at $4\times$ and 0.181 at $14.2\times$. The two compose (0.591, 0.200). The largest gain comes from placing the ring *inside the objective*. a codec fitted to the exact-plus-latent context it sees at serving time reaches 0.674 at $4\times$ after 800 steps (0.500 on served MATH-500 against 0.14), whereas 2000 further ring-free steps stay at 0.592. At $14.2\times$ it reaches 0.386 after 1850 steps (0.277 after 800), the best served accuracy at that budget, and keeps the few-shot #### format that the ring-free three-hop codec drops (0.087 strict, 0.374 flexible, and 0.300 on served MATH-500 against 0.01). At $21.3\times$ the window alone cuts KL from 1.21 to 0.19 (served GSM8K ≤ 0.005), so there a served-semantics codec rather than a post-hoc window is required. Nor is this a 1.4B artefact: on Ouro-2.6B-Thinking at $4\times$ (2-shot, 500 problems) the streaming decoder scores 0.406. $W{=}128$ raises it to 0.658, a 1400-step fine-tune to 0.732, both to 0.786, and 600 window-aware steps to 0.844, matching the oracle (0.834 full set, $0.820/0.845$ for the base/fine-tuned codec on the first 200 problems, Table 12), with MC-avg unchanged (0.667 vs 0.662). The measured codec is the served codec, at oracle accuracy above the teacher's throughput.

8 Discussion and conclusion

The evidence supports recurrence-specific, not axis-agnostic, cache redundancy: reapplying one block to a stabilizing recurrent state produces a much steeper loop spectrum than the head or layer spectra, matched-budget codecs on those axes lose more accuracy, the ordering transfers to Ouro-2.6B and Huginn, and direct endpoint reuse collapses at the same budget. Rank sets the regime, from near drop-in to $2\times$ and on-policy-assisted to $5\times$ to a capacity tool beyond $10\times$. The codecs that establish accuracy are the ones we serve, above the teacher's throughput at every budget. Fitted under the served semantics with a small exact window they reach oracle accuracy at 2.6B and within eleven points at $4\times$ on 1.4B. Native latent-cache training is the next step. Recurrence is thus a first-class compression axis for looped decoders, measured on the decoder that serves it.

AI USE STATEMENT

AI tools were used to identify grammatical mistakes and to select parts of the paper that needed more detail. The authors ran every experiment, reviewed all AI-assisted changes, and take responsibility for the final text, claims, and artifacts.

ETHICS STATEMENT

This work uses publicly released models and standard public benchmarks and does not involve human subjects or private data. Cache compression can reduce the memory and compute needed to serve looped language models, but aggressive compression may also change model behavior. Deployments, especially in safety-critical settings, should validate task-specific accuracy and failure rates at the selected compression level.

REPRODUCIBILITY STATEMENT

Section 3 specifies the codec, initialization, objectives, and exact cache accounting. Appendix A reports the models, calibration data size, optimization schedule, evaluation protocols, and hardware, and the remaining appendices provide rank sweeps, per-task results, and long-context diagnostics. An anonymized implementation and configuration files should be included with the submission materials.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Hugh Blayney, Alvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael M. Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026.

William Brandon et al. Reducing transformer key-value cache size with cross-layer attention. *arXiv preprint arXiv:2405.12981*, 2024.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Coleman Hooper et al. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

Akide Liu et al. Minicache: Kv cache compression in depth dimension for large language models. *arXiv preprint arXiv:2405.14366*, 2024.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Yutao Sun et al. You only cache once: Decoder-decoder architectures for language models. *arXiv preprint arXiv:2405.05254*, 2024.

Victor Conchello Vendrell, Arnau Padres Masdemont, Niccolo Grillo, Jordi Ros-Giralt, Arash Behboodi, and Fabio Valerio Massoli. Memory-efficient looped transformer: Decoupling compute from memory in looped language models. *arXiv preprint arXiv:2605.07721*, 2026.

Guangxuan Xiao et al. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Zhenyu Zhang et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A EXPERIMENTAL DETAILS

Models. All experiments use frozen, publicly released looped/recurrent-depth Transformers. **Ouro-1.4B** (the main teacher): D=24 weight-tied layers, H=16 heads of width d_{head}=128, run at T=4 recurrence steps. **Ouro-2.6B-Thinking**: a deeper Ouro variant, also T=4, for the scale study. **Huginn-3.5B**: a different recurrent-depth family with a four-layer recurrent block, H=55 K/V heads of width d_{head}=96, evaluated at R=32 recurrence steps for family transfer (Geiping et al., 2025). Teacher weights are never updated; only the K/V codec is trained.

Calibration data. We run `build_rmla_calib.py --calib 256 --seed 0` on a decontaminated 233,960-prompt pool built only from training splits of 15 Hugging Face math, code and science datasets, including GSM8K, MATH, NuminaMath-CoT, NVIDIA OpenMath/AceReason/Nemotron/OpenScience, OpenThoughts3, OpenCodeReasoning and MBPP. Domain balancing yields 85 prompts per domain (255 total); calibration, fine-tuning and evaluation partitions are disjoint by prompt SHA-1. The pool was filtered by exact 13-gram and MinHash-0.7 matching against GSM8K test, MATH-500, HumanEval, MBPP test, MMLU/MMLU-Pro, GPQA, AIME 2024/25 and BBH; the selected set has no exact or substring MATH-500 match, and its GSM8K/MATH items come only from training splits. We use the Ouro chat template with a generation prompt, truncate to 1024 tokens and collect bf16 teacher activations at T=4. The SVD pass caps data at 8,192 real tokens, so approximately the first 70–100 prompts (file order) contribute to the SVD.

LLA codec and conversion. For each layer, head and axis (K, V) we collect the teacher's pre-RoPE per-loop activations on that calibration stream and initialize W_{down} from the top-r right singular vectors of the stacked-loop matrix (loop slices give the initial up-projections). The SVD-initialized codec is then fine-tuned with the teacher frozen: objective = forward KL(teacher $\|$ swapped) + λ_{attn}=0.5 per-loop attention-output matching; 3000 steps (300 for the design ablations), batch size 8, sequence length 512, AdamW, learning rate 10^{-3} with 100-step warmup and cosine decay to 5%, held-out KL evaluated every 200 steps. We select the best-KL checkpoint (not the final step). We use $r_v > r_k$ throughout (V converges more slowly across loops). **LLA-2D** stacks the loop and head axes as x^{2D}=$[x_{1,1}; \ldots; x_{T,H}] \in \mathbb{R}^{1\times THd_{\text{head}}}$, encodes with $W^{\text{2D}}_{\text{down}} \in \mathbb{R}^{THd_{\text{head}}\times r}$, and reconstructs each (t, h) slice with $W^{\text{2D}}_{\text{up},t,h} \in \mathbb{R}^{d_{\text{head}}\times r}$. One K and one V latent are stored per token and layer, so ρ_{2D}=$2THd_{\text{head}}/(r_k + r_v)$; this gains cache reduction by coupling heads. The full-reconstruction path applies RoPE after reconstruction (the rotary phase is shared across loops for a token); only the absorbed path requires the decoupled positional treatment in Table 14.

Matched-axis baselines. All learned baselines use the identical recipe; only the codec architecture changes. *mla_head*: per-loop MLA (head-axis latent, no cross-loop sharing). *cla*: cross-layer K/V sharing over $s \in \{4, 6, 12, 24\}$ adjacent layers. *kv_quant*: a simple inference-only low-bit K/V precision control at $\{2, 3, 4, 8\}$ bits (no codec training or KL); it is not an optimized non-uniform quantizer. *final-loop reuse*: a zero-parameter endpoint-copy control that caches only the last loop's K/V and reuses it for all loops; it does not represent learned single-state distillation.

Evaluation. Accuracies are single-run point estimates; the binomial standard error is ≤ 0.013 on the full GSM8K test set (1319), ≤ 0.021 on 500 problems, ≤ 0.033 on 200 and ≤ 0.05 on 100, so differences smaller than two standard errors at the stated n are not interpreted. **GSM8K**: 5-shot at 1.4B and 2-shot at 2.6B, strict-match EM, `max_gen_toks`=256. The 1.4B teacher is scored on all 1319 test items. Main Table 1 uses a 100-item subset and checkpoints selected under the matched-axis protocol with the faithful re-forward decoder. Appendix Table 5 instead uses six independently trained rank-sweep checkpoints, a 200-item subset, an earlier streaming reconstruction decoder, and a 100-item teacher reference. These protocol differences explain 0.800 versus 0.590 at 4× and 0.794 versus 0.820 for the teacher; they are not repeated estimates of one checkpoint. At 2.6B, the teacher and per-head/head-axis arms use all 1319 items; LLA-2D uses 100. **MATH-500**: 4-shot, `max_gen_toks`=2048, batch 1, symbolic equivalence; teacher on all 500, broad-suite codecs on 200, the on-policy rank Pareto on 100, and bootstrap-CI re-evaluation on 300. **BBH**: 3-shot exact match; teacher 675 documents and codecs the same 135-document subset across arms. HumanEval/MBPP use full-set EvalPlus pass@1 (base/plus), and MMLU-Pro uses full-set exact match. The eight-task MC suite is zero-shot loglikelihood accuracy on full sets and decoder-independent. The legacy label

"train-KL" in plots means held-out teacher-forced KL. Passkey uses a 5-digit key at random depth, 15 trials per cell (five keys $\times$ three depths), and exact retrieval.

Decoding, caching and prefill. The faithful cached-reconstruction decoder is two-pass: pass 1 captures pre-RoPE loop K/V, the codec reconstructs it, and pass 2 re-forwards while forcing that reconstruction so the model applies its native RoPE. It is used for the iso-cache, broad-suite, 2.6B and MATH-500/BBH results; the independent rank sweep uses the earlier streaming decoder noted above. The latent-store decoder caches only the latent and reconstructs on read; because the K/V it encodes for generated tokens are the swapped model's own, its generative accuracy ("served" in Table 2) differs from the faithful decoder's ("oracle"), see Sec. 7. During autoregressive decode, uncompressed loop states exist only in a scratchpad for the active token and are discarded after compression. During prefill, complete loop trajectories are needed before down-projection; our two-pass implementation streams captures layer-by-layer and frees each layer immediately. Thus persistent cache falls by ρ, but transient prefill memory does not, explaining the 2.6B 64k OOM marker.

On-policy Stage 2. We use the first 600 prompts (file order) from the 1,333 math prompts in the fine-tuning split, identical across all five ranks and disjoint from MATH-500 and the 255-prompt calibration set. The Stage-1 student generates one completion per prompt once (temperature 1.0, top-p 0.95, no top-k, 512 new tokens, seed 1234+shard); these rollouts are reused without correctness filtering. From $3.2\times$ through $21.33\times$, respectively 527/501/483/466/206 of 600 hit the length cap without EOS. Starting from the off-policy checkpoint, we train 1,000 steps with batch size 4, sequence length 1,536, no accumulation, AdamW (5×10^{-4}, zero weight decay), 50-step warmup, cosine decay to 5%, and gradient clipping at 1.0. The loss is forward KL $+0.5$ per-loop attention-output matching on student-completion positions only; training is 100% on-policy. We report the final step-1000 checkpoint because keep-best-on-KL degenerately selected step 0. Ranks (r_k, r_v) are $(128, 192), (96, 160), (64, 128), (32, 64), (16, 32)$.

Hardware. Codecs are trained and evaluated in bfloat16; generative evaluations and the end-to-end serving measurements in Table 18 use B200 GPUs.

B Cold-start rank sweep and full quality suite

Table 5: **Independent GSM8K rank sweep for Ouro-1.4B.** Six separately trained codecs use the earlier streaming reconstruction decoder on 200 items; the teacher reference uses 100. This is not the matched-axis checkpoint/protocol of main Table 1. The plateau from $4.0\times$ to $2.67\times$ breaks at $2.0\times$.

Model	(r_k, r_v)	GSM8K strict ↑	held-out KL ↓
teacher	n/a	0.820	n/a
LLA $4.00\times$	$(96, 160)$	0.590	0.0160
LLA $3.20\times$	$(120, 200)$	0.575	0.0153
LLA $2.67\times$	$(144, 240)$	0.575	0.0157
LLA $2.00\times$	$(192, 320)$	**0.750**	0.0155
LLA $1.60\times$	$(240, 400)$	0.755	0.0070
LLA $1.33\times$	$(288, 480)$	**0.795**	0.0035

Table 6: **Quality of the independent rank-sweep checkpoint family across the broad suite.** These are not the iso-cache-selected checkpoints in main Table 1. GSM8K is 5-shot strict EM; HumanEval/MBPP use EvalPlus pass@1 (base/plus); MATH-500 uses symbolic verification. Sample sizes and decoder differences are specified in Appendix A.

Model	GSM8K	HumanEval b/p	MBPP b/p	MATH-500	BBH	MMLU-Pro
Ouro teacher (1×)	0.794	0.707 / 0.671	0.720 / 0.595	0.74	0.708	0.481
LLA 1.33×	0.795	0.756 / 0.713	0.725 / 0.606	0.70	0.705	0.467
LLA 1.60×	0.755	0.726 / 0.689	0.733 / 0.624	0.67	0.695	0.475
LLA 2.00×	0.750	0.744 / 0.695	0.733 / 0.603	0.63	0.677	0.469
LLA 2.67×	0.575	0.543 / 0.488	0.672 / 0.566	0.54	0.578	0.402
LLA 3.20×	0.575	0.591 / 0.537	0.646 / 0.537	0.72	0.495	0.399
LLA 4.00×	0.590	0.610 / 0.561	0.616 / 0.513	0.52	0.527	0.416

C Scale generalization on Ouro-2.6B-Thinking

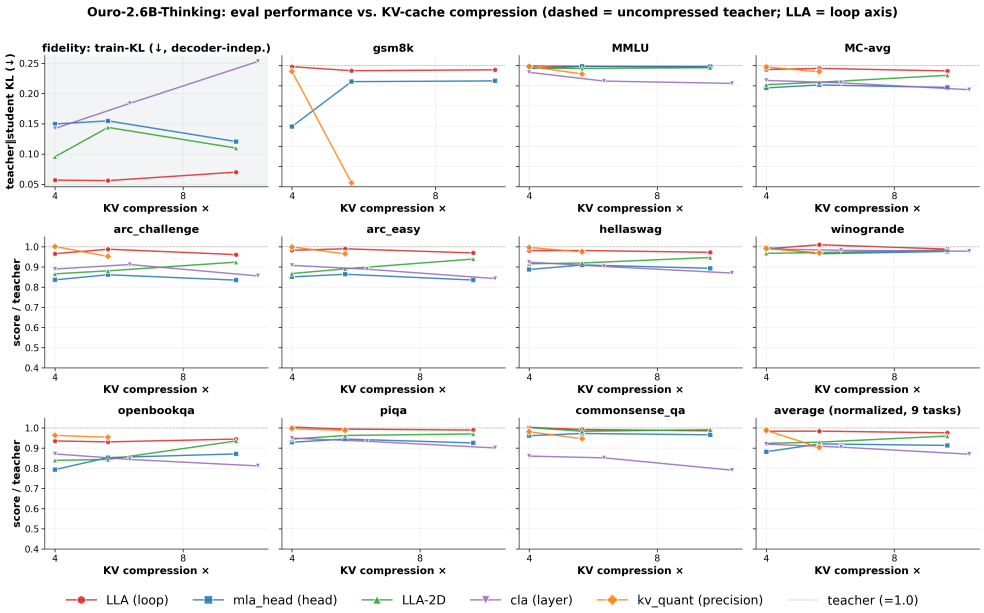


Figure 7: **Ouro-2.6B-Thinking performance versus KV-cache compression.** The loop-axis codec remains closest to the teacher on held-out KL, GSM8K and the commonsense/knowledge suite; the precision baseline cannot reach the largest ratios without collapse.

Table 7: **Full Huginn-3.5B transfer results** (R=32). Decoder-independent fidelity against the uncompressed model. MC-avg uses the SVD checkpoint at 16–32× and the distilled checkpoint at 64–128×; no generative-rollout claim is made.

ρ	r_k/r_v	SVD KL ↓	SVD top-1	distill KL ↓	best top-1	MC-avg
teacher	n/a	n/a	n/a	n/a	n/a	0.523
16×	128/256	0.005	0.970	n/a	0.970	0.523
32×	64/128	0.010	0.954	n/a	0.954	0.525
64×	32/64	0.197	0.799	0.145	0.834	0.486
128×	16/32	0.763	0.620	0.177	0.812	0.473

D Cross-loop structure: extra diagnostics

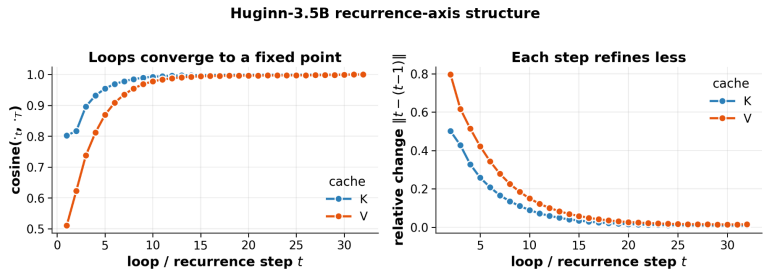


Figure 8: **The recurrence-axis mechanism transfers to Huginn-3.5B.** The same convergence diagnostics as the main paper, applied to a different family with up to 32 recurrence steps: successive steps converge toward a fixed point, each step refines less, and V converges more slowly than K.

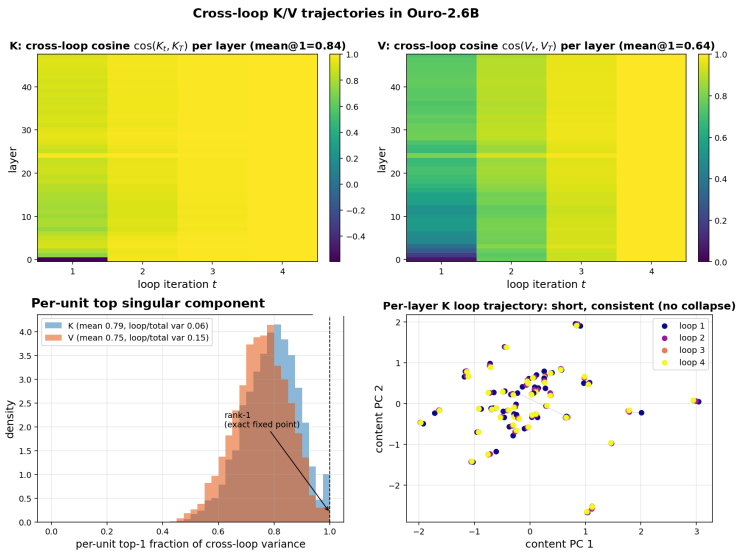


Figure 9: **Cross-loop K/V stabilization on Ouro-2.6B-Thinking.** The 2.6B model reproduces the Ouro-1.4B pattern: K/V stabilizes across loops, one direction dominates without explaining all variance, and the value cache carries higher cross-loop variance than the key cache.

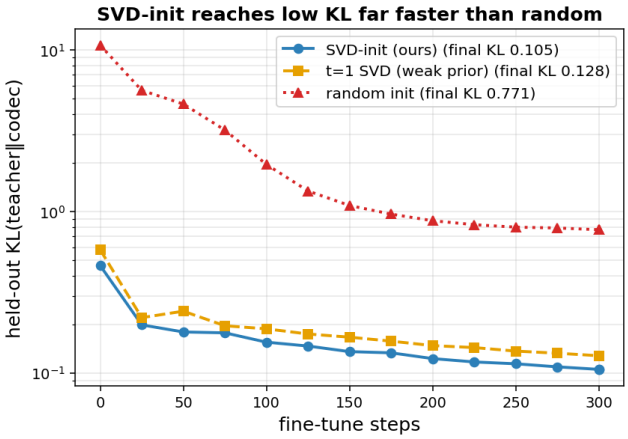


Figure 10: **SVD initialization dominates random initialization.** Held-out KL for the $4\times$ codec falls quickly from the SVD teacher prior; a random codec does not match SVD initialization within the same conversion budget.

Table 8: **Design-choice ablations.** Held-out teacher-forced KL for a $4\times$ codec after a matched 300-step conversion; the reference is SVD init, split ranks r_k/r_v=96/160, and attention-output matching.

Ablation	Variant	held-out KL ↓
init prior	SVD (ours)	**0.105**
init prior	t=1 SVD	0.128
init prior	random	0.771
rank split	$r_k<r_v$ (96/160)	**0.105**
rank split	shared (128/128)	0.129
loss	KL $+\lambda_{\text{attn}}$=0.5	**0.105**
loss	KL only	0.110

E Full on-policy results

Table 9: **On-policy versus off-policy distillation across the rank Pareto** (MATH-500, teacher 0.74). The Pareto uses 100 problems; the hollow-marker CI re-evaluation in Fig. 4 uses 300. The main figure's filled accuracy curve reports round 1 except its $4\times$ point, which uses the validation-selected rerun (0.66 versus 0.64); both complete accuracy runs are shown here. No-answer rates are round 1.

ρ	off-policy	on-policy r1	Δ r1	reselect	off no-ans	on no-ans
teacher (1×)	0.740	n/a	n/a	n/a	0.020	n/a
3.2×	0.460	**0.700**	↑0.24	0.660	0.21	0.06
4×	0.430	**0.640**	↑0.21	**0.660**	0.15	0.03
5.33×	0.530	**0.690**	↑0.16	0.680	0.12	0.03
10.67×	0.650	**0.690**	↑0.04	0.690	0.09	0.07
21.33×	**0.660**	0.620	↓0.04	0.560	0.11	0.05

F Absorbed decode and RoPE characterization

Latent-domain attention. The LLA-D rows of Tables 2 and 18 use a layer-level absorbed codec, not per-head LLA or LLA-2D: one latent per token and layer is shared across all loops and heads with d_c=512, d_r=64, and d_v=512, fitted with the same SVD-init plus KL and attention-output distillation recipe (1500 steps; held-out KL 0.19 for 512/64/512, 0.29 for 768/128/768, 0.26 for 1024/128/1024). These oracle-trained codecs read every position, including the current one, through a latent; a streaming decoder necessarily has the current token's exact K/V, and evaluating the oracle-trained codec under that semantics collapses it (MMLU 0.257, CommonsenseQA 0.188; served KL 8.3). The served-trained 512/64/512 codec of Table 2 is fitted with the true K/V on the diagonal (1500 steps, lr 3×10^{-4} after 10^{-3} diverged; held-out KL 0.46); the exposure-consistent variant re-encodes the latents from the swapped model's own K/V trajectories and runs a third forward per step (800 steps warm-started from the served-trained codec, held-out KL 0.57 under its own three-pass objective; teacher-forced oracle KL along GSM8K continuations 0.32). Neither the batch size of the evaluation (batch 1 vs 8) nor the evaluation semantics (all-latent vs served) changes the served-trained codec's MMLU (0.36–0.38).

Served-objective fine-tuning of the full-RoPE codecs. The fine-tunes of Table 11 warm-start the base codecs and run 600 steps (batch 4, 768 tokens, AdamW, cosine schedule) on the teacher's completions of the conversion prompts; the generated tokens' latents are re-encoded from the K/V the swapped model produces when it reads the previous hop's latents (one to five hops), and the "mixed" objective averages the oracle-latent and served-latent KL. Checkpoints are selected by served KL on a fixed 571-completion shard; for the 1.4B runs this shard is part of the training pool (a selection set, not a held-out set; the 2.6B and format-neutral runs use disjoint held-out completions), so no fine-tune quality is reported on it: every KL in Table 11 is measured by the streaming probe on GSM8K test continuations, and the selection-set KL of runs longer than one epoch (995 steps) drops at epoch boundaries through memorization and is not used. A second run of the $4\times$ mixed three-hop fine-tune

Table 10: **Full quality–serving table** behind Table 2 (Ouro-1.4B, one B200, context 4 096): adds MATH-500 under both decoders (oracle on the first 100 problems), served passkey at 64k, and the LLA-D variants trained under served semantics and those not served-benchmarked (768/128/768, 1024/128/1024). n/a = not measured.

			GSM8K		MATH-500				passkey	tok/s	max	peak	KiB/
method	ρ	KL	oracle	served	oracle	served	MC-avg	MMLU	64k	B=32	B	tok/s	tok
teacher	1.00×	0.000	0.794	0.794	0.74	0.74	0.646	0.682	1.00	178	48	188	768.0
LLA 4×	4.00×	0.059	0.788	0.308	0.43	0.14	0.640	0.672	0.87	210	192	481	192.0
LLA 4× on-policy	4.00×	0.078	0.778	0.369	0.66	0.37	0.636	0.676	1.00	210	192	481	192.0
LLA 14.2×	14.22×	0.104	0.744	0.005	0.64	0.01	0.628	0.662	0.47	223	768	933	54.0
LLA 21.3×	21.33×	0.128	0.740	0.000	0.66	n/a	0.622	0.668	0.00	207	1024	1102	36.0
LLA-D 512/64/512 oracle-trained	15.06×	0.191	0.665	0.000	0.61	n/a	0.582	0.649	n/a	214	768	3994	51.0
LLA-D 512/64/512 served-trained	15.06×	0.460	n/a	0.004	n/a	0.02	0.598	0.360	n/a	214	768	3994	51.0
LLA-D 512/64/512 served-tr. + self-latents	15.06×	0.571	n/a	0.006	n/a	0.01	0.564	0.310	n/a	214	768	3994	51.0
LLA-D 768/128/768 oracle-trained	9.85×	0.293	n/a	0.000	n/a	n/a	0.572	0.657	n/a	150	512	2040	78.0
LLA-D 1024/128/1024 oracle-trained	7.53×	0.258	n/a	n/a	n/a	n/a	0.573	0.669	n/a	n/a	n/a	n/a	n/a

Table 11: **All served-gap ablations** (full version of Table 4). Oracle and streaming KL to the teacher along GSM8K continuations, top-1 agreement of the streaming decoder, served GSM8K (full test set; flexible extraction in parentheses where it differs), decoder-independent MC-avg and MMLU, and the per-token store at 4k with its ratio, for every fine-tune and window configuration: one-hop and unmixed objectives, 600/2000-step fine-tunes and their continuations, format-neutral data, and the 2.6B runs. Rows without a served GSM8K entry were not run.

	KL(T‖·)		top-1	GSM8K			KiB/	
method	oracle	streaming	streaming	served	MC-avg	MMLU	tok	ρ
LLA 4× (base codec)	0.018	0.084	0.93	0.308	0.640	0.672	192	4.0
+ served-objective fine-tune (mixed, one hop)	0.014	0.073	0.93	0.442	0.639	0.672	192	4.0
+ served-objective fine-tune (mixed, 3 hops)	0.015	0.059	0.94	0.507	0.638	0.671	192	4.0
+ served-objective fine-tune (mixed, 3 hops, 2000 steps)	0.013	0.048	0.95	0.554	0.638	0.676	192	4.0
+ exact window W=64	0.018	0.067	0.94	0.509	0.640	0.672	204	3.8
+ exact window W=128	0.018	0.032	0.96	0.556	0.640	0.672	216	3.6
+ 2000-step fine-tune + window W=64	0.010	0.041	0.96	0.591	0.638	0.676	204	3.8
+ window-aware fine-tune (800 steps, W=64 in the loss) + window W=64	0.012	0.032	0.96	0.674	0.639	0.670	204	3.8
LLA 14.2× (base codec)	0.044	0.409	0.82	0.005	0.628	0.662	54	14.2
+ served-objective fine-tune (3 hops)	0.052	0.241	0.88	0.003	0.616	0.651	54	14.2
+ exact window W=128	0.044	0.090	0.93	0.181	0.628	0.662	78	9.8
+ 3-hop fine-tune + window W=128	0.043	0.064	0.94	0.087 (0.374)	0.616	0.651	78	9.8
+ mixed 3-hop fine-tune, 2000 steps	0.027	0.254	0.88	0.005	0.623	0.660	54	14.2
+ 2000-step fine-tune + window W=128	0.022	0.065	0.94	0.200 (0.314)	0.623	0.660	78	9.8
+ window-aware fine-tune (800 steps, W=128 in the loss) + window W=128	0.025	0.059	0.94	0.277 (0.351)	0.620	0.651	78	9.8
+ window-aware fine-tune, 1850 steps + window W=128	0.023	0.056	0.95	0.386	0.625	0.652	78	9.8
+ mixed 3-hop fine-tune, format-neutral data	0.033	0.268	0.87	0.000	0.620	0.653	54	14.2
+ mixed 3-hop (format-neutral) + window W=128	0.027	0.071	0.94	0.107 (0.331)	0.620	0.653	78	9.8
LLA 21.3× (base codec)	0.057	1.249	0.66	0.000	0.622	0.668	36	21.3
+ exact window W=128	0.057	0.155	0.90	0.002	0.622	0.668	60	12.8
+ mixed 3-hop fine-tune, format-neutral data	0.043	0.645	0.75	0.000	0.617	0.654	36	21.3
+ mixed 3-hop (format-neutral) + window W=128	0.037	0.131	0.91	0.005 (0.109)	0.617	0.654	60	12.8
Ouro-2.6B-Thinking, LLA 4× (base codec)	0.032	0.177	0.87	0.406	0.662	0.749	384	4.0
+ exact window W=128	0.032	0.084	0.91	0.658	0.662	0.749	432	3.6
+ served-objective fine-tune, 1400 steps	0.010	0.067	0.93	0.732	0.666	0.748	384	4.0
+ 1400-step fine-tune + window W=128	0.010	0.067	0.92	0.786	0.666	0.748	432	3.6
+ window-aware fine-tune (600 more steps, W=128 in the loss) + window W=128	0.010	0.032	0.95	0.844	0.667	0.748	432	3.6

on a shuffled completion order reproduces the first: streaming KL on GSM8K continuations 0.062 vs 0.059, served GSM8K 0.522 vs 0.507; a second window-aware 4× run (seed 1, shuffled order) serves 0.649 against 0.674 with W=64. The oracle GSM8K of the fine-tuned 4× codec is 0.764 against 0.788 for the base codec (500 problems). Streaming-KL cells in Table 11 use 64 GSM8K questions where re-measured and the 16 of Table 3 otherwise; the two agree to within 0.01 at 4× and 0.02 at 14.2×. Fine-tuning on completions without boxed answers (the "format-neutral" rows) does not restore few-shot #### formatting at 14.2× or 21.3×. The "window-aware" rows keep the exact K/V of the last W generated tokens inside the served objective (the same ring the decoder uses), so the codec is fitted to the mixture of exact and latent context it will see at serving time; 800 steps from the base codec (600 from the 1400-step codec at 2.6B). These codecs are specialised to their window: served without it, the 4× codecs score 0.557 (1.4B) and 0.710 (2.6B). Longer served-objective training without the ring does not help: continuing the 2000-step 4× codec to 4000 steps leaves served GSM8K with W=64 at 0.592 (from 0.591), and continuing the 2.6B codec from 1400 to 2600 steps gives 0.712 without and 0.800 with W=128 (from 0.732 and 0.786; 500 problems). Continuing the window-aware 4× codec from 800 to 2800 steps gives 0.685 (from 0.674), within noise; at 14.2× the same continuation reaches 0.277 at 800 steps, 0.386 at 1850 and 0.337 at 2800 (checkpoints picked by the selection-set KL), so Table 11 reports the 1850-step checkpoint.

Oracle and served accuracy on identical problems. The oracle re-forward decoder is $O(n^2)$ at batch 1, so its GSM8K numbers use the first 500 test problems while served numbers use the full test set. Table 12 removes the subset difference by scoring the served generations on the same 500 problems; the served-minus-oracle gap is unchanged to within two points, and the fine-tuned codecs' own oracle accuracy is reported alongside so the remedies are not credited with oracle gains.

Table 12: **Oracle vs. served GSM8K on the same 500 test problems** (strict-match; flexible extraction in parentheses). 1.4B rows are 5-shot on the first 500 problems; 2.6B rows are 2-shot and, because the oracle re-forward runs at batch 1, compared on the first 200 of the same problems. n/a = not run.

model / codec	oracle	served decoder	served
Ouro-1.4B, LLA 4×	0.788	base codec	0.288 (0.304)
		2000-step fine-tune	0.558 (0.568)
		+ window W=64	0.612 (0.658)
		window-aware fine-tune + W=64	0.672 (0.676)
Ouro-1.4B, LLA 4× (fine-tuned codec oracle)	0.774	n/a	
Ouro-1.4B, LLA 14.2×	0.744	base codec	0.004 (0.030)
		base codec + window W=128	0.170 (0.192)
		2000-step fine-tune + W=128	0.192 (0.320)
		window-aware fine-tune + W=128	0.304 (0.376)
		window-aware fine-tune, 1850 steps + W=128	0.386 (0.398)
Ouro-2.6B-Thinking, LLA 4× (2-shot, first 200)	0.820	base codec	0.460 (0.475)
		base codec + window W=128	0.640 (0.660)
		1400-step fine-tune	0.715 (0.720)
		1400-step fine-tune + W=128	0.795 (0.810)
		window-aware fine-tune + W=128	0.855 (0.870)
Ouro-2.6B-Thinking, LLA 4× (fine-tuned codec oracle, first 200)	0.845	n/a	

For loop t, bilinearity moves the K up-projection to the query, $q_t^\top (W^K_{\mathrm{up},t} c_i^K) = ((W^K_{\mathrm{up},t})^\top q_t)^\top c_i^K$. Values likewise remain latent across cached positions and are expanded only after aggregation: $o_t = (\sum_i \alpha_{ti} c_i^V)(W^V_{\mathrm{up},t})^\top$. With Ouro-1.4B geometry (T=4, 24 layers, 16 heads, d_{head}=128) and bf16 caches, the per-layer payload ratio is 16384/1088=15.06×: 51.0 rather than 768 KiB/token across 24 layers. This payload accounting excludes allocator metadata and positional state.

Serving protocol. Ouro-1.4B runs at T=4 on a B200 reporting 183,359 MiB via `nvidia-smi`, with no additional headroom. All arms run the real, unmodified model forward end to end (embeddings, all $T \times D$ blocks, MLPs, LM head); only the attention read differs. The teacher keeps a preallocated full bf16 KV cache (no per-step concatenation) and uses the model's SDPA attention. Full-RoPE LLA keeps only (c_k, c_v) per token, head and layer and reads them with a fused Triton kernel that reconstructs $\hat{K}_t = c_k W^\top_{\mathrm{up},t} + \mu_t$, applies RoPE and the online softmax in on-chip memory, and aggregates $\sum_j p_j c_v(j)$ in latent space (one W^V_{up} per read); LLA-D keeps (c^C, k^R, c^V) per token and layer and reads them with a kernel that scores in latent space and serves all heads from one block load. "Max batch" is the largest entry of the fixed grid $[1, 2, 4, \ldots, 1024]$ whose allocation-plus-decode trial completes without CUDA OOM; fixed-batch decode is timed over 128 generated tokens after 8 warm-up steps, medians of five runs (min/max spread below 2%); prefill

latency is the CUDA-synchronized time to encode a real random prompt at batch 1, medians of three (Table 13). The decode steady state is measured over a fabricated context (its cost depends only on tensor dimensions). This protocol supersedes an earlier draft's attention-only microbenchmark with random codec weights; the numbers in the main text are those of the trained checkpoints. The same protocol on Ouro-2.6B-Thinking (48 layers, 1 536 KiB per token uncompressed) with its $4\times$ codec: at 4k the teacher does not fit batch 32 (max batch 16, peak 81 tok/s) while LLA decodes batch 32 at 283 ms/token (113 tok/s) with a 49.6 GiB store, max batch 96 and peak 201 tok/s ($2.5\times$ the teacher); at 16k the max batch is 4 for the teacher (23 tok/s) and 16 for LLA (44 tok/s).

Table 13: **Prefill latency** (s, batch 1, one B200). Latent stores are encoded with a two-pass chunked re-forward in unfused PyTorch (complete loop trajectories are needed before down-projection), which is the current bottleneck of the latent path; the teacher's prefill is a single forward.

method	4k	16k	64k	128k
teacher	0.1	0.8	6.1	20.2
LLA $4\times$	1.5	9.7	109.0	408.3
LLA $14.2\times$	1.4	9.2	105.2	395.1
LLA $21.3\times$	1.5	9.5	105.4	393.0
LLA-D $15.1\times$	1.3	11.1	141.8	545.5

Why the throughput and accuracy paths differ. RoPE prevents exact absorption of the positional component, so LLA-D uses an MLA-style NoPE content path plus an explicit rotary branch. For a frozen pretrained model this creates a post-hoc fidelity floor because its queries were trained against full-RoPE keys. Query adaptation improves the decoupled path, but does not match full reconstruction (Table 14); the layer-level LLA-D codec trained for Sec. 7 reaches held-out KL 0.20 at $15.1\times$, and its downstream cost is measured directly in Table 2. All other task-accuracy results use full-RoPE reconstruction.

Table 14: **Post-hoc decoupled-RoPE characterization** (Ouro-1.4B, $4\times$, d_{rope}=64). A learned query adapter improves the absorbed path, but no post-hoc configuration reaches the full-RoPE reference; a $d_{\text{rope}} \in \{32, 64, 96\}$ sweep preserves this ordering.

Configuration	held-out reconstruction KL ↓
full-RoPE reconstruction (accuracy path)	**0.059**
decoupled RoPE, no query adapter	0.55
decoupled RoPE + query adapter	0.34
decoupled RoPE + adapter, extended training	0.30

G SERVING: EVICTION COMPOSITION

Table 15: **Stress test with token eviction.** GSM8K (5-shot, strict EM) under an aggressive 260-token budget (sink 4 + window 256). The sequence-axis control includes 8-bit quantization; LLA-$4\times$ adds recurrence compression and remains close. This demonstrates compatibility in this degraded configuration, not a state-of-the-art eviction comparison or an isolated proof of axis independence.

Method (260-token eviction budget)	GSM8K strict ↑
eviction alone (kv_quant-8, sequence axis)	0.510
LLA $4\times$ + eviction (loop $\times$ sequence)	0.520

H ADDITIONAL FIGURES AND TABLES

This section collects the figures and tables referenced from the main text that support, but are not required to follow, the argument there.

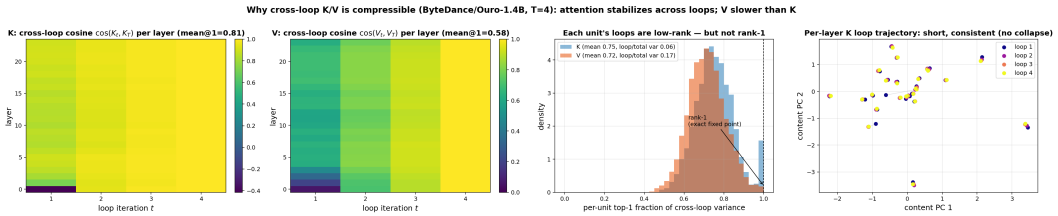


Figure 11: **Loop trajectories are low-rank, not collapsed** (frozen Ouro-1.4B). Cosines increase toward the final loop, while the PCA paths retain variation that endpoint reuse discards.

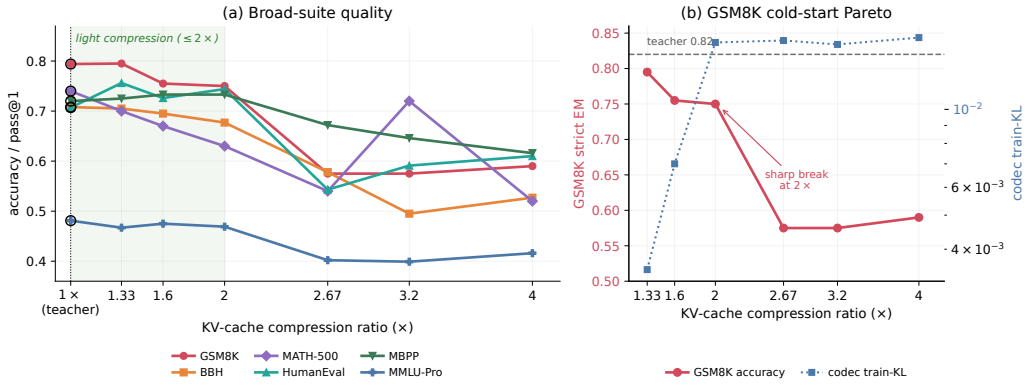


Figure 12: **Broad-suite quality as KV-cache compression increases** (Ouro-1.4B). Code matches or exceeds the teacher through 2× and MMLU-Pro changes modestly; GSM8K and BBH degrade gently in the light-compression region (shaded, $\leq$ 2×), whereas MATH-500 is more sensitive. HumanEval/MBPP report EvalPlus pass@1 (base).

Table 16: **Ouro-2.6B-Thinking broad-suite transfer.** Zero-shot loglikelihood accuracy is decoder-independent. HSwag, WGr, C-QA and OBQA denote HellaSwag, WinoGrande, CommonsenseQA and OpenBookQA. Per-head LLA (highlighted) stays near the uncompressed teacher at every budget.

Method	ARC-e	ARC-c	HSwag	WGr	C-QA	MMLU	PIQA	OBQA	Avg
teacher (1×)	0.833	0.541	0.568	0.684	0.752	0.750	0.785	0.310	0.653
compression 4×									
LLA	0.818	0.521	0.557	0.676	0.754	0.749	0.782	0.302	**0.645**
mla_head	0.733	0.460	0.494	0.681	0.723	0.740	0.729	0.234	0.599
LLA-2D	0.738	0.463	0.513	0.661	0.753	0.741	0.742	0.242	0.607
cla	0.756	0.484	0.518	0.676	0.647	0.725	0.750	0.260	0.602
kv_quant	0.820	0.517	0.565	0.677	0.738	0.747	0.782	0.292	0.642
compression 5.33×									
LLA	0.818	0.526	0.557	0.691	0.746	0.748	0.783	0.300	**0.646**
mla_head	0.751	0.474	0.504	0.660	0.731	0.747	0.739	0.244	0.606
LLA-2D	0.753	0.477	0.520	0.664	0.740	0.740	0.760	0.236	0.611
kv_quant	0.816	0.519	0.550	0.663	0.712	0.719	0.777	0.292	0.631
compression 10.67×									
LLA	0.813	0.526	0.552	0.676	0.741	0.745	0.780	0.288	**0.640**
mla_head	0.730	0.461	0.498	0.668	0.726	0.747	0.725	0.256	0.601
LLA-2D	0.786	0.512	0.539	0.672	0.745	0.742	0.761	0.294	0.631

Table 17: **Huginn-3.5B fidelity** (R=32). Top-1 uses SVD at 16–32× and distillation at 64–128×; full metrics: Appendix Table 7.

ρ	SVD KL ↓	distill KL ↓	top-1 ↑
16×	.005	n/a	.970
32×	.010	n/a	.954
64×	.197	**.145**	.834
128×	.763	**.177**	.812

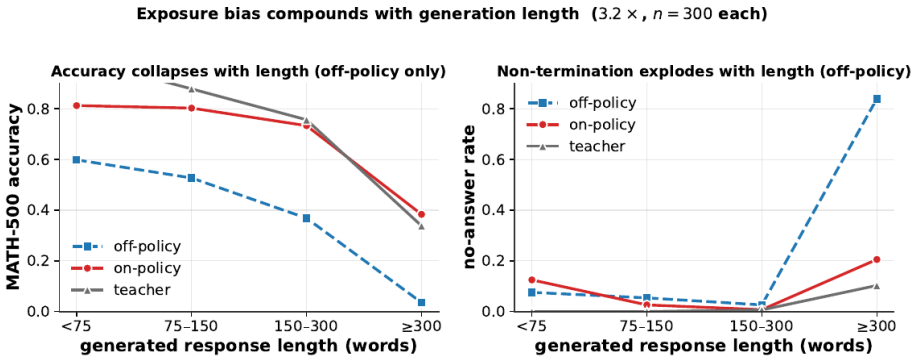


Figure 13: **Exposure bias compounds with response length** (Ouro-1.4B, 3.2×). On-policy refinement tracks the teacher across length bins; off-policy accuracy falls and its no-answer rate spikes.

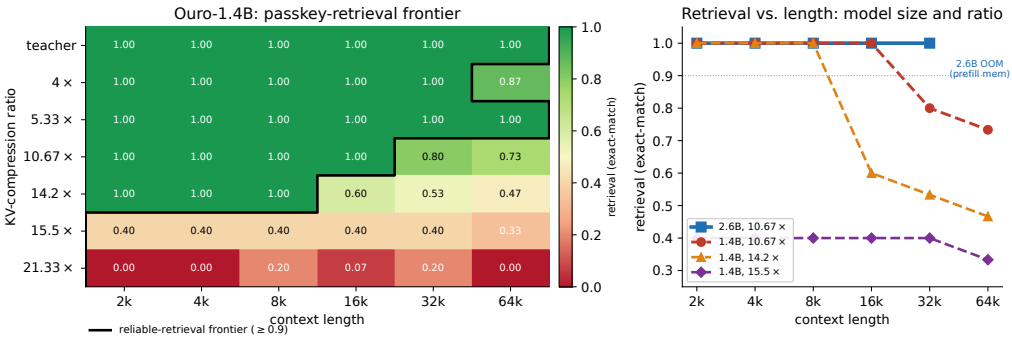


Figure 14: **Long-context passkey retrieval under KV-cache compression** (15 trials/cell). *Left:* exact retrieval over the compression–context grid; the staircase is the ≥ 0.9 reliable-retrieval frontier. *Right:* the length-limited and compression-limited failure regimes, plus the 2.6B model-size benefit.

Table 18: **Serving across context length** (same protocol). For each context: decode tok/s at batch 32 (OOM = the batch does not fit), then max batch / aggregate tok/s at that batch. The teacher's capacity collapses to a single sequence at 128k while the latent stores still batch.

	4k		16k		64k		128k	
method	B=32	max / tok/s	B=32	max / tok/s	B=32	max / tok/s	B=32	max / tok/s
teacher	178	48 / 188	OOM	8 / 51	OOM	2 / 13	OOM	1 / 6
LLA 4×	210	192 / 481	109	48 / 122	OOM	8 / 28	OOM	4 / 14
LLA 14.2×	223	768 / 933	137	192 / 187	46	48 / 47	OOM	16 / 22
LLA 21.3×	207	1024 / 1102	176	256 / 279	65	64 / 70	35	32 / 35
LLA-D 15.1×	214	768 / 3994	211	192 / 1124	219	48 / 297	OOM	16 / 110